Training parsers for low-resourced languages: improving cross-lingual transfer with monolingual knowledge

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'Université Paris-Sud

École doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Orsay, le 6 avril 2018, par

Mme Lauriane Aufrant

To phh.

Abstract

***

As a result of the recent blossoming of Machine Learning techniques, the Natural Language Processing field faces an increasingly thorny bottleneck: the most efficient algorithms entirely rely on the availability of large training data. These technological advances remain consequently unavailable for the 7,000 languages in the world, out of which most are low-resourced.

One way to bypass this limitation is the approach of cross-lingual transfer, whereby resources available in another (source) language are leveraged to help building accurate systems in the desired (target) language. However, despite promising results in research settings, the standard transfer techniques lack the flexibility regarding cross-lingual resources needed to be fully usable in real-world scenarios: exploiting very sparse resources, or assorted arrays of resources. This limitation strongly diminishes the applicability of that approach.

This thesis consequently proposes to combine multiple sources and resources for transfer, with an emphasis on selectivity: can we estimate which resource of which language is useful for which input? This strategy is put into practice in the frame of transition-based dependency parsing.

To this end, a new transfer framework is designed, with a cascading architecture: it enables the desired combination, while ensuring better targeted exploitation of each resource, down to the level of the word. Empirical evaluation dampens indeed the enthusiasm for the purely cross-lingual approach – it remains in general preferable to annotate just a few target sentences – but also highlights its complementarity with other approaches. Several metrics are developed to characterize precisely cross-lingual similarities, syntactic idiosyncrasies, and the added value of cross-lingual information compared to monolingual training. The substantial benefits of typological knowledge are also explored.

The whole study relies on a series of technical improvements regarding the parsing framework: this work includes the release of a new open source software, PanParser, which revisits the so-called dynamic oracles to extend their use cases. Several purely monolingual contributions complete this work, including an exploration of monolingual cascading, which offers promising perspectives with easy-then-hard strategies.

Résumé

***

Le récent essor des algorithmes d'apprentissage automatique a rendu les méthodes de Traitement Automatique des Langues d'autant plus sensibles à leur facteur le plus limitant : la qualité des systèmes repose entièrement sur la disponibilité de grandes quantités de données, ce qui n'est pourtant le cas que d'une minorité parmi les 7.000 langues existant au monde.

La stratégie dite du transfert cross-lingue permet de contourner cette limitation : une langue peu dotée en ressources (la cible) peut être traitée en exploitant les ressources disponibles dans une autre langue (la source). Les progrès accomplis sur ce plan se limitent néanmoins à des scénarios idéalisés, avec des ressources cross-lingues prédéfinies et de bonne qualité, de sorte que le transfert reste inapplicable aux cas réels de langues peu dotées, qui n'ont pas ces garanties.

Cette thèse vise donc à tirer parti d'une multitude de sources et ressources cross-lingues, en opérant une combinaison sélective : il s'agit d'évaluer, pour chaque aspect du traitement cible, la pertinence de chaque ressource. L'étude est menée en utilisant l'analyse en dépendance par transition comme cadre applicatif.

Le coeur de ce travail est l'élaboration d'un nouveau méta-algorithme de transfert, dont l'architecture en cascade permet la combinaison fine des diverses ressources, en ciblant leur exploitation à l'échelle du mot. L'approche cross-lingue pure n'étant en l'état pas compétitive avec la simple annotation de quelques phrases cibles, c'est avant tout la complémentarité de ces méthodes que souligne l'analyse empirique. Une série de nouvelles métriques permet une caractérisation fine des similarités cross-lingues et des spécificités syntaxiques de chaque langue, de même que de la valeur ajoutée de l'information cross-lingue par rapport au cadre monolingue. L'exploitation d'informations typologiques s'avère également particulièrement fructueuse.

Ces contributions reposent largement sur des innovations techniques en analyse syntaxique, concrétisées par la publication en open source du logiciel PanParser, qui exploite et généralise la méthode dite des oracles dynamiques. Cette thèse contribue sur le plan monolingue à plusieurs autres égards, comme le concept de cascades monolingues, pouvant traiter par exemple d'abord toutes les dépendances faciles, puis seulement les difficiles.

Remerciements

***

Avant de remercier ceux qui ont rendu ces dernières années formidables, mes premières pensées vont d'abord à ceux qui les ont rendues possibles, en premier lieu Édouard Geoffrois – une rencontre fortuite qui a donné un tournant inattendu à mon cursus – et la DGA, qui a financé cette thèse, grâce au soutien actif de Laura C., Philippe L., Jérôme 3D et Sandrine C.

Merci également à Pierre, Benoît, Anders et Xavier, qui ont accepté de clore cette étape de ma vie en participant à mon jury de thèse.

Si j'écris ces lignes aujourd'hui, c'est aussi grâce à Guillaume et François, qui m'ont encadrée avec bienveillance tout au long de ce voyage, de mon premier jour de stage à la soutenance de mon doctorat, et plus encore. Ensemble, nous avons discuté, réfléchi, compris, et pas à pas, parfois l'un, parfois l'autre, parfois les deux, ils m'ont guidée dans la réalisation de ce qui allait devenir ma thèse. Je la leur dois.

J'ai eu la chance de bénéficier de conditions de travail idéales au sein du LIMSI, une institution d'excellence animée par des individus brillants, venant d'horizons variés, et je suis fière d'y avoir appartenu, ne serait-ce que quelques années. Au sein de la thématique Traduction, que ce soit en groupe de lecture ou en salle café, les idées fusent, les projets se montent, et déjà les expériences ont tourné ! J'ai énormément appris de ces échanges.

Le quotidien du laboratoire, c'est aussi le va-et-vient de ses thésards. Je suis dès le début restée pantoise devant mes prédécesseurs, Li, Thiago, Quoc Khanh, Benjamin, Nicolas, Yong... des scientifiques accomplis, qui m'ont fait entrevoir tout le chemin qu'il me restait à parcourir avant d'atteindre, moi aussi, un jour, cette fameuse soutenance. Ma gratitude va également à Julia et Elena pour un aperçu haut en couleurs de la Russie, à Lucile pour m'avoir faite rire, à ma "promo" – Kevin, Pierre et surtout Matthieu, pour ces années de bureau partagé – ainsi qu'aux plus jeunes, Rachel, Yuming, Pooyan et Caroline. Ces derniers mois j'ai vu arriver une toute nouvelle génération : Benjamin, Aina, Syrielle, José, Margot, je vous remercie pour le renouveau que vous apportez et je vous souhaite bonne chance, la route peut parfois sembler longue mais le résultat vaut vraiment le coup ! Merci aussi à celles qui ont tenu bon face à mes incitations à la thèse, Charlotte et Catherine.

Je dois également beaucoup à nos postdoctorants. C'est Ophélie qui m'a fait découvrir combien la science se faisait mieux à plusieurs. Un grand merci à Franck pour nos pauses, pour son amour du roumain, pour nos discussions animées... et surtout parce que c'est un réginaburgien ! Quant à Guillaume, il a rendu plus d'une conférence mémorable. Mes pensées vont également aux postdoctorants ILES, Lionel, Eva, François, ainsi qu'à toute l'équipe de Bring Your Own Merguez et autres instants ludiques limsiens.

Merci aux 2 gares et à tous ceux qui s'y sont pressés, pour refaire le monde ou juste parler de science.

Les permanents ne sont évidemment pas en reste : Hervé, Claude, Philippe, Ioana pour ses bonnes leçons de roumain, Cyril, Thomas, Marianna la dynamique, Hélène l'engagée, Alexandre pour être lui-même, Éric pour sa réactivité... Anne Vilnat a dirigé le pôle doctoral avec naturel et bienveillance, et Laurence Rostaing n'a eu de cesse de m'aider dans mes démarches, missions, demandes, toujours avec le sourire.

Tous les membres de ParSiTi et anciens d'Alpage méritent de même ma gratitude, au titre de l'important soutien scientifico-culinaire qu'ils m'ont prodigué. Mes remerciements vont aussi à mes camarades en thèse dans d'autres domaines, Simon, Alice, Cyprien, Martin, Victor et Adrien, pour leur présence et pour l'ouverture qu'ils m'ont apportée. Je n'oublie pas non plus ceux qui ont été là et ne le sont plus.

Merci encore à Matthieu, Ophélie, Franck, Guillaume et François, et enfin à ma famille, qui a su être là au bon moment, en particulier Micheline qui m'a vue écrire la première et la dernière lignes de mon manuscrit. Et évidemment à phh, qui a été, avant, pendant et après, tout simplement parfait.

Contents

***

List of Tables

***

List of Figures

***

Glossary

***

Abbreviations

DCA	**D**irect **C**orrespondence **A**ssumption (see Hwa et al., 2002)
KL	**K**ullback-**L**eibler divergence
LAS	**L**abeled **A**ttachment **S**core
MST	**M**aximum **S**panning **T**ree
MT	**M**achine **T**ranslation
NLP	**N**atural **L**anguage **P**rocessing
OOV	**O**ut-**O**f-**V**ocabulary words
PoS	**P**art **o**f **S**peech
UAS	**U**nlabeled **A**ttachment **S**core
UD	**U**niversal **D**ependencies (see Nivre et al., 2016)
UDT	**U**niversal **D**ependency **T**reebank (see McDonald et al., 2013)
UPOS	**U**niversal **P**art **o**f **S**peech (see Petrov et al., 2012)

Definitions

Action	See transition.
Beam search	A strategy for structured prediction which maintains a set of hypotheses and explores the search space from there.
Cascading	An ensembling method based on a sequential pool of classifiers, where each input is annotated by exactly one stage.
Dependency class	Any typological criterion describing a subset of dependencies in a treebank. It can be based on length, depth, edge direction, relation label, PoS tags, empirical values (e.g. frequency), etc. The notion extends to the child: in this context, the class of a token is the dependency class of its attachment.
Dependency depth	The distance in the tree from the dependency child to the ROOT token: root attachments have depth 1.

Dependency length	The distance in the sequence from the dependency child to the head: neighbouring attachments have length 1.
Derivation	A transition sequence starting in an empty configuration (with empty stack/list and full buffer).
Dynamic oracle	A complete and non-deterministic oracle which computes gold actions tailored to the current configuration.
Epoch	One complete traversal of the training data, using each sentence in turn as training example. It is the basic unit for cross-validation on a development set.
Feature template	Specification of feature tuples, used instead of simple features to compensate for the linearity of the classifier.
Feature vector	The set of features extracted from a given input or configuration and provided to the classifier.
Iteration	See epoch.
Model	The weight vector resulting from training.
Oracle	The training component of a transition-based parser which computes the gold action to apply in each training configuration and compares it with the actually predicted action, to detect errors and trigger model updates.
Parser configuration	The stack (or lists), buffer, transition history and enriched input (word sequence, tags and partial tree), corresponding to a partial analysis of a sentence. It is functionally equivalent to the (unique) derivation resulting in this configuration.
Parser state	All parser parameters related to the input at a given time: the parser configuration in greedy search, the beam of hypotheses (i.e. a set of parser configurations) in beam search.
Postposed	Qualifies words which appear after their head.
Preposed	Qualifies words which appear before their head.
Source	The well-resourced language whose knowledge is used to process the target language.
Static oracle	A deterministic oracle which is based on the precomputation of a canonical reference derivation, before processing a training example.
Weight vector	The set of parameters indicating the score corresponding to each feature, for each possible decision. In linear models, the score of a decision is obtained by a scalar product between the feature vector and the corresponding weight vector.
Target	The low-resourced language of interest.
Transition	An operation on a parser configuration, which affects its stack/lists, buffer and partial tree to produce a new configuration. Even when notations do not emphasize it, transitions are to be considered as relative to a given configuration (e.g. SHIFT when buffer head has index 3 means SHIFT_3).

Transition sequence	The series of transitions followed from one parser configuration to another.
Word alignment	The word-level pairing of sentence translation pairs, in bilingual corpora. Depending on the method, on both sides each word can be aligned to 0, 1 or several words in the other language.

Notations

$\overset{\curvearrowleft}{\mathrm{P}}$	The class of preposed words with PoS P.
$\overset{\curvearrowright}{\mathrm{P}}$	The class of postposed words with PoS P.
$\mathrm{P}\overset{\curvearrowleft}{\ }\mathrm{P}_h$	The class of preposed words with PoS P and head PoS P_h.
$\mathrm{P}_h\overset{\curvearrowright}{\ }\mathrm{P}$	The class of postposed words with PoS P and head PoS P_h.
$\mathrm{PR}\left[\mathrm{P}, \mathrm{P}_h\right]$	The proportion of preposed words among those with PoS P and head PoS P_h.
$\mathrm{UAS}\begin{bmatrix}\mathrm{C}_h \\ \mathrm{C}\end{bmatrix}$	The proportion of correctly attached words among those in class C and whose head is in class C_h.
$c \circ t$	The successor configuration resulting from following transition t in parser configuration c.

Chapter 1

Introduction

***

Contents

There are 7,000 languages in the world – only a few of them have access to Natural Language Processing. Thanks to recent progresses in the Machine Learning field, the last few years have seen the rise of new usages of those technologies: nowadays, numerous devices based on Natural Language Processing (NLP) are used daily in personal and professional life, now even at home, and yet this is a privilege that only a handful of linguistic areas can enjoy. The main reason for that situation is the resource bottleneck: while Machine Learning algorithms improve by exploiting always larger amounts of data, the development of such resources is in general conditioned to financial and political support. Obviously, not all languages in the world have benefited from such support and to date, most of them remain low-resourced, that is, the amount of data available in those languages is insufficient to build accurate NLP systems.

With the military concerns that affect the French *Direction Générale de l'Armement*, which funded this work, the lack of resources becomes even more sensitive, and operational challenges add up. First, the interests of such institutions are international, and change over the years. It is thus not realistic to expect them to organize, for each one of the focus languages in turn, large campaigns of data collection, as local or cultural organisations specialized in a single language would do. Second, the languages of interest are unrelated to the size of the corresponding populations. Depending on the chain of events, very rare languages may come under focus, in which case human resource issues arise: even if resource acquisition was considered, it can be difficult to find a speaker of that particular language or dialect, able to provide the linguistic annotations that are crucial to develop resources actually usable by NLP algorithms.

Last, but not least, the urge to set up NLP systems in very short time, for instance when a new conflict has just broken out, rules out annotation campaigns anyway. As a matter of fact, similar challenges are faced when using NLP to overcome the language barriers in any context where every minute counts, like the humanitarian response to natural disaster.

Among the scientific attempts to overcome these difficulties, the recent years have seen the emergence of a new approach, *cross-lingual transfer*. The idea is to leverage existing resources in another language, as a substitute to the missing resources in the desired language. The information available on one language, the source, can indeed be *transferred* to another language, the target, by exploiting either bilingual data or linguistic similarities. Accurate NLP systems can then be built in any language of interest, notwithstanding its lack of resources.

This thesis adopts the cross-lingual transfer approach and explores ways to maximize its efficiency in real-world scenarios: its goal is to increase the usability of the various available resources, regardless of their diversity, size and sparsity.

1.1 Maximizing the usability of existing resources

The literature of cross-lingual transfer has proposed a large variety of concrete methods, each one using a different kind of cross-lingual resource: one exploits large bilingual corpora, others rather rely on linguistic similarities, while some prefer using bilingual lexicons.

However, in practice, the situation regarding resources can be more blurry than the research scenarios: available data will likely be much more diverse, but also way sparser. It may happen that, at the same time, no well-resourced language is close enough to leverage their similarities, while bilingual data is scarce. For instance in Chechen, which is only surrounded by low-resourced languages, publicly available bilingual corpora with any language amount to 500 Chechen words only, far from the 60 *millions* used in common experimental settings based on French or German. On the other hand, it is rare to be completely devoid of any resource in a given language: one often knows a few linguistic facts about that language, a couple of words, or there exists a sample of just a dozen annotated sentences. It is for instance documented online,[1] in a computer-readable format, that Chechen indicates nominal plurals with suffixes, has no article, organizes the sentences in a Subject-Object-Verb (SOV) order, uses postpositions and places relative clauses before the noun; a handful of annotated Chechen sentences can additionally be found in published works in linguistics (Conathan and Good, 2000; Good, 2003). All that information is valuable, and raises the hopes for granting such languages with access to NLP technologies. Yet, most existing works disregard such mixed scenarios, focusing instead on cases

[1] http://wals.info

where either the cross-lingual or the monolingual data is large; datasets as tiny as a few sentences are rarely considered.

For the *Direction Générale de l'Armement*, the matter is again especially delicate. Because the military is subject to specific operational constraints, sometimes it is vital to understand a given sentence, so that it has to be processed by any possible means: giving up on that sentence, which is what other institutions or companies would do when even transfer approaches fail, is not an option. In such cases, every additional resource is worth it, in order to make the best out of the situation.

In the light of these real-world needs, I have chosen to design a transfer method that is able to exploit all sources of information at one's disposal, without any assumption on the amount of each resource, or the availability of a given resource.

1.2 Problem statement: multi-(re)source combination

Algorithmically, my approach comes down to one key question: how to leverage multiple sources of sparse cross-lingual information?

There is already a large body of research on the combination of multiple source languages, and I pursue on that track, but I also extend it to the topic of combining multiple *types* of resources – this has already be done as well, but only for specific sets of resources and not in a systematic fashion. In the end, their sparsity just adds to the challenge.

More specifically, my proposal aims at a *selective combination* of those tidbits of information, extracting from each resource what is especially useful about it. Indeed, not all information provided by a given source language is uniformly as relevant for the target: for instance, knowing the English conjugation system, consisting in just a few wordforms, is of no use to process a language with rich morphology, like French, where each verb corresponds to dozens of wordforms. Conversely, English belongs to the Germanic language family, but at the same time has been strongly influenced by its Romance neighbours, which results in numerous cognates with both families, i.e. with plenty of languages and dialects. As such, valuable knowledge can be extracted from these lemmas specifically, if identified. I believe that measuring this kind of phenomena will maximize the benefits drawn from such resources.

1.3 Dependency parsing as a case study

The cross-lingual transfer method I propose is thought and designed for any sequence labeling task in the NLP field. However, for the sake of illustration and empirical evaluation, I conduct most of my work on a specific task: syntactic parsing. More precisely, I use the *dependency parsing* formalism and the popular technique of *transition-based parsers*. A whole chapter is notably dedicated to specific improvements of parsing, but these are in fact technicalities of my main method, and similar contributions could be made for other tasks.

The choice of the parsing task is interesting in several regards. First, as it consists in structuring a natural language input, parsing constitutes the first step of many methods for more complex tasks like Machine Translation, Information Retrieval or Sentiment Analysis, and as such has downstream benefits for all those. Parsing is also more challenging than other annotation tasks like Part-of-Speech (PoS) tagging, in the sense that the annotations to produce are complex (the output is said *structured*) and that the parsing algorithms are tailored to that expectation. This property can make transfer arduous, but also more interesting.

Another motivation to explore dependency parsing is the availability of substantial experimental data. Indeed, most of the recent efforts regarding cross-lingual transfer have in fact been federated by the *Universal Dependencies* project, henceforth UD (Nivre et al., 2016). The purpose of that project, whose origins date back to 2013, is to build a multilingual dataset with cross-lingually consistent dependency annotations, a property which has been noted as crucial for proper evaluation of cross-lingual transfer. This initiative has succeeded in bringing together the community, which now works together for developing, and constantly improving, a rich and diverse dataset for experimental evaluation of cross-lingual methods – at least for the task of dependency parsing. Covering initially just a handful of languages, the dataset now contains 73 treebanks, covering 54 very diverse languages. It is consequently a natural choice to work on dependency parsing, thereby benefiting from this huge annotation effort, which notably allows me to perform large-scale multilingual experiments and to assess my results on various languages in a systematic manner.

As a result, most of the experiments conducted throughout this thesis are based on UD data. However, because UD is a work in progress, it has evolved a lot during the time of my thesis work, and I consequently use different versions at different points of the thesis (UDT 2.0, UD 1.3, UD 2.0, and the CoNLL 2017 version of UD 2.0), which vary in size and specifications. The document also contains a number of linguistic examples, most of which are drawn from UD 2.0. When referring explicitly to PoS tags or relation labels, I follow UD specifications as well: for instance DET and ADJ stand for determiner and adjective, ADP for adposition, `nsubj` for nominal subject.

1.4 Contributions

The contributions of my work are of three kinds: one main contribution, a few side contributions, and a series of incidental findings and improvements which slightly stray from the core of this work but remain valuable.

My main contribution is also the common thread of this thesis:

- With the goal of maximizing the usability of the sparse resources available for low-resourced languages, I propose a new framework for cross-lingual transfer in general. Using a cascading architecture, my method enables a finer combination of multiple sources and resources: its high robustness to sparse and non-standard data ensures that no valuable resource is overlooked, and at the same time each source is used first and foremost on its area of expertise. This framework has also a wide array of applications outside of cross-lingual transfer, and thus opens new research avenues.

The topics I have additionally contributed to range from parsing to cross-lingual transfer, including conceptual matters:

- I release a new dependency parser, PanParser, which is a contribution in three regards: algorithmically (designing new ways to train and use parsers), but also in terms of formalism (unifying several previous frameworks) and implementation (with an emphasis on modularity).
- The efficiency of the transfer approach is reassessed: on one hand I uncover that even promising state-of-the-art methods in that field fail to outperform the straightforward approach of training on a monolingual sample; on the other hand I reveal and quantify the fact that both approaches learn different kinds of knowledge.
- I propose a way to incorporate linguistic knowledge into the transfer process, thereby increasing drastically the amount of languages covered by this approach.
- An in-depth reflection is conducted on what differentiates two languages, their typological and structural differences, together with empirical ways to identify their common syntactic ground.

The remaining contributions of this work cover a variety of aspects. First, my experiments provide a better understanding of the inner workings of parsers, and notably of low-level interactions in feature-based systems. In that context, I develop a complexity metric to characterize what can be accurately learned in a sample of data. I also investigate ways to control the generalization capacity of both taggers and parsers, by carefully adapting the lexicalized part of their feature set. Regarding the standard training procedures of beam parsers, uncovering a sampling bias toward the beginning of sentences leads me to devising advanced training strategies that improve train-test consistency. The use of non-projective training data, which is usually discarded, rewritten, or handled with dedicated algorithms, is also enabled with very simple and transparent means, outperforming a popular method to exploit them. I additionally revisit the result whereby the ArcEager system is arc-decomposable, showing that this property does not always hold, depending on the position of the ROOT token, but this issue is solved at once by the proposal of a dynamic oracle tailored to that case. Finally, I have paved the way to achieve cross-lingual transfer of bilingual knowledge, in the case of word alignment models.

Research activities In the course of this thesis work, I have taken part in two evaluation campaigns: the *WMT 2016 News Translation Shared Task* and the *CoNLL 2017 UD Shared Task*. While Machine Translation is not the focus of my work, that year Romanian was among the evaluated languages. Having investigated the morphosyntactic idiosyncrasies of Romanian for transfer purposes, I thus capitalized on that work to participate to my team's joint submission to the WMT task. In that context, I have developed and released two new resources for Romanian processing, a crowd-sourced lexicon of PoS and morphological tags and a rule-based tokenizer, tokro.

The CoNLL task was organized as part of the UD project: using a newly released version of the UD dataset, it consisted in a large-scale evaluation of parsers in a wide array of languages. Part of the task was to process low-resourced and surprise languages, therefore leveraging the other well-resourced treebanks, in a cross-lingual or multilingual fashion. I consequently took this opportunity to assess the efficiency of my transfer method proposal, in realistic conditions.

All resources and software developed in the context of my thesis are publicly available.[2]

Publications At the time of the writing, my contributions have lead to 12 publications. Six of them concern the core of my thesis, cross-lingual parsing: one is a case study on multi-source transfer to Romanian (Aufrant et al., 2016b), further investigated by quantifying dependency complexity (Aufrant et al., 2018b); two show how a better parsing framework can make transfer through parallel data painless (Lacroix et al., 2016b; Lacroix et al., 2016a); another tackles a zero-resource scenario and proposes ways to incorporate linguistic knowledge into the parser transfer process (Aufrant et al., 2016d); and the last publication corresponds to my participation in the CoNLL shared task (Aufrant and Wisniewski, 2017). The remaining six address side projects I have conducted in the course of my thesis work, regarding advanced parsing techniques (Aufrant et al., 2016c, 2017, 2018a), the WMT shared task (Allauzen et al., 2016; Peter et al., 2016) and the transfer of bilingual knowledge, and in particular alignment models (Aufrant et al., 2016a).

[2]`https://perso.limsi.fr/aufrant`

1.5 Outline

The first part of this document presents the cross-lingual transfer approach and reviews the state-of-the art methods for transferring, parsing, and transferring parsers. It then takes an analytical turn, with an empirical investigation of the strengths and weaknesses of state-of-the-art cross-lingual parsers; the purpose of that part is to identify their major shortcomings and thus to develop the design guidelines of the new transfer framework that I propose. This framework is put into practice in a third part, which starts with technical contributions regarding parsers and data transformation methods, and concludes by presenting concrete uses of the transfer framework and the associated metrics, as well as an extensive empirical assessment of its success.

The document is accompanied by three appendices, which report additional works I have conducted on topics that slightly differ from the core of my proposal.

Part I – The state of the art in cross-lingual parsing

- **Chapter 2 – Cross-lingual transfer and multilinguality.** This chapter draws a typology of the main approaches for cross-lingual transfer, with an emphasis on new trends involving multilingual systems and how they relate to the traditional cross-lingual literature. It is also the occasion to draw a picture of the available resources, out of which many are often overlooked.
- **Chapter 3 – Recent advances in transition-based dependency parsing.** In this chapter, I present the transition-based approach to dependency parsing, as well as a technical description of the few most popular algorithms. I also review how that field has been greatly improved in the recent years, along three lines of research: improving the underlying classifier, as well as beam search and dynamic oracles. A detailed analysis reveals the complementarity of the last two, even though their combination has been hardly studied in the literature.
- **Chapter 4 – Cross-lingual parsing.** Based on the aforementioned typology, this chapter describes concrete methods for cross-lingual parsing, as well as specific issues raised by that application. It also recounts the history leading to the UD project, primarily motivated by the annotation consistency requirement. The main conclusions I draw from that review are the rigidity of existing methods regarding the exploitation of resources, as they poorly adapt to their expected sparsity and diversity in real-world scenarios, as well as the lack of fine-grained combination of multiple source languages.

Part II – Cross-lingual parsing for low-resourced languages: an empirical analysis

- **Chapter 5 – How good are we at cross-lingual parsing?** This chapter opens with a negative result on the success of transfer: using even advanced transfer techniques, a multi-source cross-lingual parser fails in Romanian to outperform a monolingual parser trained on just a sample of sentences. This result is confirmed on a wide array of languages, but further investigation hints at the complementarity of both approaches. Motivated by linguistic examples, I therefore draw a typology of syntactic knowledge, distinguishing language-specific and shared content. A new notion is introduced on that basis: the relevance of a piece of data for the task at hand.
- **Chapter 6 – Heterogeneity in parsing: monolingual and cross-lingual issues.** This chapter gathers a series of empirical studies conducted at a fine-grained level, shown to be crucial for accurate estimation of parser usefulness. Investigating the impact of the diversity of dependencies, it exhibits several interaction effects, based on a variety of measures (from feature-level monitoring to accuracy measures), diverse analysis techniques (data transformation, preinitialization and modified learning procedure) and focusing on three aspects (lexicalization, difficulty and imbalance). The results motivate the choice of a cascading architecture, whereby each language (both the sources and the target) is assigned a subset of dependencies and forwards partially processed inputs to the next stage.

Part III – A new framework for cross-lingual transfer

- **Chapter 7 – Developing a more flexible parsing system.** As my cascading proposal requires to parse in conditions that are out of the scope of standard parsers, I develop several extensions of parsing specifically. This chapter revisits dynamic oracles to extend their applicability to arbitrary data and beam parsers, and shows how they can be used to handle partial trees in several ways (as training material, output, or dependency constraints). The main outcome of this chapter is a unified parsing framework under which most state-of-the-art algorithms can be gathered, an idea which I implement and release under the name PanParser.
- **Chapter 8 – Incompatible representations of knowledge.** This chapter calls into question the common assumption that dependencies governed by the same syntactic rules in source and target are accurately transferred. Empirical analysis reveals indeed severe transfer failures associated with word order divergences. By proposing two techniques that reshape source data to better

match target regularities, I show how linguistic knowledge can be successfully incorporated in the transfer process, thereby addressing such issues.

- **Chapter 9 – Cascading models for multi-(re)source selective transfer.** In this chapter, I provide concrete ways to use a cascading framework for cross-lingual transfer, and notably study how to measure the relevance of each source dependency for processing the target. Several perspectives for using that framework in monolingual parsing are also presented. My whole work is then put into practice in realistic conditions through a shared task evaluation, which validates my main approach, as well as several other contributions.

Appendices

- **Appendix A – Advanced strategies for global training in parsing.** This appendix uncovers a sampling bias in the training procedure of beam parsers, whereby the beginning of the derivation is over-represented in comparison to the end. Based on my extension of dynamic oracles to beam parsers, I propose a simple modification of the global training strategies, which restores the balance between both parts of the sentence and improves the accuracy on the end of the sentence. The appendix also reports empirical evidence of a better train-test sampling consistency, as well as an improvement of convergence time.

- **Appendix B – Word alignment transfer.** In this appendix, I tackle the transfer of a new kind of information, bilingual knowledge, thereby alleviating the thorny issue of parallel data quality: since word alignment is usually done in an unsupervised way, the smaller a parallel corpus is, the less accurate its alignments are, and so will be the systems transferred through that data. Hence, transfer to the lowest-resourced languages, for which even parallel data is scarce, is impacted twice: there is less training material, and it is of poorer quality. I propose several methods to address this bottleneck via alignment transfer, and evaluate them with both intrinsic and extrinsic metrics, in a context of tagger transfer.

- **Appendix C – Leveraging lexical similarities: zero-resource tagger transfer.** Direct lexicalized transfer is explored in this appendix, in the case of PoS tagging. Evaluation on a large number of closely related language pairs yields promising results regarding the exploitation of lexical similarities, further improved by feature-level refinements to ensure proper generalization.

Part I

The state of the art in cross-lingual parsing

Abstract

The first part of this thesis presents successively the state of the art in cross-lingual transfer, transition-based dependency parsing, and the literature at the crossroads of both fields: cross-lingual parsing. It itemizes the various sources of information available for cross-lingual transfer and draws a thorough typology of the main transfer approaches. Several transition-based algorithms for dependency parsing are presented, as well as their recent improvements on three aspects: classifiers, inexact search and training oracles. Concrete methods for cross-lingual parsing are described and put into perspective with the proposed typology of resources; most kinds of available resources appear to have been exploited by the cross-lingual parsing literature, but not together. Specific comments on the difficulty of fair evaluation and on the need for cross-lingually consistent treebanks like the Universal Dependencies complete this review.

Chapter 2

Cross-lingual transfer and multilinguality

***

Contents

Cross-lingual transfer is a strategy to overcome low-resource scenarios, which leverages knowledge already acquired in one or several *source* languages to help building accurate models in a *target* language. To implement this promising idea, three aspects need to be addressed: which language(s) to use as a source, what knowledge to extract from it, and how to incorporate this knowledge into target models.

Section 2.1 addresses the first two questions by examining and categorizing the various resources and properties that can act as a bridge between two languages, while Section 2.2 presents the main approaches proposed to answer the third question. Section 2.3 completes this picture by describing new trends which have recently emerged in the literature, and Section 2.4 mentions practical issues regarding evaluation.

2.1 Cross-lingual bridge: an attempt at an exhaustive typology

From a broad perspective, two languages are two independent objects, and processing one has nothing to do with processing the other. Knowledge transfer between both languages is thus out of question without some kind of cross-lingual input to build a bridge between those languages: this binding can be either explicit (a resource), like a bilingual lexicon, or implicit (a property), in which case transfer is merely based on a reasonable expectation of encountering similar patterns on both sides. I consider several types of such cross-lingual bridges, which I present below, starting with the most popular ones.

Parallel data Bilingual corpora, or when not available, comparable corpora, provide an obvious, and more often useful than not, bilingual binding. They exist in various forms, ranging from large parallel data on a specific pair, to succinct but massively multilingual corpora like the Bible (Mayer and Cysouw, 2014; Christodouloupoulos and Steedman, 2015). The OPUS platform (Tiedemann, 2012) hosts such data for a wide array of languages.

When considering such resources, the literature often uses English as a source, on the grounds that this very well-resourced language shares vast amounts of parallel data with many languages. But bilingual data with any language may be leveraged, even one with very different properties, as the parallelism ensures the cross-lingual binding in a wordform- and order-agnostic way.[1]

Several state-of-the-art works rely on a machine translation system as an alternative to parallel data (Tiedemann et al., 2014). I believe though that with respect to the resources involved, this is no different, and that it is only one specific way to leverage parallel data: access to machine translation indeed presupposes the existence of parallel data at some point, at least through a pivot language.[2]

Structured bilingual information In many languages, sense databases like WordNet (Miller, 1995) include correspondence with other languages and can provide a semantic and lexical binding. It is also possible to exploit cross-lingual page links in Wiktionary, and even Wikipedia titles (Tyers and Pienaar, 2008). Bilingual lexicons from any source can also be successfully leveraged. The PanLex database (Kamholz

[1]Still, it can be argued that word and phrase correspondences are easier to detect, and thus to leverage, when the languages are similar enough, because cognates can bootstrap alignment (Kondrak et al., 2003), and monotonic word orderings make the simplest alignment models applicable, among others.

[2]Rule-based machine translation systems are indeed ruled out by the low-resourced language scenario, in which linguists are not available to annotate target data.

et al., 2014) gathers a large number of such resources, with over a billion lexical translations.[3]

Linguistic relatedness Languages from the same family often present strong syntactic, morphological or lexical similarities. While not always overt (like cognates are) or thoroughly described by linguists, these properties can at least be queried implicitly. Indeed, bilingual binding arises also from the design of the processing algorithm, built on the assumption that some features will fire on both languages: it is not necessary to know precisely which features are concerned, the guarantee that some such features exist is sufficient to bootstrap a target model.

Relatedness can however be thought in a broader sense than pure genealogy: this concerns indeed any language that has been influential on the target language, across history. As such, even distantly related languages can help processing some part of target data: see the high number of anglicisms in Japanese. Similarly, Romanian is a Romance language but bears the legacy of various Slavic influences, in the vocabulary but also in morphology and syntax. Notably, Tsvetkov (2015) proposes lexical borrowings as a cross-lingual bridge, based on the evidence that 40% of Swahili's vocabulary is borrowed from Arabic. Yet, such borrowings also affect other levels, like word order (Harris and Campbell, 1995, chap. 6), and Georgi et al. (2010) assess experimentally that some linguistic similarities (e.g. morphology) do not correlate as well as others (e.g. lexical typology) with the traditional phylogenetic language groups.

Linguistic and extra-linguistic universals In a broad sense, this category concerns anything that can be learned in a way that is fully independent of the language: either basic linguistic mechanisms ('*determiners depend on nouns*'), or common knowledge on the semantic and pragmatic grounds ('*Ferdinand de Saussure is a linguist*').

The inventory and analysis of such universals have been the focus of a large body of work in linguistics, over the years (Greenberg, 1963; Hawkins, 1983; Croft, 2002). Yet, Evans and Levinson (2009) regard this theory with sharp criticism: for all universals they consider, they come up with counter-examples. Still, this open debate does not affect cross-lingual transfer, which differs from linguistic theories in the fact that the envisioned properties do not need to be actually universal. They just need to be broad enough to cover the whole panel of languages whose processing is targeted, which in fact cannot be the 7,000 worldwide languages at short term. For instance, many NLP tasks are based on written text processing; cross-lingual methods for those tasks would not be applicable to unwritten languages, in which case

[3]One way to build bilingual lexicons is to induce them automatically from parallel corpora, so that both kinds of resources do not always have a clear boundary. They remain fundamentally different though: parallel corpora contain more information than just word pairs (typically regarding word order), while high-quality lexicons can also be acquired by other means (like crowd-sourcing).

there is no point in restraining them to language universals also shared by unwritten languages.

Besides, the purpose of cross-lingual transfer is not to build a language-independent system,[4] but rather to gather as much useful information as can be captured in any other language. In that sense, the field matches even Evans and Levinson (2009)'s point of view, whereby language diversity is merely based on clusters representing broad linguistic properties. For a given language, the more such clusters can be leveraged, the better: it will always be useful, and in any case better than relying on chance only.

Linguistic typology Useful knowledge is also to be searched in typological features, formalized by linguists as in Haspelmath et al. (2005)'s World Atlas of Language Structure (WALS), or by automatic processing (Lewis and Xia, 2008; Östling, 2015; Coke et al., 2016; Wang and Eisner, 2017). They bring to light important similarities, as well as divergences.

Typological similarities can be fortuitous, independent of language relatedness, and yet relevant. For instance, the Pirahã isolate language follows an SOV word order (Everett, 1986), as Japanese does (Shibatani, 1990). Hence, Japanese knowledge may be used to bootstrap a Pirahã parsing system, even though they are obviously in no way related.

In my opinion, divergences among languages should not be neglected either, even more when they can be characterized, or better, encoded in a task-relevant way. For instance, if the source and target languages are known to have opposite typological properties, it seems reasonable to assume that the target prediction should be anything but what the source system would have predicted; in some cases this information may be sufficient to achieve target-side learning.

Target data as a support Last but not least, another type of linguistic knowledge can be used in low-resourced language processing, and to my mind should not be neglected by studies on cross-lingual transfer: the knowledge provided by sparse target data. Indeed, even if it is noisy or incomplete, it may successfully complement cross-lingual knowledge, by acting as a canvas.

In this spirit, any sample of annotated data is worth leveraging. This is also the case for priors on the language, whose knowledge may result from computation (e.g. by processing raw data) or from various sources like linguistic studies. A brief access to a native speaker is one more way to collect information on the target language, which, when properly targeted, can provide more information than lengthy efforts to acquire more traditional resources (Garrette and Baldridge, 2013). As exemplified

[4]We will see however in Section 2.3 that the recent years have seen a reemergence of this idea.

by Druck et al. (2009), specific (yet sparse) knowledge on the classification features can also be more useful than actual corpus annotations.

Querying the Web, for instance Twitter, is often not an option for very low-resourced languages, which generally lack online presence. Yet, it is worth supporting this option, as such resources are highly valuable. Crowd-sourcing is another way to get sparse target data: for instance, more than a hundred of Wiktionary websites (one per language) contain at least 5,000 entries, and while their morphosyntactic content varies in precision, this resource still covers a wide array of otherwise low-resourced languages.

Such monolingual data can be used to improve the cross-lingual input in many ways: filtering, reweighting, disambiguation, etc. For example, the knowledge that the target language has no determiner could be used to improve word alignment of parallel data, by preventing right away the source determiners from aligning with any target word. Depending on the size and quality of target data, the reverse view may dominate: we can also consider the cross-lingual knowledge as a way to improve noisy target data.

As the title of this section claims, this list intends to be exhaustive; it is probably not. It is indeed likely that there remain many other types of hidden resources, which are not considered yet for cross-lingual transfer, but should be.

This work is not focused on one specific type of resources or properties, but rather aims at leveraging all of them. Chapter 5 studies in which way target data can complement the use of all other types, Chapter 6 characterizes its contribution to complement relatedness and universals, Chapter 7 proposes an extension of a method based on parallel data and Chapter 8 shows how linguistic typology can considerably improve transfer when only universals are available.

2.2 State-of-the-art methods for cross-lingual transfer

This section reviews the state-of-the-art of cross-lingual transfer. While relatively young, the field has benefited from a growing interest in the recent years, and this intense research has produced a rich and wide array of methods. But this diversity has the downside that the proposed methods adopt often very different points of view, drawing from different intuitions, pursuing different objectives and making different assumptions; in many cases it is not even sure whether the proposal actually belongs to the field of cross-lingual transfer, or for instance to domain adaptation.

In order to clarify what I consider to be cross-lingual transfer, and the main ways to achieve it, I thus start this review by a high-level description of transfer approaches, which I group into two categories: data space and parameter space methods. I then describe concrete methods of both types, and additionally present a series of methods, namely direct transfer, whose categorization is unclear in the vanilla version and depends on the enhancements actually performed. I conclude with a few general remarks regarding the need for common representations.

2.2.1 Meta-methods in data and parameter spaces: a reading grid

When confronted with a lack of annotated data, two strategies present themselves: either create such annotated data, or dispense with it and find an alternate way to build the model. In both cases, cross-lingual transfer repositions the setting in standard well-resourced conditions (where either a dataset or a model is available in the target language), so that all traditional methods to build or improve a system can then be envisioned: supervised, unsupervised or semi-supervised learning, self-training, ensembling methods, etc.

In the first strategy, synthetic data in the target language is produced thanks to source data or knowledge, which enables target-side training as usual; I name this *data space transfer*. Conversely, *parameter space transfer* consists in borrowing a trained model in the source language, in order to avoid training a model on target data: source model parameters are directly reused for target processing, sometimes with some refinements. We will see later that the frontier between these two kinds of transfer is sometimes blurry, because some interesting ideas have been applied concurrently to both transfer approaches; in such cases I look more closely at the intended use of transferred knowledge. A typical controversial case is when transfer is followed by semi-supervised learning, with source parameters acting as a guide. I still categorize this as parameter space transfer, since the data used in the end for training is purely target-side, and not the outcome of data transfer; target-side learning is only a refinement of the core method. Figure 2.1 illustrates both strategies and their differences.

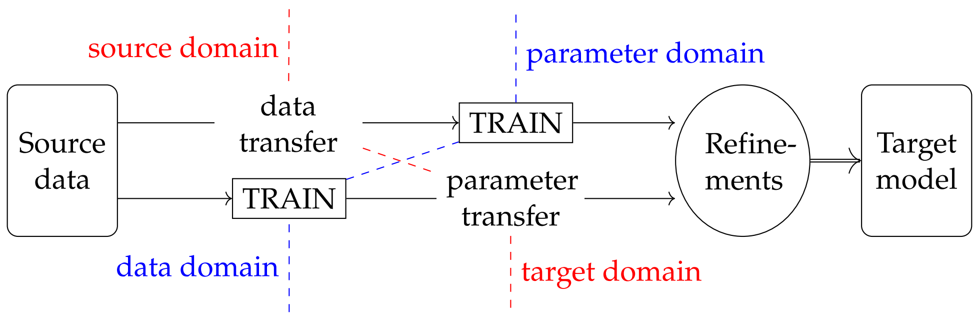


FIGURE 2.1: Cross-lingual transfer in data and parameter spaces: outline schematic. Transfer aims at exploiting source data to build a target model, which requires two domain shifts, from data to parameters (training) and from source to target (transfer). The main difference between both approaches is in the order of these shifts.

On the genericity of transfer methods Historically, cross-lingual transfer has been mostly studied on a few emblematic tasks: some sequence labeling tasks (PoS tagging, noun phrase chunking, named entity recognition, parsing, semantic role labeling), sentiment analysis and text classification. However, its motivation holds for all NLP tasks, and as such should be applicable to all of them. One way to factorize the research efforts on all those fronts is genericity; I propose to pursue this objective. This means letting transfer abstract out from the task at hand, and that the proposed algorithms should only wrap existing systems, considered as black boxes.

We can make two factual statements about data-driven systems: they all need classifiers, and they all need data. Both can consequently act as an abstract interface to achieve task agnosticism, and the transfer wrapper can seamlessly plug in both positions, if the chosen algorithms are appropriate though.

On the classifier side, designing generic transfer methods can be achieved if the machine learning framework is task-agnostic and the task-specific components of the system are themselves wrappers to the core classifier – for instance an additional final layer in a neural network. There has been a recent surge of interest for this approach, in particular with the multi-task learning framework (Caruana, 1997). I have not explored this path in my work though, rather focusing on the data side, which offered more opportunities to link my analyses with linguistic intuitions.

Data-side genericity relies on standardized formats of the data, for instance in a tabular form for word-level annotations; the transfer approach must then be compatible with arbitrary tabular annotations, remaining agnostic to their actual content. But as long as the produced target data ends up in the same standardized format, the system would not know the difference from standard data and it can learn information of any kind. This is the reason why most of my work operates at the data level; and the core of my contributions in Chapters 8 and 9 is actually data wrappers, which rewrite or filter training corpora.

2.2.2 Data space transfer

Annotation projection The seminal work of Yarowsky and Ngai (2001) establishes the concept of annotation projection through parallel data. While their primary goal is to build data for PoS tagging and noun phrase chunking, the method is extended to other tasks by Yarowsky et al. (2001). As illustrated in Figures 2.2 and 2.3a, the idea is to perform automatic word-level alignment of parallel sentence pairs, annotate the source side with a source model, and assume that aligned word pairs share the same word-level annotation. The newly annotated target side of the parallel data now constitutes the desired target data.

Annotation projection is a rather generic approach, as it can project any kind of label and adapt to alignment links of various granularity, yet it is limited by

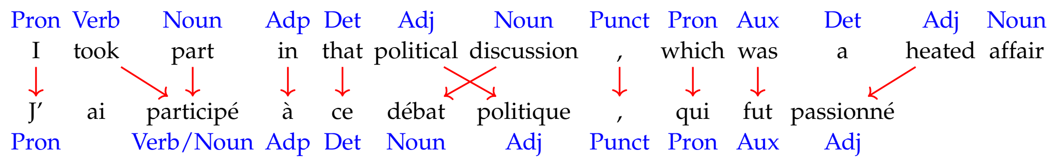

Pron Verb Noun Adp Det Adj Noun Punct Pron Aux Det Adj Noun
I took part in that political discussion , which was a heated affair
J' ai participé à ce débat politique , qui fut passionné
Pron Verb/Noun Adp Det Noun Adj Punct Pron Aux Adj

FIGURE 2.2: Projection of PoS tag annotations from English to French. The issues raised by the unaligned word *ai* and the multi-aligned word *participé* are traditionally handled by task-specific heuristics.

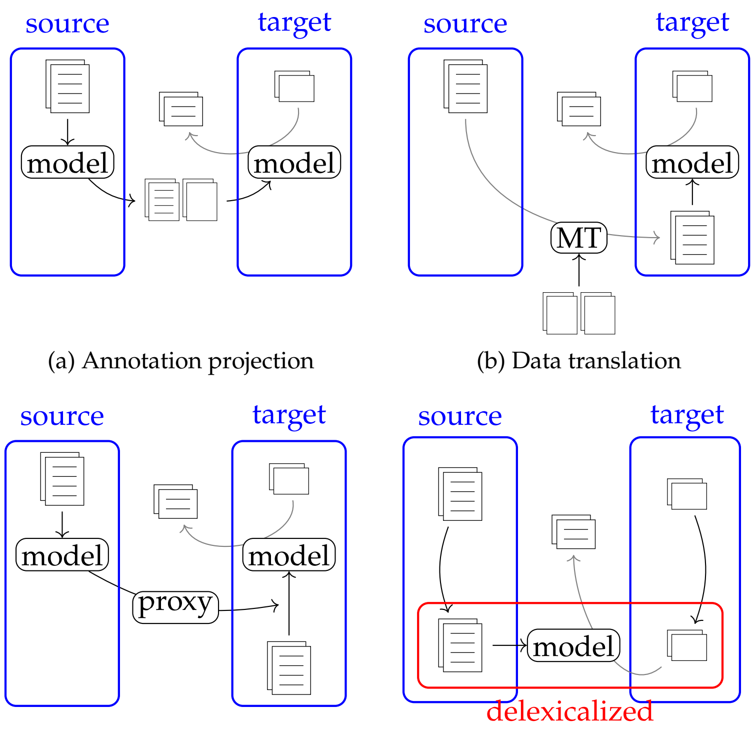
the poor 1-to-1 correspondence of word sequences in many language pairs (Ma et al. (2007) measure 60-70% of 1-to-1 links between English and Chinese, Italian or Dutch). Indeed, Yarowsky and Ngai (2001) solve the multiple alignment issue with task-specific heuristics, assuming for instance precedence relations between function words and open PoS classes. Besides, not all tasks can be easily formulated as a label over a sentence part, for instance dependency parsing whose edges involve two distinct words.

Many refinements to annotation projection have been proposed since then. A variant with multiple source languages is suggested by Fossum and Abney (2005), while Agić et al. (2015) take a step further with multi-parallel data and a multiparty vote on the projected annotation.

To the best of my knowledge, Xi and Hwa (2005) are the first to suggest complementarity between projected and manually annotated data. They build a target model with minimal knowledge overlap with the projected data, and they use the projected model as a backoff when monolingual data is not confident enough. Täckström et al. (2013) also improve tagger projection with target knowledge, by including tag constraints extracted from Wiktionary; Wisniewski et al. (2014) pursue on this track by adapting the learning framework to better exploit the crowd-sourced constraints.

Finally, several works propose to relax the single-label constraint on projection, mostly to address alignment issues (Wróblewska and Przepiórkowski, 2014). Similarly, Wang and Manning (2014) project the full *distribution* of posterior probabilities (that is, one score per possible label), which is more informative, on top of being straightforward to combine over multiple aligned words, by averaging.

Data translation An alternative way to build annotated data is by translating source data into the target language (see Figure 2.3b). This strategy has been experimented with for classification tasks at document level (e.g. text categorization: Rigutini et al., 2005), sentence level (e.g. subjectivity analysis: Banea et al., 2008) and word level (typically, parsing). In the word-level case, the proposed methods preserve the word sequence in various extents, ranging from glossing (Zhao et al., 2009) and word-level marginalization (Durrett et al., 2012) to the use of a full machine translation system (Tiedemann et al., 2014), in which case 1-to-1 correspondence is no more guaranteed than with parallel sentences.



(a) Annotation projection (b) Data translation

(c) Training guidance (d) Direct delexicalized transfer

FIGURE 2.3: Outline schematic of several methods for cross-lingual transfer. Training and test data are represented by large and small documents respectively; horizontal lines on documents denote annotations.

Sources of noise These methods are all likely to generate noisy labels, but for various reasons. Annotation projection is subject to annotation noise, alignment noise and poor bilingual correspondence, because of translational divergence or non-isomorphism between the languages. It is also prone to domain effects, if the selected parallel data does not match the domain on which the source model was trained.

Glossing methods avoid all these sources of noise, as they are based on manually annotated data, but they depend on the quality of the bilingual lexicon, and even so they often result in target-side fluency mismatch: the synthetic data follows a wrong word order, and the system will not be trained to perform well on natural target sentences.

As for machine translation techniques, they suffer from translation noise, which is even worse when domains do not match, but also from the poor correspondence issue. However, outputs of machine translation systems tend to have a word order that is closer to the source-side one, in which case it is a trade-off between projection and glossing; besides, alignment links come out as a by-product of many MT systems. Duh et al. (2011) point out other hidden effects of domain adaptation when using translated corpora. Yet empirically, methods based on machine translation have proved successful in many cases.

2.2.3 Parameter space transfer

Parameter transfer is based on the reuse and modification of model parameters in other languages; focus is on the model. For instance, Cohen et al. (2011) build the target model as a mixture (either uniform or empirically estimated) of several models in related languages, retaining only the unlexicalized parameters (that is, not involving any wordform). Conversely, Kozhevnikov and Titov (2014) are able to transfer from a single language, since they take into account linguistic differences like morphological richness, through feature mapping. Indeed, what is expressed as a suffix in the target language may well correspond to an independent token in the source. They first learn compact vector representations in both languages, then use parallel data to estimate a linear mapping. When processing the target language, this mapping can then be used as a proxy to the (unchanged) source model.

Regarding semi-supervised models, the most straightforward guidance is to use the transferred knowledge as soft constraints, i.e. parameter regularization (Ganchev et al., 2009; Duong et al., 2015b; Martins, 2015). This still requires selection or mapping mechanisms to bind the parameters in both languages, as shown in Figure 2.3c. In a similar fashion, joint learning of models in languages with different amounts of data is another way to improve low-resource training with knowledge from another language. I consequently also interpret it as transfer, in a broader sense. This strategy has been explored in various tasks, like sentiment analysis (Wan, 2009; Lu et al., 2011), named entity recognition (Burkett et al., 2010; Wang et al., 2013) and constituency parsing (Burkett et al., 2010).

2.2.4 An ambivalent case: direct delexicalized transfer

Zeman and Resnik (2008) introduce the simplest, and most emblematic, of all transfer methods: direct delexicalized transfer. As shown in Figure 2.3d, it consists in training the source model on a delexicalized version of the source data, and using it on a delexicalized version of the test data to produce the annotations.[5]

For parsers, delexicalization means using only PoS-based features, but it is possible to go further and use even looser representations of words, typically to build delexicalized PoS taggers as Yu et al. (2016) did. They represent a word using for instance its length and its entropy, estimated on raw monolingual data. This is one

[5]When languages are so close that they actually share wordforms, direct *lexicalized* transfer could be considered. However, the literature of cross-lingual transfer does not address this case in general, which is rather associated with domain adaptation or dialectal studies. A first step in that direction is proposed in Appendix C.

more example of how target data can complement the cross-lingual resources. However, Zeman et al. (2016) describe such enhanced delexicalization as 'planting trees in the desert': the looser representations are, the thinner the cross-lingual binding becomes, to the point that complex models like dependency parsers fail to transfer.

Since basic direct transfer does not involve actual transformations, it can be interpreted in two ways, either as data transfer (delexicalized sentences are considered as an approximate form of target data) or as parameter transfer (the source-side model is directly reused, without alteration). Both interpretations correspond to different paths of research: data-side refinements aim at making the sentences more target-like, while as parameter transfer it involves making the model more usable on target input.

In the case of glossing techniques for instance, glossing training data (Zhao et al., 2009) is a means to transfer data, while glossing test data (Zeman and Resnik, 2008) uses the source model as is, and thus belongs to the parameter space category. Other data space improvements of direct transfer often revolve around instance weighting (Søgaard and Wulff, 2012). Refinements in the parameter space can involve using the model to parse raw data and then perform self-training (McDonald et al., 2011), or computing mixtures of several models (Søgaard and Wulff, 2012).

2.2.5 Common representations

The methods or refinements based only on linguistic similarities rely explicitly on the use of common representations across languages. This is an important line of research that overcomes the field of cross-lingual transfer. The purpose is to attribute similar representations to translation word pairs and equivalent morphosyntactic properties.

While morphosyntactic specifications were traditionally specific to each language, the Multext-East project (Dimitrova et al., 1998; Erjavec, 2010) designs a unified specification system that covers several morphologically rich languages of Eastern Europe. In the same perspective, Petrov et al. (2012) propose a tagset of 12 UPOS (universal PoS) whose coarse tags are cross-linguistically consistent (e.g. nouns, adverbs, adpositions); the UD project (Nivre et al., 2016) refines this set with an extension to 17 UPOS and a collection of universal features for finer-grained tagging.

One alternative to using PoS and morphological tags is automatic word clustering, which accounts for both morphosyntactic and semantic similarities. Täckström et al. (2012) design cross-lingual word clusters, by projecting Brown-like clusters (Brown et al., 1992) through parallel data and enforcing cluster agreement up to convergence.

While often very valuable, the design of such common representations still relies on a strong hypothesis, the fact that word classes and concepts can be cross-linguistically consistent on a wide array of languages: in other terms, it assumes the validity of language universals.

Besides, common representations often hide the fact that the very same input (typically, an identical PoS sequence) may call for different system behaviours in different languages; conversely, they may expect the same behaviour for wholly different inputs. For instance, in parsing, languages with preposed and postposed determiners (e.g. French and Arabic) will parse the 'NOUN DET NOUN' phrase differently, and at the same time they both solicit the universal knowledge that determiners depend on nouns, but one will only apply it on 'DET NOUN' inputs and the other only on 'NOUN DET' ones. I investigate and address such issues in Chapters 6 and 8.

2.3 Recent advances: cross-lingual embeddings and multilingual models

With the rise of better classifiers based on neural networks, an increasing interest has emerged in the recent years for transfer in the parameter space.

To this end, the concept of cross-lingual word clusters has been translated into its continuous counterpart, cross-lingual word embeddings (Klementiev et al., 2012), which better fits into the neural framework. Here again, the idea is that syntactically or semantically similar words get similar embeddings, both monolingually and across two languages.

After Klementiev et al. (2012), who rely on parallel data with word alignments, many have tried to reduce the need for bilingual supervision. Some replace word-level alignments with sentence-level alignments (Hermann and Blunsom, 2013; Gouws et al., 2015), which are easier to get. Others rather turn to bilingual lexicons (Mikolov et al., 2013; Xiao and Guo, 2014), whose size requirements have been gradually lowered through years, down to 10 word pairs (Zhang et al., 2016), yet in this case the resulting embeddings are only usable for coarse labeling tasks (Artetxe et al., 2017). Ruder (2017) conducts an extensive survey of cross-lingual embedding models, to which I refer the reader for further information.

Despite the diversity of these models, Upadhyay et al. (2016) note that they all rely on alignments of some sort, either at document level or sentence level, sometimes with additional word-level alignment, or on bilingual lexicons. They also assess empirically that depending on the transfer task, training such cross-lingual embeddings requires bilingual data of various granularity; for instance dependency transfer performs best with word-level alignments. In the same vein, Levy et al.

(2017) show that cross-lingual embeddings and word alignments have more in common that thought before, both theoretically and empirically. To my mind, these studies draw an interesting parallel between model transfer based on cross-lingual embeddings, and annotation projection. It appears that both exploit in fact the same cross-lingual input, but while the outcome of annotation projection is task-specific (data with a given type of annotations), methods based on cross-lingual embeddings single out the bilingual binding and provide the common representations as a by-product, which is valuable when reusability is a concern.

While cross-lingual word embeddings were primarily intended for direct transfer (Klementiev et al., 2012; Xiao and Guo, 2014; Guo et al., 2015), their ability to make source and target data indistinguishable in the input space, but without the information loss due to delexicalization, has found other uses, namely multilingual models.

The recent literature indeed exploits the multi-task learning framework, with a deep neural flavor (Collobert and Weston, 2008; Luong et al., 2016), to build a single model covering multiple languages, similarly to what was already done in the speech recognition field (Huang et al., 2013; Ghoshal et al., 2013). Whether they are called multilingual, language-independent or universal models, I categorize such models as a special case of joint learning, as they are fundamentally based on parameter sharing.

Research on multilingual neural models (Dong et al., 2015; Duong et al., 2015a; Ammar et al., 2016a; Susanto and Lu, 2017; Pappas and Popescu-Belis, 2017) has reported for various tasks that most covered languages benefit from this knowledge sharing, but above all the lesser-resourced ones. In this regard, multilinguality enables cross-lingual transfer.[6]

For all, a seamless sharing of upper layers is in fact enabled by the shared representations that cross-lingual embeddings provide: as soon as the input words are projected to the embedding space, sentences in various languages are supposedly indistinguishable. In order to increase the coverage of multilingual systems, embeddings models have also been extended to specifically support cross-lingual consistence across an arbitrary number of languages (Hermann and Blunsom, 2013; Huang et al., 2015; Guo et al., 2016; Ammar et al., 2016b).

[6]Multilinguality to improve on low-resource scenarios has also been explored in machine translation (Firat et al., 2016a,b; Johnson et al., 2017; Ha et al., 2016), but in that case it rather aims at learning a neural Interlingua (Schwenk and Douze, 2017), in an encoder-decoder framework. Such multilinguality is achieved by concatenating parallel corpora in a way that tightly binds internal representations of all covered languages. This method enables zero-shot learning, in the sense that it covers zero-resource language pairs, but not zero-resource languages: both languages still need a significant amount of parallel data with a bridge language. While the longing for multilinguality is similar to the other models mentioned here, the approach and the data requirements differ; as a result, to my mind, it does not fit into the framework of cross-lingual transfer for low-resourced languages.

2.4 Evaluation of cross-lingual transfer

The task fulfilled by transferred systems is no different from that of the corresponding supervised models, transfer is just another way to achieve the same goal. Evaluating the accuracy of a given cross-lingual model is consequently rarely an issue, because it can easily reuse the metrics and process used in monolingual cases. But evaluation of cross-lingual transfer in general still faces a major issue, which is the choice of a fair and realistic experimental setup.

Indeed, transfer methods are meant for low-resource scenarios, i.e. for languages for which not enough annotated data is available; and yet, for most NLP tasks, evaluation supposes *precisely* access to some data, namely the test set, with gold-standard manual annotations. So, in order to avoid unrealistic scenarios, simulated using well-resourced languages (which is a biased set, first of all towards European languages), the only way is to choose a truly low-resourced language and create some annotated data specifically for this evaluation. But in that case, the language stops being low-resourced. Chapter 5 will show indeed how blurry the frontier between low-resourced and well-resourced languages can be: datasets as small as 20 sentences can be successfully considered as training sets, while reliable test sets contain typically much more than 20 sentences.

This seemingly unsolvable issue unveils another specificity of the field: the true purpose of cross-lingual transfer is to build and use NLP systems *whose accuracy is completely unknown*, or otherwise the test data would rather be used as training data. Traditional systems always come bundled with experimental scores, which are not an actual guarantee, but at least an estimation of real-world performance; for transferred systems it is only extrapolated from transfer successes *in other languages*, and if for the language at hand the linguistic similarities have been misjudged or universals over-estimated, the system may be completely wrong, without any way to tell. This raises the question whether users outside of the academic world, like companies or government agencies, are ready to overcome that risk, or still prefer to support costly annotation campaigns – or simply give up on that language.

Conversely, whatever the chosen setup, it runs the risk of under-estimating the accuracy of cross-lingual systems. Indeed, using resources in one language and testing in another often means that two separate teams have annotated the data, in which case there is no guarantee that they follow exactly the same annotation scheme. So, even a very accurate cross-lingual model can be granted with a low score, just because its predictions encode the same information as the test data but in a different way, and as such are considered incorrect. Regarding PoS tags, the UPOS tagset (Petrov et al., 2012) alleviates a lot the discrepancies: *adjective*, *A*, *JJ* and all other tags denoting adjectives in various languages are gathered under the same ADJ tag. But issues remain. For instance, many Japanese adjectives conjugate; in this regard a cross-lingual PoS tagger may annotate them as VERBs, which is true from the viewpoint of many languages, while test data would consider them as ADJs, and flag an error. More will be said in Chapter 4 on the recent community efforts for cross-language consistency, in the specific case of parsing.

As of now, no definite solution has been found for the numerous challenges faced by the evaluation of cross-lingual methods. But there are many ongoing efforts to alleviate some of its issues.

2.5 Conclusion: a flourishing field, yet hard to categorize

This chapter has presented the field of cross-lingual transfer, both conceptually and in the most recent literature.

It has discussed the various kinds of cross-lingual inputs that can allow the source and target languages to interact. These range from bilingual data and linguistic similarities to a series of often overlooked sources of information, like typological databases and small pieces of target knowledge.

I have also proposed a reading grid of transfer approaches, which distinguishes the methods that operate cross-lingual transfer in data and parameter spaces. In several cases (semi-supervised and joint learning, neural Interlingua, etc.) applying this typology is less straightforward, which I discuss to some extent. I have then presented various algorithms and extensions that have been developed for cross-lingual tasks, including the last advances in parameter transfer, based on shared representations with cross-lingual embeddings, and multilingual neural models.

Finally, I have raised the topic of evaluation of cross-lingual transfer, and shown that it faces several important challenges, which often hinder fair and realistic evaluation.

In this work, I address with further interest two aspects of the cross-lingual literature: refinements to data transfer (Chapter 8) and the diversity of usable resources, with the purpose of softening the specific data requirements of many state-of-the-art techniques (Chapter 9). Another concern is an empirical and qualitative evaluation (Chapters 5 and 6) of what has been achieved in cross-lingual transfer so far: is it already a good alternative to data acquisition? If not, what remains to be done to close the gap?

Chapter 3

Recent advances in transition-based dependency parsing

***

Contents

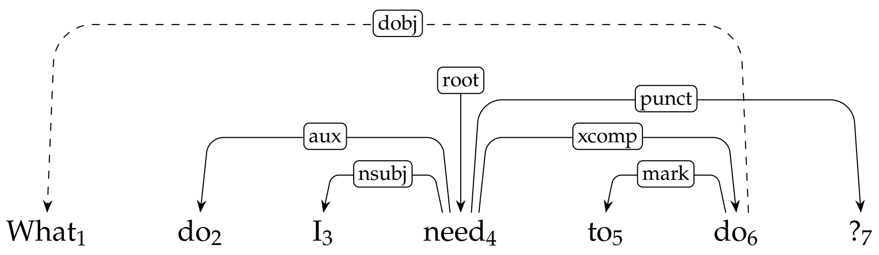
Both in linguistics and in NLP fields, the literature studies syntactic structures through several frameworks: dependency parsing, constituency parsing, combinatory categorial grammars... However, according to the application frame chosen for this thesis, I focus in the following on dependency parsing.

I first introduce dependency parsing and transition-based methods in Sections 3.1 and 3.2, and then describe the main transition systems (Section 3.3), as well as three aspects that have seen important advances in the recent years: the underlying classifier (Section 3.4), beam search (Section 3.5) and dynamic oracles (Section 3.6).

3.1 The dependency parsing formalism

Dependency parsing formalizes the syntactic structure of a sentence as a set of edges from a *head* word to a *child* word. These edges form a tree over the words, i.e. the graph is acyclic, connected, and each word is *attached* to exactly one head, except for the root word that is either left unattached or attached to a dummy ROOT token. Depending on the annotation scheme, a sentence may however contain several roots.

Dependencies are usually decorated with relation labels, which denote the type of dependency (e.g. nominal subject, clausal complement, indirect object...). Thus, the task of dependency parsing consists in predicting, for each word in the sentence, the index of its head word, and a relation label; not all such predictions are valid parses however, because of the tree constraint. Figure 3.1 illustrates this task, for a given dependency tree.



Indices	1	2	3	4	5	6	7
Words	What	do	I	need	to	do	?
Heads	6	4	4	0	6	4	4
Labels	dobj	aux	nsubj	root	mark	xcomp	punct

FIGURE 3.1: Dependency tree, represented in a human-readable form and as a tabular prediction task.

Among the various properties that dependency trees can have, one is particularly important in this context: *projectivity*, which denotes the absence of edge crossings in a tree. A dependency tree is said *non-projective* if it contains at least one crossing, i.e. an edge between words of indices i and k and one between j and l, with i < j < k < l. For instance, in Figure 3.1, the dependency *What* $\curvearrowleft$ *do* makes the tree non-projective. Most languages and annotation schemes present a small number of non-projective trees, around 5-10%.[1]

Nowadays, the accuracy of dependency parsers is mostly evaluated using two metrics, the Unlabeled Attachment Score (UAS) and the Labeled Attachment Score (LAS). UAS is the percentage of words in the test set that are assigned the correct head, while LAS is the percentage of words that are assigned both the correct head and the correct label. Since the number of predictions is fixed, with exactly one prediction per word, this accuracy corresponds to both precision and recall.[2] It is common in the literature, but not systematic, to compute UAS and LAS on all tokens except punctuation, because even though punctuation tokens provide useful information on the structure of the sentence, their own attachment is often purely conventional, and as such hard to predict, while not even representative of the syntax of the sentence.

3.2 Transition-based parsing: strengths and limitations

State-of-the-art dependency parsers can be classified in two major approaches: graph-based and transition-based parsing.

Graph-based parsing consists in scoring all possible dependencies between two words, and searching a subset of high-scored dependencies through the resulting graph, thus extracting a high-scored tree. When the score of each dependency can be computed separately, with no reliance on the neighbouring dependencies, the model is called *arc-factored*, and parsing a sentence sums up to the graph-theoretic task of finding a Maximum Spanning Tree (MST). The main implementation in this line of work is the MSTParser (McDonald et al., 2005), whose parsing time is quadratic in the length of the sentence.

The kind of decisions that are required by such algorithms is said to be *global*: each dependency tree is assigned a score, which is the only one to be maximized. This way, edges are not selected individually, but as a whole, taking sentence-level interactions into account: one high-scored dependency may be rejected if (because of the tree constraint) it prevents many other dependencies with relatively good scores.

[1]They represent 4.96% of trees (0.48% of dependencies) for English, 12.45% (0.83%) for French, 8.19% (0.33%) for Arabic. However, the numbers rise for specific languages like Dutch (30.87%, 4.10%) or Ancient Greek (63.22%, 9.78%). See (Straka et al., 2015) for complete data on 37 treebanks.

[2]Still, those notions can diverge in particular settings, for instance in raw text parsing, where the correct word tokenization is unknown.

To the contrary, in transition-based parsing, whose most emblematic instance is the MaltParser (Nivre et al., 2006), the dependency tree is not scored as a whole but built incrementally. The parser has an internal state (its *configuration*), representing a partial analysis of the sentence, and the algorithm applies a series of *transitions* (also called *actions*) on this state, typically consuming a word or adding an edge to the dependency tree. When the parser reaches a final state, the sentence is fully parsed; the corresponding sequence of transitions is called the *derivation*.

In this case, the prediction task is about the next transition to apply, and parsing is done purely through *local decisions*: the underlying classifier has only access to a limited view of the sentence, and selects edges one by one.

Because words are read and processed sequentially, the number of transitions needed to parse a sentence is linear in the length of the sentence. As a result, transition-based parsing is computationally more efficient than graph-based parsing, which has caused a growing interest for this approach. But the price of efficiency is paid by restrictions on expressivity: while graph-based parsers are able to produce any tree over the input sentence, the standard algorithms for transition-based parsing are restricted to projective trees, because the linear flow of transitions forbids to add some dependencies. This is a significant issue for accuracy, since at test time, a projective parser is guaranteed to be wrong for a number of edges, namely the non-projective dependencies.

Several strategies have been proposed to overcome the projectivity restrictions. One line of research is to authorize non-projectivity in transition-based parsers, by designing alternate transition systems that examine more word pairs. Such systems are less efficient though, with quadratic (Covington, 2001) or at worst quadratic (like the SwapStandard system of Nivre (2009)) parsing time, which reduces the benefits of transition-based parsing. Others rather introduce non-projective dependencies by post-processing the output projective trees. This is the case of the popular pseudo-projectivization method (Nivre and Nilsson, 2005) that encodes crossings in augmented relation labels. Finally, another way of abstracting from the projectivity constraint while maintaining a linear parsing time is reparsing (Sagae and Lavie, 2006) which consists, in the frame of ensembling methods, in applying MST techniques on a set of candidate trees.

Reparsing can also be considered as a method for combining the strengths of graph-based and transition-based parsing, which others have attempted at the feature level (Zhang and Clark, 2008; Nivre and McDonald, 2008).

According to the chosen case study, I focus here on transition-based parsing. One notable upside of this choice is that the state of the art of this approach presents more diverse algorithms than graph-based parsing, hence it allows a broader creativity in the frame of this thesis. Still, I rely sometimes on the MST algorithm throughout the study, but only for reparsing purposes.

3.3 Transition systems

The various transition systems that have been proposed over the years break down into three categories. *Stack-based* systems rely on a stack of partially processed words and a buffer of incoming words, and process the sentence from left to right in a shift-reduce manner. *List-based* systems traverse the sentence once per candidate head word and their internal state consists of two lists and a buffer. Such algorithms are less efficient, with a quadratic parsing time (Covington, 2001). The last category is that of *non-directional* parsing, typically the easy-first system (Goldberg and Elhadad, 2010), which consists in choosing at each step both a position in the sentence and a transition to apply. It is an intermediate to graph-based parsing, but remains efficient with an appropriate implementation.

3.3.1 Stack-based systems

Thanks to their efficiency and their implementation simplicity, the most popular systems are the stack-based ones. In the following, I describe the main stack-based transition systems: ArcStandard, ArcEager, ArcHybrid and SwapStandard. None of them is better by design than the others, but depending on properties of the classifier (robustness to spurious ambiguity, feature extraction adapted to bottom-up parsing, etc.) and of the processed language (e.g. non-projectivity), one system may be more adapted than another; all four have been used in the recent literature. I present them using the following notations: σ stands for the stack, s and s' for the first (topmost) and second stack elements, β stands for the buffer, b for the first buffer element, and P is the partially built tree. The description of the systems also includes preconditions of each transition: in that context, the term 'word' refers to a token which is not a padding token. Figure 3.2 illustrates an ArcEager derivation.

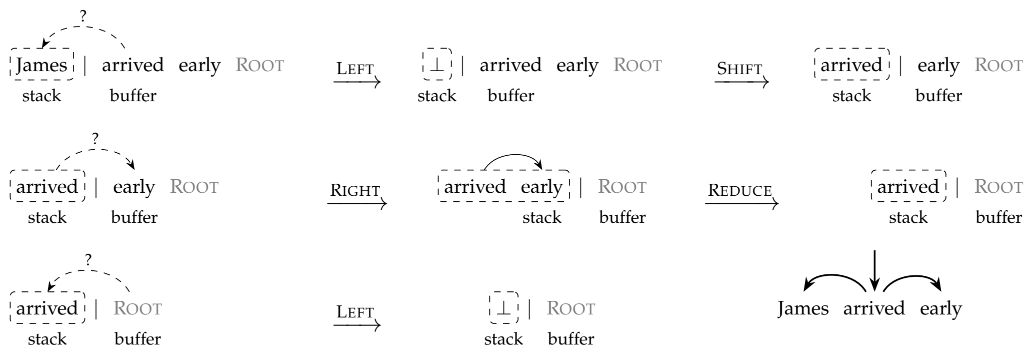


FIGURE 3.2: A complete ArcEager derivation and the resulting tree.

ArcStandard The ArcStandard transition system (Nivre, 2003) is based on 3 actions, described in Table 3.1. It is restricted to projective parsing and parses the sentence in a bottom-up manner, i.e. each word receives first its children, then its head. The main downside of this system is a particularly high rate of spurious ambiguity: depending on the order in which left and right children are assigned to their head, many different sequences of transitions may lead to the same parse tree.

SHIFT	$(\sigma,$	$b\|\beta,$	$P)$	$\Rightarrow$	$(\sigma\|b,$	$\beta,$	$P)$	
LEFT	$(\sigma\|s'\|s,$	$\beta,$	$P)$	$\Rightarrow$	$(\sigma\|s,$	$\beta,$	$P + (s \rightarrow s'))$	if s' is a word
RIGHT	$(\sigma\|s'\|s,$	$\beta,$	$P)$	$\Rightarrow$	$(\sigma\|s',$	$\beta,$	$P + (s' \rightarrow s))$	

TABLE 3.1: The ARCSTANDARD transition system.

ArcEager The ARCEAGER transition system (Nivre, 2004) is based on 4 actions, described in Table 3.2. It is also restricted to projective parsing. The system is said *eager* because words receive their head as soon as possible, i.e. when both the head and the child have been seen: a given word receives its head either just after its left children (if the head is on the left), in which case it is kept on the stack for a little longer, or (if the head is on the right) when it has received all its children, in which case it is removed from the stack immediately. Compared to ARCSTANDARD, ARCEAGER has less spurious ambiguity, but it introduces an asymmetry in the classification task, depending on the direction of the edge: some decisions must be taken without knowledge of the head word, while others have already access to it.

SHIFT	$(\sigma,$	$b\|\beta,$	$P)$	$\Rightarrow$	$(\sigma\|b,$	$\beta,$	$P)$	if b is a word
LEFT	$(\sigma\|s,$	$b\|\beta,$	$P)$	$\Rightarrow$	$(\sigma,$	$b\|\beta,$	$P + (b \rightarrow s))$	if s is a word and s is unattached
RIGHT	$(\sigma\|s,$	$b\|\beta,$	$P)$	$\Rightarrow$	$(\sigma\|s\|b,$	$\beta,$	$P + (s \rightarrow b))$	
REDUCE	$(\sigma\|s,$	$\beta,$	$P)$	$\Rightarrow$	$(\sigma,$	$\beta,$	$P)$	if s is attached

TABLE 3.2: The ARCEAGER transition system.

ArcHybrid The ARCHYBRID transition system (Kuhlmann et al., 2011) is based on 3 actions, described in Table 3.3. It has been designed to combine ARCSTANDARD's and ARCEAGER's strengths. Also restricted to projective parsing, it parses in a bottom-up manner, has no spurious ambiguity because it enforces attaching the left children before the right ones, and it also inherits some useful properties from ARCEAGER, like arc-decomposition (see §3.6).

SHIFT	$(\sigma,$	$b\|\beta,$	$P)$	$\Rightarrow$	$(\sigma\|b,$	$\beta,$	$P)$	if b is a word
LEFT	$(\sigma\|s,$	$b\|\beta,$	$P)$	$\Rightarrow$	$(\sigma,$	$b\|\beta,$	$P + (b \rightarrow s))$	if s is a word
RIGHT	$(\sigma\|s'\|s,$	$\beta,$	$P)$	$\Rightarrow$	$(\sigma\|s',$	$\beta,$	$P + (s' \rightarrow s))$	

TABLE 3.3: The ARCHYBRID transition system.

SwapStandard As described in Table 3.4, the SWAPSTANDARD transition system (Nivre, 2009) is an extension of ARCSTANDARD, whose 4th action allows the production of any non-projective tree: whenever a non-projective dependency is required, a SWAP action temporarily displaces the corresponding word, to process it in the common manner. In contrary to the previous systems, parsing with SWAPSTANDARD requires a number of transitions that is not guaranteed linear. The parsing time is at best linear and at worst quadratic; still, in average, it is empirically close to linear (Nivre, 2009). Additionally, a later study from Björkelund and Nivre (2015) proposes a way to train for minimizing the number of SWAP actions needed by a given parse.

SHIFT	$(\sigma,$	$b\|\beta,$	$P)$	$\Rightarrow$	$(\sigma\|b,$	$\beta,$	$P)$	
LEFT	$(\sigma\|s'\|s,$	$\beta,$	$P)$	$\Rightarrow$	$(\sigma\|s,$	$\beta,$	$P+(s\to s'))$	if s' is a word
RIGHT	$(\sigma\|s'\|s,$	$\beta,$	$P)$	$\Rightarrow$	$(\sigma\|s',$	$\beta,$	$P+(s'\to s))$	
SWAP	$(\sigma\|s'\|s,$	$\beta,$	$P)$	$\Rightarrow$	$(\sigma\|s,$	$s'\|\beta,$	$P)$	if s' is a word and $s < s'$

TABLE 3.4: The SWAPSTANDARD transition system.

In all these systems, relation labels can be handled in various ways. A straightforward approach is to add the label at the same time as the edge, i.e. the LEFT and RIGHT actions are replaced by LEFT$_l$ and RIGHT$_l$ actions for each label l. The alternative is to keep the transition system unlabeled, and let another component of the parser handle labeling, by adding the label for instance just after the edge, or at the end, by post-processing the whole unlabeled tree.

Variants In addition to the main four stack-based systems, a number of variants have been proposed, adding for instance non-monotonicity (Honnibal et al., 2013; Honnibal and Johnson, 2015), a limited degree of non-projectivity (Fernández-González and Gómez-Rodríguez, 2012; Pitler and McDonald, 2015) or constraints on the output (Nivre and Fernández-González, 2014; Nivre et al., 2014).

Additionally, it can be noted that depending on the conventions for roots (root unicity, use and position of a dummy token, etc.), the algorithms may differ significantly, regarding implementation, accuracy, and system properties. For instance, ARCEAGER performs significantly better with a ROOT token placed at the end rather than at the beginning, while no difference has been noted for ARCSTANDARD, as shown by Ballesteros and Nivre (2013) and reported in Table 3.5.[3]

UAS	ARCEAGER	ARCSTANDARD
No ROOT	84.35	84.41
ROOT in first position	83.67	84.44
ROOT in last position	84.35	84.38

TABLE 3.5: UAS variations obtained by Ballesteros and Nivre (2013) when moving or removing the dummy ROOT token, averaged over the Penn Treebank plus CoNLL-X data.

[3]Similarly, I will show in Chapter 7 that the arc-decomposability property of ARCEAGER holds when the dummy token is at the end but not when it is at the beginning of the sentence.

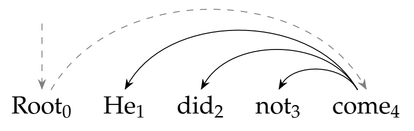
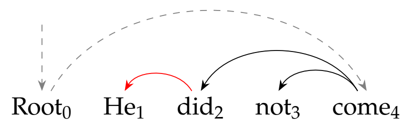
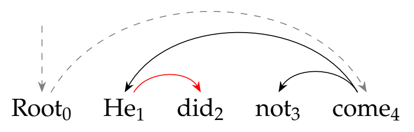
Derivation	Resulting parse
Shift$_1$ **Shift$_2$** Shift$_3$ Left$_{3\leftarrow4}$ Left$_{2\leftarrow4}$ Left$_{1\leftarrow4}$	
Shift$_1$ **Left$_{1\leftarrow2}$** Shift$_2$ Shift$_3$ Left$_{3\leftarrow4}$ Left$_{2\leftarrow4}$	
Shift$_1$ **Right$_{1\rightarrow2}$** Reduce$_2$ Shift$_3$ Left$_{3\leftarrow4}$ Left$_{1\leftarrow4}$	

FIGURE 3.3: Counter-example to direct action-edge correspondence. This uses the ARCEAGER transition system, with a dummy ROOT token in first position. Underlined actions denote actual decisions, while for the others only one action is legal at once.

3.3.2 Understanding derivations: action-edge correspondence

Understanding what a derivation represents requires some clarification: despite action names like LEFT or RIGHT, I claim that in general, the decision to create a given edge cannot be clearly related to one specific action. Figure 3.3 shows an example why.

Supposing that the gold parse is the upper one, comparison with the middle one reveals that the actual decision on left arc $1 \leftarrow 4$ has nothing to do with deciding to apply the Left$_{1\leftarrow4}$ action (taken when no other choice was possible); instead, part of that decision is already taken when following Shift$_2$, even though it seems that it does not operate on token 1. Indeed, applying Left instead of this Shift prevents from later retrieving the $1 \leftarrow 4$ arc, and thus the decision of *not* applying Left matters. However, the decision on the second action of the derivation is not a decision on the head of *He*$_1$ only: as shown by the third row, applying a Right action in this slot has consequences for both the head of *He*$_1$ and that of *did*$_2$.

Overall, the arc $1 \leftarrow 4$ is not the result of a single decision but of a series of interacting decisions that may not even operate on tokens 1 or 4; conversely, in a given configuration, the decision on the next action is not about one given head, but about several of them, which are impacted in various extents depending on the effectively chosen action. To sum up, a derivation cannot be interpreted by singling out decisions on each edge.

As a result, some algorithmic refinements that are straightforward in other tasks, where the link between the decisions of the system and subparts of the output structure is direct (e.g. 'backoff to another model when the currently annotated word is unknown' in PoS tagging or language modeling), may be inapplicable in this context. In parsing the classification task must be considered with extra care, because some standard notions like 'the current word' can be ill-defined, and bear the risk of under-specified algorithms, unfair comparison, or misinterpretation of empirical results: for instance, a parser which produces more right dependencies than left dependencies is *not* a model whose parameters weight more RIGHT actions than LEFT actions.

3.4 Improving the classifiers: from the averaged perceptron to Stack-LSTMs

Since Collins and Roark (2004), the main classifier used to predict transitions in parsing has been the averaged multi-class perceptron.

The multi-class perceptron is a linear model that scores each possible decision t in a given configuration c, by means of a cross-product between a feature vector ϕ representing the configuration, and a weight vector w. In transition-based parsing, the next transition to apply is thus:

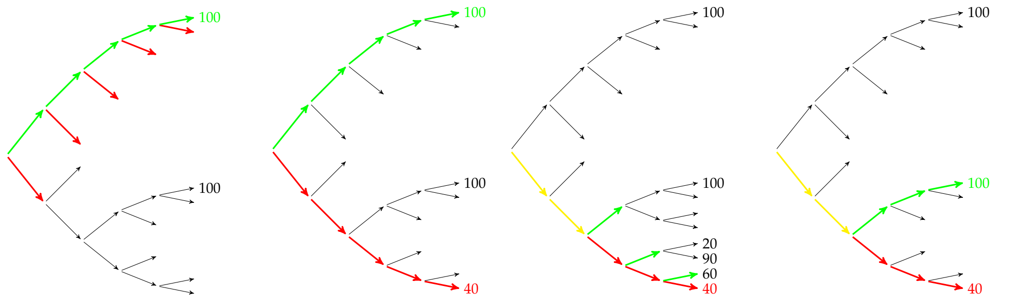
$$t^* = \text{argmax}_{t \in \textsc{Valid}(c)} \, w \cdot \phi_{c,t}$$

where $\textsc{Valid}(c)$ denotes the set of transitions that can be applied in configuration c (that is, whose preconditions are met).

The weight parameters are trained using annotated data. For each training sentence, a derivation leading to the gold tree is given by an *oracle*; for each configuration along this derivation, the predicted action t^- is compared to the gold action t^+ (see Figure 3.4a). If they differ, the perceptron update rule is applied:

$$w \leftarrow w + \phi_{c,t^+} - \phi_{c,t^-}$$

In the averaged perceptron variant (Freund and Schapire, 1999), all values taken by the w vector throughout training are remembered, and the value retained for w at the end of the training is not its last value, but the average of all its values through history. This improves online training by alleviating its bias toward the last configurations seen. Averaging does not alter the convergence guarantees of perceptron updates, but it makes the long-term effect of each update harder to interpret than for standard training. For instance, if two sentences in the training data lead to opposite updates, in standard training their effects cancel each other out, while with averaging, the final value of the parameters depends on the time that has elapsed between both sentences: the longer the delay, the more important the first update is in the average.



(a) Greedy static (b) Beam static (c) Greedy dynamic (d) Beam non-deterministic

FIGURE 3.4: Representation of the search space and the effected updates, for four types of search/oracle. Edges represent decisions, and numbers indicate the UAS of the parse produced by final derivations. The updates penalize the red and reward the green decisions, while the yellow ones correspond to correctly predicted actions and do not intervene in updates.

The last few years have seen the emergence of neural networks in NLP; following Henderson (2004)'s pioneering proposal, they have also been successfully introduced in the parsing field as an alternative to the averaged perceptron. At the time of the writing, neural dependency parsers have been designed with either feed-forward networks (Chen and Manning, 2014; Andor et al., 2016), whose comparative accuracy is reported in Table 3.6, or a subtype of recurrent neural networks, Stack-LSTMs (Dyer et al., 2015).

	Classifier	UAS	Speed (sent/s)
MaltParser	Averaged perceptron	89.9	560
Chen and Manning (2014)	Feed-forward neural network	92.0	1,013

TABLE 3.6: Effect on UAS and parsing speed of introducing neural networks with a precomputation trick, as measured by Chen and Manning (2014) on the Penn Treebank. Here both parsers use the ARCSTANDARD system.

In parallel to these improvements, notable work has been done on the vector representations that are fed to the classifiers. For the averaged perceptron, it concerns the feature templates that are used to combine simple atomic features into tuple features, and thus compensate for the linearity of the model (Zhang and Clark, 2008; Huang and Sagae, 2010; Zhang and Nivre, 2011); the popular templates of Zhang and Nivre (2011) are described in Table 3.7. As for neural networks, research on representations for parsing has concerned the embeddings of PoS tags and relation labels (Chen and Manning, 2014), OOV words (Dyer et al., 2015), in-stack subtrees (Dyer et al., 2015) and character-based word embeddings (Ballesteros et al., 2015).

	Standard templates
1 word	w, p and wp for S_0, N_0, N_1, N_2
2 words	$wp{\cdot}wp$, $wp{\cdot}w$, $w{\cdot}wp$, $wp{\cdot}p$, $p{\cdot}wp$, $w{\cdot}w$ and $p{\cdot}p$ for $S_0{\cdot}N_0$; $N_0p{\cdot}N_1p$
3 words	$p{\cdot}p{\cdot}p$ for $N_0{\cdot}N_1{\cdot}N_2$, $S_0{\cdot}N_0{\cdot}N_1$, $S_{0h}{\cdot}S_0{\cdot}N_0$, $S_0{\cdot}S_{0l}{\cdot}N_0$, $S_0{\cdot}S_{0r}{\cdot}N_0$, $S_0{\cdot}N_0{\cdot}N_{0l}$
	New templates with rich non-local features
Distance	$S_0w{\cdot}d$, $S_0p{\cdot}d$, $N_0w{\cdot}d$, $N_0p{\cdot}d$; $S_0w{\cdot}N_0w{\cdot}d$, $S_0p{\cdot}N_0p{\cdot}d$
Valency	S_0wv_l, S_0pv_l, S_0wv_r, S_0pv_r, N_0wv_l, N_0pv_l
Unigrams	w and p for S_{0h}, S_{0l}, S_{0r}, N_{0l}; l for S_0, S_{0l}, S_{0r}, N_{0l}
Third-order	w and p for S_{0h2}, S_{0l2}, S_{0r2}, N_{0l2}; l for S_{0h}, S_{0l2}, S_{0r2}, N_{0l2};
	$p{\cdot}p{\cdot}p$ for $S_0{\cdot}S_{0h}{\cdot}S_{0h2}$, $S_0{\cdot}S_{0l}{\cdot}S_{0l2}$, $S_0{\cdot}S_{0r}{\cdot}S_{0r2}$, $N_0{\cdot}N_{0l}{\cdot}N_{0l2}$
Label set	S_0ws_l, S_0ps_l, S_0ws_r, S_0ps_r, N_0ws_l, N_0ps_l

TABLE 3.7: Feature templates proposed by Zhang and Nivre (2011). S_0 is the top stack element, N_0-N_2 the 3 first buffer elements. w, p and l stand for word, PoS tag and label. d is the token distance from S_0 to N_0. v_l (v_r) is the number of left (right) children, whose labels are s_l (s_r). $_h$ and $_{h2}$ denote the head and its own head, $_l$, $_{l2}$ ($_r$, $_{r2}$) the first and second leftmost (rightmost) children.

3.5 Beam search and global training

One major challenge transition-based parsing has to face is the locality of the decisions. Indeed, the dependency structure of a free text sentence often implies long-distance interactions: interpolated clauses, long dependencies, high arity nodes, multi-clause sentences, etc. In such cases, the classifier needs information from a distant context to decide on the next action to take. While Zhang and Nivre (2011)'s feature templates introduce some non-local information, they do not cover all the potentially relevant nodes, which can be arbitrarily far.

3.5.1 Globalization of local parsers

To overcome this problem, the standard strategy, first applied on transition-based dependency parsing by Johansson and Nugues (2006), is to use *beam search*: instead of predicting the transition sequence greedily, by selecting the one-best action at each step, with beam search the parser maintains an agenda of the k best (partial) derivations so far, exploring new transitions from all of them at once. To discard the worst hypotheses in the beam at each step, and maintain its size at k, the derivations are compared using the sum of all the transition scores (*local scores*) in the sequence. Thus, each partial analysis is assigned a *global score*, and in the end the best parse is selected as a whole; this closes the gap to the global decisions taken in graph-based parsing.

3.5.2 Role of the beam

The effect of beam search on the parsing process can be interpreted in four ways.

As delaying difficult decisions: Some decisions cannot be taken locally because they require information that will only be available later. By keeping in the beam most of the paths from this point on, the parser is able to wait until it knows the future cost of each action: the earlier decisions are settled when a high score is assigned to some later, more informative features.

As a means of error recovery: The beam marginally compensates for a model that is not perfectly trained. Indeed, if two actions have very tight scores, it may be that the difference results from a training error, in which case it is safer to wait for a later decision whose margin is large enough to be reliable. With greedy parsing that takes decisions immediately, any error is unrecoverable.

As a solution to the label bias problem (Bottou, 1991; Lafferty et al., 2001): Even with a perfectly trained model, it may be mathematically impossible to model some dependency structures, because the globally optimal action is truly locally suboptimal. In such cases, dead end actions can hide the optimal derivation, which is currently low-scored, and mislead a greedy parser; beam search prevents this effect by exploring the neighbourhood of the locally optimal derivation.

As modeling actual ambiguities: In some cases, sentences are truly ambiguous at start, even for a human being. When the wrong interpretation remains dominant for a large part of the sentence, it is referred to as a *garden path sentence* (Frazier, 1979). In this case, a greedy parser will likely stick to the wrong interpretation. Thus, keeping several hypotheses in the beam enables to maintain concurrent interpretations of the beginning of the sentence, up to the disambiguating point.

3.5.3 Structured perceptron

On the theoretical ground, using beam search and global scores has a strong impact on the classifier: it turns the standard perceptron into a structured perceptron (Collins, 2002). In other terms, the classification task is no longer to predict actions, but rather to predict *complete derivations*. Given a complex input (a sequence of words) as before, the classifier must now directly produce a complex object (a parse).

This change of task has been shown by Zhang and Nivre (2012) to strongly impact training: because of the discrepancy it introduces between what the parser trains for and what it predicts at test time, when search is global, training must be global as well. Otherwise, beam search is only detrimental, as shown by their measures reported in Table 3.8.

Following Zhang and Clark (2008), global training is performed by updating the weight vector with the perceptron rule only once for the whole sentence, represented as a global feature vector. In practice, thanks to additivity, the global feature vector

UAS	Local [train]	Global [train]
Local [test]	89.04	87.07
Global [test]	79.34	92.27

TABLE 3.8: UAS drops obtained by Zhang and Nivre (2012) on the Penn Treebank with the ZPar parser, when mixing local and global versions of the perceptron. The local one corresponds to a greedy parser, and the global one to using a beam of size 64.

of a derivation can be defined as the sum of the feature vectors of all its transitions: it results that its cross-product with the weight vector is indeed the same as the previously defined global score (sum of all local scores), and that a single update on a pair of global vectors is equivalent to a series of local updates on pairs of local vectors, for each transition in the sequence. Consequently, the main difference between local and global training is the configurations which are updated on, and how they are computed: in local training, at each step of greedy search, a pair of configurations is selected, an update takes place and parsing goes on, while in global training, a full beam search is first performed, and only then the update configurations are extracted from the predicted derivation, which is not the same as the greedy one (see Figure 3.4b).

Using the same score and search strategy at training and testing times ensures that the objective function remains the same, and avoids the mentioned discrepancy. However, it introduces a strong binding between the search and update procedures, which are interleaved during training, and the necessity to keep them at most consistent to avoid discrepancies manifests itself in many subtle ways.

3.5.4 Partial updates

Unfortunately, global training itself is not sufficient to make the model accurate. Indeed, Collins and Roark (2004) have pointed out the need for another refinement, which is to perform model updates on partial derivations, before reaching a final state. Huang et al. (2012) analyze the issue by recalling that even if it has a better coverage, beam search remains, just like greedy search, a kind of inexact search. Following Daumé III and Marcu (2005), they distinguish two sources of parsing errors: modeling errors and search errors. The model induces errors directly when it scores a wrong derivation higher than the correct one, in which case the wrong parse is predicted. By contrast, wrong parses are attributed to search errors when the model has indeed assigned the best score to the correct derivation, but it was not found by the parser because it was discarded early from the beam, hidden by a lot of locally better looking derivations. However large the beam is, for long sentences the search will remain inexact, and such errors can occur. They are specific to beam search though, since a greedy parsing error can always be attributed to a locally wrong score.

The unfortunate consequence of search errors is that when an update happens due to a search error, it is invalid: indeed, the model already scores the correct derivation high (or at least higher than other candidates), so there should be no increase. Such updates only add noise to the model. The risk is a higher complexity of the model, and training may even diverge, because convergence is only guaranteed for valid updates, which fix an actual error of the model (a *violation*). However, invalid updates occur only when the score of the correct derivation is temporarily low, making the hypothesis fall off the beam, and then goes up again but with no possible return in the beam: the invalidity of a full update is only due to the last part of the derivation, which is better scored. So, the strategy proposed by Collins and Roark (2004) is an *early update* on partial derivations, as soon as the correct derivation falls off the beam; at that point the model is effectively wrong, and the corresponding configuration constitutes a violation. Alternatively, Huang et al. (2012) propose an improved update criterion, *max-violation*, that targets the step where the score of the correct derivation is at lowest compared to the top of the beam, which is also a violation. The main advantage is a faster convergence than *early update*, thanks to a broader coverage of the sentence. Table 3.9 displays the empirical effect of both strategies.

Update criterion	Convergence time		UAS
Full update	1 it.	0.4 h	79.14
Early update	38 it.	15.4 h	92.09
Max-violation	12 it.	5.5 h	92.18

TABLE 3.9: Effect of partial derivation updates on convergence time and accuracy, as reported by Huang et al. (2012) on the Penn Treebank.

3.5.5 Use in recent parsers

Despite recent advances in greedy parsing, the issues addressed by beam search remain: Andor et al. (2016) show how the label bias problem theoretically holds even with lookahead features of any range. They consequently apply global training even to recent neural models, by means of global normalization in the objective function. Their gains are reported in Table 3.10.

UAS	Locally normalized	Globally normalized
Beam size = 1	92.95	–
Beam size = 32	93.59	94.61

TABLE 3.10: UAS gains obtained by Andor et al. (2016) on the Penn Treebank, when adding global normalization to a feed-forward neural parser.

An alternate solution proposed by the literature is the unbounded lookahead provided by Stack-LSTMs (Dyer et al., 2015): the LSTM provides a compact encoding of information extracted from the whole sentence, with both unbounded lookahead and unbounded lookbehind. While this model strongly alleviates the locality of greedy parsers, it is unsure whether it can handle actual ambiguities and recover from errors, as beam search does.

3.6 Dynamic oracles

This section presents another recent line of research, which also concerns training, but from a different point of view. Indeed, a perceptron update relies on a pair of configurations: it penalizes the predicted one (negative configuration) and rewards the correct one (positive configuration). And while the previous section handles the question of which negative configuration to update *from*, I now move on to the question of which positive configuration to update *to*.

The oracle is specifically the parser component which, given a configuration at training time, returns the gold action to apply, in order to produce the reference dependency tree. This oracle then plays a role in both the error criterion (an error is detected whenever the predicted action differs from the oracle one) and the update process (the configuration provided by the oracle is used as positive configuration).

The standard strategy to implement this component is to preprocess the whole treebank to extract gold derivations prior to training; in case of spurious ambiguity, heuristics are used to select a single reference derivation per sentence. The training data composed of a series of sentence/tree pairs is effectively substituted by a series of sentence/derivation pairs, which constitute the only training data. The reference derivation is then used by greedy parsers to identify the best action at each step, and by beam parsers to detect when it falls off the beam.

In the frame of greedy parsers, Goldberg and Nivre (2012) introduce a new way to compute oracles, with *dynamic oracles*. By contrast, they call *static* the oracles resulting from a precomputation.

3.6.1 Issues of static oracles

The main motivations for dynamic oracles are two flaws of static oracles, brought to light by Goldberg and Nivre (2012).

First, static oracles are *deterministic*. Indeed, for the same sentence example the reference derivation will always be the same, whatever the current value of the model and current state of the parser are. However, in case of spurious ambiguity, it may happen that the parser produces the correct tree, but an error is still raised, because the predicted (correct) derivation differs from the one chosen by the oracle, which acts as the only reference. The update that necessarily ensues is then prone to introducing new errors, even though the model is already good. Because of the exponential size of derivation equivalence classes, it is not possible to acknowledge all correct derivations, when using precomputed references.

Second, static oracles are *incomplete*. Transition-based parsers face the known issue of error propagation (McDonald and Nivre, 2007): at test time, as soon as they stray from the gold derivation, they start making worse and worse decisions. Indeed, in such cases they can see features that are not possible in the space of gold configurations and that they consequently have not seen at training time; the classifier has no clue on how to score them. It has ended up in an unknown territory, in which it is not trained to make good decisions, so it simply acts randomly. A straightforward solution is to make the classifier aware of that territory, by extending training to the *suboptimal space* (all the configurations that exist only after an initial error). But this is forbidden by the incompleteness of the oracle, i.e. the fact that it is not defined properly for every configuration, but only for those along the gold derivation, in others terms the *gold space*. In the suboptimal space, from which the gold parse is unreachable, there is no reference, and no possible training. Choi and Palmer (2011) have actually already pointed out the unknown territory issue and experimented with suboptimal training, but they do not address the case when the gold tree is unreachable, simply resorting to a heuristic construction of a suboptimal treebank.

The purpose of dynamic oracles is to solve both these issues at once. Formally, a dynamic oracle is defined as a complete non-deterministic oracle.

3.6.2 Training with dynamic oracles

The strategy retained by dynamic oracles is to tailor the reference action to each configuration. More precisely, these oracles consider that an action is incorrect if it degrades the UAS, i.e. if it introduces (now or later) an incorrect edge.[4]

Formally, if each configuration is associated with a UAS_{max}, the maximum UAS value that can be achieved by any of its successor derivations, then the action cost is defined as the difference between the current UAS_{max} and the future UAS_{max}. The best decision in that configuration is the one providing the best hopes for UAS, i.e. when UAS_{max} is left unchanged. So the best action is that with zero cost; by definition of UAS_{max}, in all configurations at least one action has zero cost, and there may be several in case of spurious ambiguity. Hence, when asked for a reference action, the dynamic oracle simply returns the set of zero-cost actions.

[4]This assumes of course that such an edge cannot be corrected; some non-monotonic transition systems are indeed able to revise some of their earlier decisions.

Non-determinism is immediate with such computation, because model updates happen only when the predicted action has non-zero cost. If it is zero, nothing happens, whether the chosen gold action is unique or not. In practice, non-determinism provides a small but consistent improvement in accuracy, as shown in Table 3.11.

For suboptimal training, Goldberg and Nivre (2012) follow a strategy close to Searn (Daumé III et al., 2009), that gradually introduces a percentage of exploration: after a few iterations, the parser will sometimes choose to apply the gold action, sometimes the predicted one, which may be suboptimal. Thus, the suboptimal space is known to the classifier, which partially trains there. Figure 3.4c illustrates a case of full exploration. As can be seen in Table 3.11, the exploration strategy alleviates error propagation and brings significant improvements. Note that this effect distinguishes from error recovery, in the sense that it does not fix an already incorrect edge, but only allows the system to pursue parsing without being impacted by an earlier error.

UAS	Static oracle	Dynamic oracle
Gold space training	89.88	90.18
Suboptimal space training	–	90.96

TABLE 3.11: UAS gains obtained by Goldberg and Nivre (2012) on the Penn Treebank, when using non-deterministic oracles and training in suboptimal space.

3.6.3 Practical computation of action costs

To preserve the efficiency of transition-based parsers, action cost computation, which happens at each step, should be done in nearly-constant time. Hence, exhaustive search to compute maximal scores is not feasible, and action costs must be computed directly.

To this end, Goldberg and Nivre (2013) introduce the concept of *arc decomposition*: a transition system is *arc-decomposable* if in every configuration, all the gold dependencies that are still reachable can be reached simultaneously by the same derivation. It ensues that for such systems, the action cost is simply the number of arcs that are directly forbidden by the action (that is, which could still be predicted but become impossible to output once the action is chosen), for instance if it removes the head word from the stack.

Goldberg and Nivre (2013) prove ARCEAGER to be arc-decomposable when the dummy token is at the end, and derive its cost, detailed in Table 3.12. They do the same for the ARCHYBRID and EASYFIRST systems, with a dummy token in first position.[5]

[5]It can be noted that since ARCHYBRID has no spurious ambiguity, a non-deterministic oracle may appear less useful, but it still matters, because in the suboptimal space the references may not be unique, even without spurious ambiguity.

SHIFT	$(\sigma,\ b\|\beta)$	$\sigma \curvearrowright b$	$\rightsquigarrow$	b if h^*_b is in stack
	$(\sigma,\ b\|\beta)$	$\sigma \curvearrowleft b$	$\rightsquigarrow$	children of b that are in stack and unattached
LEFT	$(\sigma\|s, b\|\beta)$	$s \curvearrowleft \beta$	$\rightsquigarrow$	s if h^*_s is in buffer but not on top
	$(\sigma\|s,\ \beta)$	$s \curvearrowright \beta$	$\rightsquigarrow$	children of s that are in buffer
RIGHT	$(\sigma,\ b\|\beta)$	$b \curvearrowleft \beta$	$\rightsquigarrow$	b if h^*_b is in buffer but not on top
	$(\sigma\|s, b\|\beta)$	$\sigma \curvearrowright b$	$\rightsquigarrow$	b if h^*_b is in stack but not on top
	$(\sigma,\ b\|\beta)$	$\sigma \curvearrowleft b$	$\rightsquigarrow$	children of b that are in stack and unattached
REDUCE	$(\sigma\|s,\ \beta)$	$s \curvearrowright \beta$	$\rightsquigarrow$	children of s that are in buffer

TABLE 3.12: Computation of action cost, for the ARCEAGER system: cases when an arc is forbidden. Tokens with faulty attachments are indicated in each case; h^*_w denotes the gold head of word w.

While Goldberg and Nivre (2013) prove that ARCSTANDARD is not arc-decomposable, Goldberg et al. (2014) show that it does not prevent from deriving the action cost for such systems. It is only harder to define, and the computation can be arbitrarily complex; Goldberg et al. (2014) do it for ARCSTANDARD using dynamic programming.

I have similarly studied the matter for ARCEAGER when the dummy token is in first position, and I will show in Chapter 7 that in this case it is *not* arc-decomposable; I propose however another way to still derive its action cost.

3.6.4 Use in recent parsers

In the most recent works, dynamic oracles are still considered as a useful component of parser training. Indeed, they have been successfully applied to the Stack-LSTM parser by Ballesteros et al. (2016), showing similar improvements as for other greedy parsers (see Table 3.13).

UAS	ARCSTANDARD	ARCHYBRID
Static	93.04	92.78
Dynamic	–	93.56

TABLE 3.13: UAS gains obtained by Ballesteros et al. (2016) on the Penn Treebank, when extending the Stack-LSTM parser with dynamic oracles.

3.6.5 Differences with beam parsing

Dynamic oracles have not been extended to beam parsers yet, but Björkelund and Nivre (2015) have explored the use of non-deterministic oracles for beam parsers (illustrated in Figure 3.4d), in the case of non-projective parsing. However, they do not report significant improvements, which lets them suggest that since both methods are based on the idea of neighborhood exploration, they may be redundant.

It is true that both dynamic oracles and global training involve encountering suboptimal configurations at training time. However, according to the picture I have drawn above, a strong difference remains: beam parsers only see suboptimal configurations as negative configurations, while dynamic oracles also use them as positive configurations. Indeed, a beam parser with a static oracle is only able to reward reference derivations and it has consequently no quantitative idea of each derivation's quality: similarly to greedy static oracles, there are only correct derivations and incorrect derivations, no derivation is deemed as incorrect but slightly better than another. The motivations for dynamic oracles consequently hold.

As a matter of fact, the question of dynamic oracles for beam parsers is explicitly mentioned as remaining open by both Goldberg and Nivre (2012) and Sartorio (2015), and no sound extension to such parsers has been proposed yet.

3.7 Conclusion: four open avenues

In this chapter, I have presented dependency parsing and the currently most efficient approach for this task, transition-based parsing. While the frame I have drawn here may appear like a very special case of parsing, it corresponds in fact to the most popular algorithms in the state-of-the-art of this field.

After a short review of the main transition systems (ARCSTANDARD, ARCEAGER, ARCHYBRID and SWAPSTANDARD), along with some of their main characteristics, I have presented an overview of recent research on transition-based dependency parsing, through three lines of work.

The first line is improving the underlying classifier. For long dominated by the averaged perceptron, the field has been recently turning to neural networks. As this change of paradigm was concomitant to the work of this thesis, I designed my algorithms to be classifier-independent: they can in principle apply to a parser using any classifier, considering this component as a black box with a predict/update API. Still, the experiments reported in this thesis are only performed with the averaged perceptron: systematically including the most recent classifiers would have required additional implementation work.

The two other lines of research are indeed not affected by the inner workings of the classifier, as they concern the outer conditions in which it is trained.

The second line asks from which configurations to update: this is the subtle combination of beam search and global training. I have pointed out how alleviating the locality of transition-based parsers implies tight interactions of the training and testing procedures. I have also motivated and described various ways to explore the search space to find errors to correct, a study that benefits a lot from the theoretical guarantees of the classification frameworks.

Finally, the literature of dynamic oracles addresses the configurations toward which to update. By redefining what correct actions are in a more general case, it provides a sound framework where updates are guaranteed exact. It also enables a higher flexibility in the training strategies, which has already been rewarded by important gains in accuracy. The main drawback of this extension is that it introduces limitations on the transition systems that can be experimented with, because not all of them allow to compute easily such oracles.

While the focus of this thesis is not parsing itself, its main ideas have been naturally nourished by the shortcomings of the field chosen as study case. In particular, I have identified in this literature a few open avenues that have guided both the questions I have addressed, and the methods I have entertained.

First, from the overview of transition-based parsing, the general lack of a unified formalism stands out. Indeed, the various successful improvements have resulted over the years in very different paradigms; this is clear for instance when considering how global training affects the searching and learning procedures, which do not follow the same course of action as in standard training. This gap is filled in Chapter 7, both on theoretical and practical grounds, with the release of PanParser, a modular parser which *by design* supports these various search procedures, oracles and transition systems under the same framework.

Second, the transition systems have very different properties, but they have only been studied on a per-need basis, and not exhaustively. For instance, arc-decomposition has been proved for ArcEager, but not with a dummy token in first position. And indeed, my work reveals that the distinction actually matters, because in that case the property does not hold.

Third, it is striking to note that beam search and dynamic oracles have been combined successively with many different parsers, using various classifiers and transition systems, but never with each other. This remains an open question in the literature, which I will also tackle in Chapter 7.

Last, recent improvements in all three lines of work have brought undeniable quantitative gains, but their qualitative effects have not been thoroughly studied.

The overall impression retained from this exploration of the parsing literature is that of a very active and fruitful field, with a large spectrum of algorithms and variants, but many remaining gaps due to the lack of exhaustive study. Unifying their formalism is a first step in this direction, but it will also require a marked effort on the qualitative analysis of parsing accuracy.

Chapter 4

Cross-lingual parsing

***

Contents

When considering linguistic similarities and universals, syntactic structures quickly emerge as a natural study case. Over the years, cross-lingual parsing has consequently become emblematic of cross-lingual transfer studies; thanks to these universals, the motivation for knowledge transfer is indeed easier to apprehend than in other tasks. As such, the state of the art of cross-lingual parsing deserves a dedicated chapter.

The heart of this chapter is Section 4.1, which shows how the main methods for cross-lingual transfer have been applied to the parsing task. Sections 4.2 and 4.3 address some practical considerations: treebank annotation schemes and evaluation. Section 4.4 adopts a larger perspective by carefully comparing this field with that of cross-domain parsing, and Section 4.5 recaps the review by assessing whether and how the various available resources have been used in cross-lingual parsing.

4.1 Methods for cross-lingual parsing

4.1.1 Direct transfer

In their seminal work, Zeman and Resnik (2008) address low-resource parsing with a direct transfer approach: they parse a target sentence using directly a source parser, but with some adaptations of word representations, for which they compare two strategies. The first is to delexicalize both the source data and the target input, which means excluding all lexical features at training time, thereby relying only on PoS tags of the words and their contexts. This way, all source knowledge (e.g. 'determiners depend on nouns') is directly applicable to the target PoS sequences, which share the same representations as the training data.

The second strategy is to keep representations in the source domain, i.e. training a source model in a completely standard way, but glossing the input at test time, so that the extracted features are not based on target words, but on their translations in the source language. This approach has the benefit of dispensing with the availability of a high-quality PoS tagger in the target language, but it requires parallel data or a bilingual lexicon instead, which is arguably harder to acquire. The later literature has mostly retained the delexicalized version anyway, which yielded the best results.

Since then, a large body of literature has revisited the direct transfer approach, on several aspects.

Multi-source Several works have noted that using multiple languages as sources improves cross-lingual transfer. This increases both the diversity and the amount of available data. But combining knowledge from multiple sources can be done in various ways.

The simplest method is treebank concatenation, as done by McDonald et al. (2011): PoS sequences from various sources are indistinguishable, and can be simply gathered into a single treebank. The main issue is that without further refinements, this method introduces a bias toward the source with largest treebank.

Another method for multi-source transfer, which gives a better control over each treebank's contribution, is model interpolation. Cohen et al. (2011) average the parameters of several source models, optionally with tunable mixture coefficients, while others rely on linguistic knowledge to guide the mixture, with source weights based on language genealogy (Søgaard and Wulff, 2012), language typology (Søgaard and Wulff, 2012) or language similarity (Rosa, 2015b).

The third category is output combination with MST reparsing (Sagae and Lavie, 2006). Rosa and Zabokrtsky (2015) apply this technique to combine the candidate parses issued by all source models. They obtain significant improvements over the treebank concatenation baseline, and even more with a weighted combination, following Surdeanu and Manning (2010)'s framework. Indeed, Rosa and Zabokrtsky (2015) propose a new metric for language similarity, KL_{cpos^3}, which computes the Kullback-Leibler divergence of the source distribution of PoS trigrams, with respect to the target one. Using KL_{cpos^3} (in practice, $\text{KL}_{cpos^3}^{-4}$) to weight the contribution of each source language biases the combination toward the most similar sources, which are presumably (but not always) the most reliable ones. Agić (2017) further refines this combination method with weights based on the Hamming distance of typological feature vectors, and on a combination of both similarity metrics.

Compared to multi-source combinations at training and at decision times, output combination provides a higher flexibility, in the sense that it combines subtrees from various sources and may for instance retain the high-level dependencies provided by one source (with similar word order) with the low-level dependencies provided by another source (with similar morphological richness). Another advantage is that the candidate parses are produced by self-consistent models: as explained in the previous chapter, each dependency results from complex interactions of several decisions which should stick together to actually produce the transferred dependency. Conversely, mixture decisions may follow the decisions of various models at various times, resulting overall in wholly different dependencies, that are not inherited from any source, despite the motivation of linguistic similarity.[1]

Data selection The literature has proposed data selection as another refinement of direct transfer. First, data selection occurs at the language level, when choosing a language for single-source transfer. Indeed, multi-source transfer is often on par with the single-best source, but sometimes single-source transfer remains better. In such cases, selecting the best source relies on the same kind of metrics as multi-source weighting: language similarity (Rosa and Zabokrtsky, 2015) or typology (Agić, 2017).

But data selection can also apply at a lower level. Søgaard (2011) explores per-sentence selection among multiple sources, while Søgaard and Wulff (2012) investigate a smoother approach than strict selection, with sentence-level instance weighting. Both works assume that the most relevant source sentences are the most probable ones with respect to the target language, which is estimated by a PoS language model (PoSLM) trained on target PoS sequences. This is another way to leverage PoS sequence similarity, but in a heuristic-free manner, yet harder to interpret than KL_{cpos^3}.

Bodnari (2014) pushes this idea further by adapting the selection, or weighting, for each target sentence to process. The underlying motivation is that depending on

[1]Using the experimental setup of Chapter 6, I have measured this effect to concern 7-9% of predicted dependencies, for transfer from English and French to Romanian, German, Russian or Finnish. To Japanese and Arabic, it reaches 13 and 15%.

the syntactic structures included in a given target sentence, the information needed to parse it calls on different linguistic properties, hence different source languages.

Synthetic data Direct transfer has also been exploited to bootstrap the creation of a new target corpus. Following McClosky et al. (2006)'s self-training framework, Zeman and Resnik (2008) annotate raw target data with the transferred parser and then retrain a new parser on both the source data and the newly annotated data; but they fail to improve on their best results. However, in McDonald et al. (2011)'s experiments, training only on the new target data performs well enough, but more importantly it allows to turn a delexicalized parser into a fully lexicalized one, which can in turn be fine-tuned on target data as any other monolingual parser. Täckström et al. (2013) also achieve significant improvements by combining self-training with ambiguous learning.

On a side note, treebank glossing (Zhao et al., 2009; Durrett et al., 2012) is also a way to improve direct transfer by building artificial target data.

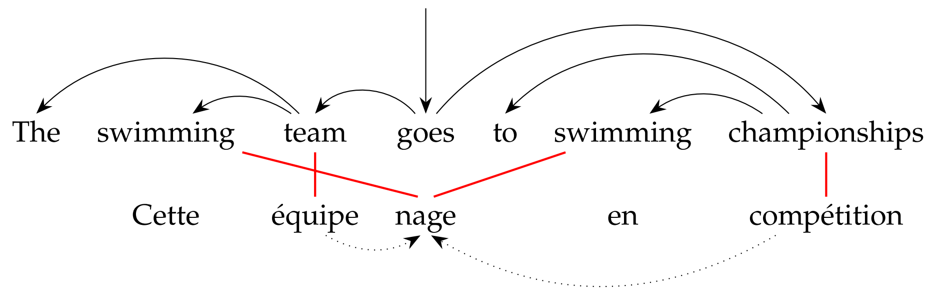
Shared representations Finally, some work has been done on the ground of shared representations. Täckström et al. (2012)'s cross-lingual word clusters improve direct parser transfer when used as additional features, which is in fact a way to preprocess the training and test data by annotating similar concepts on top of common PoS tags. In the same vein, Guo et al. (2015)'s direct transfer is transparent, thanks to cross-lingual embeddings: the model is trained and run in the same vector space.

4.1.2 Parser projection

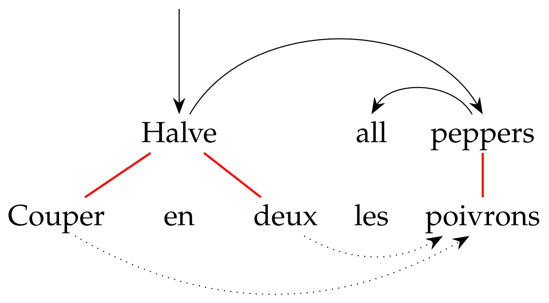
Parse tree projection The application of the annotation projection idea to parsing is largely based on the Direct Correspondence Assumption (DCA), first formulated and used by Hwa et al. (2002). It is the two-word extension of the assumption that aligned words share their labels: it states that in a pair of aligned sentences, if the source side contains the dependency $s_h \rightarrow s_c$, and the words s_h and s_c are aligned respectively with t_h and t_c, then the dependency $t_h \rightarrow t_c$ holds on target side.

However, vanilla parse projection suffers from a recurrent issue: DCA does not hold in general, but for less than half dependencies (Dorr, 1994; Georgi et al., 2012).

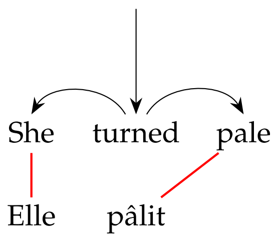
In most cases, unaligned words and multiple alignments hinder projection (see examples in Figure 4.1). This also happens in other projection tasks, but even more in parsing which suffers from alignment issues of both the head and the child. Sometimes projection is not hindered, but would result in an incorrect dependency, because of translational divergence; to prevent this, some works filter out low-confidence dependencies, on the basis of aligned PoS disagreement and projectivity constraints (Ganchev et al., 2009; Rasooli and Collins, 2015; Shen et al., 2016). This is notably the approach adopted by a joint work I took part in (Lacroix et al., 2016b), which I present in Chapter 7. For the purpose of filtering, it is also possible to leverage a baseline target parser (Li et al., 2014).



(a) Many-to-one alignment.



(b) One-to-many alignment.



(c) Unaligned word.

FIGURE 4.1: Example sentences where none of the dependencies can be projected, because alignments are not 1-to-1. The dotted edges violate single-headedness, so neither is transferred.

Anyway, it ensues that dependency projection often results in partial trees, but the state-of-the-art parsing algorithms are not suited to train on such trees, and would simply discard those training examples. Spreyer and Kuhn (2009) measure the data loss this filtering would result in, and Spreyer et al. (2010) discuss the fragmentation rate of these partial trees, as well as a case study on how partial projection affects the ability to handle subjects in Italian.

To address the partial trees issue more effectively, a number of solutions have been proposed in the literature.

One idea is to alter the projection step to actually get complete trees. Following the early work of Hwa et al. (2002), many resort to projection heuristics or completion rules based on projectivity constraints and dummy nodes (Hwa et al., 2005; Xia and Lewis, 2007; Tiedemann, 2014). Tiedemann (2015b) refines the dummy node approach and Spreyer and Kuhn (2009) extend it with fake root dependencies. Alternatively, Wróblewska and Przepiórkowski (2014) rather alter projection at its core, and propose a confidence-based extension of the projected dependency set. Their idea is to project more dependencies, along with scores based on alignment confidence, and then filter the extended set with confidence-based MST. In particular, unaligned target tokens receive candidate dependencies from all source tokens, but with very low confidence, which seamlessly solves the partiality issue. Finally, several works have considered post-processing, and complete the partial trees with a target parser bootstrapped by projection (Rasooli and Collins, 2015) or target annotated data (Shen et al., 2016).[2]

Another approach to this issue is that of McDonald et al. (2011), who take a step back and apply the projection idea to another process based on the DCA: they reduce the projection task to the reranking of target parses predicted by a bootstrapped parser, so that by design the whole process operates solely on complete trees.

In a last category, some works have preferred neither to tackle nor to avoid the partiality issue, but to keep it as a feature, and they train directly on tree fragments. Spreyer and Kuhn (2009) achieve this in their fake root framework by adapting the classifier loss functions to specifically remove updates on fake root attachments, thus learning only on the actually transferred dependencies. Ganchev et al. (2009) decide not to train directly on the projected partial trees, and rather turn to training guidance, using the partial trees as soft constraints for unsupervised parsing. Finally, Li et al. (2014) take an ambiguous learning approach and train on the whole forest of possible completed trees. The joint work I present in Chapter 7 (Lacroix et al., 2016b) pursues on this track, but by leveraging dynamic oracles it makes training on fragments even more transparent, without explicit modification of the training process.

Treebank translation The improved variant of treebank glossing, treebank translation, is actually another projection approach: as soon as the source sentence is translated, one gets a pair of parallel sentences, on which annotation projection is applied to actually translate the treebank. Otherwise put, it provides target synthetic data thanks to annotation projection, but applied on an already synthetic parallel corpus. In the parsing field, the translation approach was pioneered by Tiedemann et al. (2014) and Ramasamy et al. (2014).

Generally speaking, this approach gives rise to better projected parsers than standard projection, but it can fail on languages that are especially hard to translate into, like German or Swedish when English is the source (Tiedemann and Agić, 2016). Compared to treebank glossing, on one hand it provides sentences that are more natural, but on the other hand it adds projection issues; as a result, without

[2]Ma and Xia (2014) were the first to propose defaulting to another parser (the transferred delexicalized one) for gaps left by DCA-based projection, but they apply this idea in a training guidance framework.

further refinements the best method in fact depends of the language pair (Tiedemann et al., 2014). Compared to standard annotation projection, it does not alleviate the need for parallel data, but it affects the training time: source parser training is not needed anymore, target parser training is sped up by a much smaller synthetic data, but extra time is needed to train the MT system. Another practical difference is that it replaces the source parsing step, and the corresponding annotation noise, by machine translation and translation noise.

Weight projection Agić et al. (2016) have recently paved the road to a third kind of parser projection: the projection of weight matrices.

For word-level classification, Wang and Manning (2014) have indeed proposed to project score vectors instead of single labels, in order to 'transfer more information and uncertainty' through each alignment link. One way of seeing their approach is that instead of computing label scores, decoding, then projecting a single label per link, they rather delay the decoding step[3] and project all computed scores right away.

With this perspective, and considering the candidate heads as the label set, extending Wang and Manning (2014)'s method to graph-based parsing is exactly Agić et al. (2016)'s proposal for weight projection: they compute scores for all candidate heads, project them, and then decode the resulting scores on the target side. The case of parsing has two specificities, however. The first is that alignment links are involved twice: to identify the token to label, of course, but also to rewrite the score vector according to the candidate head identifiers on the target side. The second is that while weight projection is straightforward for tagging, thanks to per-token decoding, for graph-based parsing a global decoding is performed at the sentence level, which makes the decoding delay clearer, and allows to think this projection as a sentence-level matrix projection.

As a final remark on Agić et al. (2016)'s work, I note that they also exploit alignment confidence score, but for the purpose of multi-source voting and not to handle alignment issues like Wróblewska and Przepiórkowski (2014). For these, they indeed resort to heuristic means: sentences containing unaligned words are discarded, and in case of multiple alignments they only retain the maximum score among all corresponding edges. As a comparison, Wang and Manning (2014) average the vectors transferred through multiple alignments.

[3]This is not a fully accurate description of their work because they in fact use the projected knowledge as a target-side training guidance, and consequently suppress the decoding step. However, I am only interested here in their projection process, which is also applicable to data space transfer and whose motivations hold. Similarly, Ma and Xia (2014) project edge scores but they do not formulate it as projection and rather use them as guidance.

4.1.3 Training guidance

I consider as training guidance all algorithms that first train a source model, then use its parameters to improve training of a target model. But target-side training can be either unsupervised or supervised, and the latter remains distinct from joint learning since the source parameters are fixed.

Many works address guided unsupervised training, but with different means. In a more evolved variant of their direct transfer model, Cohen et al. (2011) use the source parameters to initialize a subset of the target parameters (the unlexicalized ones), and then let unsupervised learning unfold on raw data, with the hope that the learned model will remain close to its initial value. But their initialization approach is not significantly successful, and most other works resort to projection and cross-lingual regularization. Ganchev et al. (2009) apply a posterior regularization framework to a smooth conservation of projected edges: it minimizes the *expected* KL divergence under linear constraints on posterior *expectations*, representing a minimum conservation rate of projected edges. Note that in contrary to their related works, they do not use raw data and unsupervised learning is performed on the target side of parallel data. With close but different means, Ma and Xia (2014)'s regularization objective is the entropy, and they minimize the KL divergence of the posterior distribution, with respect to the projected scores themselves; they further improve the method with entropy regularization on raw data.

There is also a method recently proposed by Schlichtkrull and Søgaard (2017), who use the projected scores themselves as supervision data for a specially-designed classifier, which learns to reproduce scores instead of label predictions. This work is harder to categorize, as it involves a supervised classifier learning from synthetic data, but this data does not consist in dependencies, but model scores. Still, their approach is similar to that of Ma and Xia (2014), in the sense that they learn structure on unlabeled data with a regularization on projected scores.

Regarding the guidance of *supervised* training, i.e. when some annotated data is available on the target side, I only report the work of Duong et al. (2015b), who regularize directly on the model parameters (based on PoS universals and a bilingual lexicon): a neural network is trained as usual on a small target treebank, but with a strong bias towards mimicking the source network, which helps in acquiring the missing knowledge. From the lack of further work, it appears that the guided supervised approach has been overshadowed by the rising field of joint learning.

4.1.4 Joint learning

Joint learning is generally regarded as an improvement over standard learning; in this case, it applies to multilingual versus monolingual learning. So in the cross-lingual parsing field, most works have targeted the selection of the parameters that could be shared across languages.

Berg-Kirkpatrick and Klein (2010) paves the way by experimenting with phylogeny-based sharing of PoS-based features, albeit in a fully unsupervised scenario. In supervised parsing, Naseem et al. (2012) propose a typology-driven method for selective parameter sharing, but they apply it to a new generative parsing system, that is designed specifically for the task at hand and consequently does not benefit from recent advances in parsing. However, later works have then reused and improved upon the idea of typology-based selective sharing by applying the method to existing discriminative parsers (Täckström et al., 2013; Zhang and Barzilay, 2015).

Duong et al. (2015a) also study selective sharing in the sense that they share the upper layers of Chen and Manning (2014)'s neural parser, which are supposed to convey more general information. They let only the embedding layer vary across languages, but also experiment with selective parameter regularization on that layer, with respect to a bilingual lexicon, identical PoS tags or relations labels.

Fully shared representations have then enabled fully joint learning, i.e. to train a single multilingual parser like MaLOPa (Ammar et al., 2016a). Let us comment in further details this last work, in particular in the light of Duong et al. (2015a)'s work. Aside from a different choice of base parser (stack-LSTM instead of feed-forward neural network), the differences of MaLOPa with Duong et al. (2015a)'s parser are that (a) the authors have a clear intention to experiment specifically with large-scale multilinguality, (b) the introduction of language embeddings helps to extract language-specific knowledge from a multilingual parser, (c) lexicon-based embedding regularization is replaced by Guo et al. (2016)'s pretrained shared representations (i.e. the combination of projected Brown clusters and pretrained multilingual word embeddings), and (d) regularization on PoS and label embeddings is replaced by full sharing.

This last choice is quite controversial, since Duong et al. (2015a) explicitly advise against full sharing of PoS and label embeddings, on the basis that the same PoS tag or dependency type can play different roles in different languages, like Korean verbs which sometimes behave as adjectives. Hopefully, the combination of language embeddings and language-specific fine-grained PoS may be sufficient to handle this issue, and distinguish various uses of the same tag at a higher level; however, the subset of languages chosen by Ammar et al. (2016a) does not allow to assess this. What is certain is that to achieve high-performance cross-lingual parsing, such peculiarities must be accounted for one way or another; this is the same motivation that underlies Kozhevnikov and Titov (2014)'s proposal of feature mapping.

4.2 Cross-treebank consistency

The journey from Zeman and Resnik (2008)'s handcrafted mappings to McDonald et al. (2011)'s work (in which the use of standardized PoS improves transfer but Danish turns out to be the worst source for Swedish, despite their marked similarities) ends up with McDonald et al. (2013)'s use of standardized dependencies for cross-lingual transfer (which restores intuitive results on related languages) and shows how much the field has been improved by common annotation guidelines.

This standardization has occurred gradually: first Petrov et al. (2012) have proposed a universal set of PoS tags (UPOS), then standardization reached the dependency annotations, with three separate projects (Tsarfaty (2013)'s Unified Stanford Dependencies, McDonald et al. (2013)'s UDT and Rosa et al. (2014)'s HamleDT) and at last the UD (Universal Dependencies) initiative has merged all these projects (Nivre et al., 2016).

Universal PoS directly enable better transfer: thanks to shared representations, more target patterns are identified as known in the source, and more knowledge can be transferred. As for universal dependencies, they rather help sound evaluation. Indeed, many earlier works in cross-lingual parsing were presumably underrated because of annotation inconsistencies across languages: when the target is actually well parsed, but according to an annotation scheme that differs from the target one, it is wrongly deemed as erroneous. Altogether, annotation consistency has provided accuracy improvements up to +20 UAS, according to Tiedemann (2014).

In fact, most of the recent studies cited above use some version of UDT or UD as experimental setup, and almost all of them use UPOS. This large-scale effort for consistent corpus annotation has indeed driven a lot the parsing and cross-lingual parsing communities. Before then, a lot of works included rule-based transformations of the treebanks to reduce divergences, or handcrafted mappings to coarse categories.[4] All that extra work made cross-lingual evaluation cumbersome for researchers, and held back until then the cross-lingual parsing field, which is now fully flourishing.

However, coming up with such annotation guidelines, which are both consistent and sufficiently informative for high-quality parsing, is itself a whole research field. UPOS have indeed evolved from an initial set of 12 tags in 2012, to 17 a few years later. Similarly, version 2.0 of the UD guidelines, published in 2017, alters the annotation scheme on various levels, which is a clear signal that this project remains a work in progress. Among the most controversial issues is the UD choice to make content words the heads of the relations, leaving the role of leaves to function words: while this improves a lot cross-lingual correspondence, it is not sure whether this scheme

[4]Zhang et al. (2012) have however reported promising results with automatically induced mappings.

is as informative, and as easy to parse, as the traditional guidelines (Rosa, 2015a; Lhoneux and Nivre, 2016; Wisniewski and Lacroix, 2017).

4.3 Evaluation of cross-lingual parsing

Evaluation of cross-lingual parsing has long focused solely on close, European and well-resourced languages, because they were the only ones for which test treebanks were available. This has prevented a fair assessment of our ability to parse low-resourced languages, because of an over-reliance on relatedness and syntactic correspondence, and the hasty assumption that some resources were easy to get even for low-resourced languages, like large high-quality parallel data. However, the growing UD project now allows for more diversity, including more than 40 languages from various families, dialects, and treebanks ranging from 20 sentences to more than 60,000.

Metrics Cross-lingual transfer is generally evaluated with accuracy metrics only. A handful of works have also considered computing learning curves and measuring the size of supervision data which would yield the same accuracy (and whose acquisition the transfer methods dispense with), but they remain the exception rather than the rule (Zeman and Resnik, 2008; Spreyer and Kuhn, 2009; Ganchev et al., 2009; Täckström, 2012; Agić et al., 2016). The same holds for extrinsic evaluation.

The most widely used metric in the field of cross-lingual parsing is UAS, whose advantage is, on top of its simplicity, to dispense with label divergence issues, among others. Nevertheless, with a now higher confidence in UD consistency, some works have recently started to include labeling in cross-lingual parsing, and use LAS as a metric (McDonald et al., 2013; Tiedemann, 2014; Zeman et al., 2017).

Baselines As a reference, Table 4.1 reports experimental results published in the literature for each type of cross-lingual parser, using universal data and either UAS or LAS. Without surprise, the methods which leverage the most resources (typically those adding bilingual data) are systematically the most successful; but in practice their availability is not guaranteed.

Preprocessing quality There has also been some debate on the preprocessing available at evaluation time and whether to use gold or predicted PoS tags (Tiedemann, 2015a; Tiedemann and Agić, 2016) and tokenization (Zeman et al., 2017).

On one hand, I have already stated the need to experiment with truly low-resourced languages, as their challenges may differ significantly from transferring in ideal conditions. In that case, the PoS tags that are actually fed to the parser

	Target	de	en	es	fr	sv
Supervised	standard	80.34	92.11	83.65	82.17	85.97
	coarse PoS	78.38	91.46	82.30	82.30	84.52
Direct delexicalized transfer (coarse PoS)	de	70.84	45.28	48.90	49.09	52.24
	en	48.60	82.44	56.25	58.47	59.42
	es	47.16	47.31	71.45	62.39	54.63
	fr	46.77	47.94	62.66	73.71	54.89
	sv	52.53	48.24	52.95	55.02	74.55
Annotation projection	de	–	53.80	61.34	62.32	68.20
	en	63.52	–	63.18	67.04	67.74
	es	60.65	50.10	–	68.81	65.79
	fr	62.49	53.88	68.15	–	64.83
	sv	63.83	52.36	63.29	66.12	–
Treebank translation	de	–	58.60	61.00	63.45	67.88
	en	62.67	–	64.58	68.45	68.16
	es	57.13	52.65	–	69.37	63.55
	fr	61.41	56.83	68.97	–	62.56
	sv	61.73	52.13	62.34	64.50	–

(A) Data space transfer: LAS computed by Tiedemann and Agić (2016) on UDT 1.0, using gold PoS tags. Rows indicate the source language.

Target	cs	de	es	fi	fr	ga	hu	it	sv	μ
Target only	43.1	47.3	60.3	46.4	56.2	59.4	48.4	65.4	52.6	53.2
Guidance	49.6	59.2	66.4	49.5	63.2	59.5	50.5	69.9	61.4	58.8
Joint learning	55.2	61.2	69.1	51.4	65.3	60.6	51.2	71.2	61.4	60.7
Joint + guidance	55.7	61.8	70.5	51.5	67.2	61.1	51.0	71.3	62.5	61.4

(B) Parameter space transfer (with a target treebank and a bilingual lexicon): LAS computed by Duong et al. (2015a) on UD 1.0, using gold PoS tags. The target treebank contains 3,000 tokens. English is used as source language for all targets.

Target	de	es	fr	it	ko	pt	sv	μ
Supervised	81.65	83.92	83.51	85.47	90.42	85.67	85.59	85.18
Direct transfer	58.56	68.72	71.13	70.74	38.55	69.82	70.59	64.02
Guidance	73.92	75.21	76.14	77.55	59.71	76.30	78.91	73.96
Guidance + unlabeled	74.30	75.53	76.53	77.74	59.89	76.65	79.27	74.27

(C) Parameter space transfer (with parallel and raw data): UAS computed by Ma and Xia (2014) on UDT 2.0, using gold PoS tags for the test data. English is used as source language for all targets.

TABLE 4.1: Baseline results for cross-lingual parsing. Note that results from separate tables are not comparable, as they use different data and metrics.

may stem from target systems, whose quality is already far from gold tags, or most likely from transferred PoS taggers, and yet Zeman et al. (2016) have shown a case of failure to transfer both tags and dependencies. Our understanding of cross-lingual transfer cannot be based entirely on unrealistic scenarios; experimental studies on cross-lingual parsing should take into account the bottleneck effect of low-quality PoS tags.

On the other hand, ideal experimental conditions are essential for detailed analysis, to evaluate in which measure transfer methods manage to convey syntactic knowledge across languages, and identify their flaws in this specific regard. I illustrate this on a practical example drawn from the literature: Agić et al. (2016) experiment with realistic low-resource parsing based on transferred PoS tags. In their evaluation on Quechua, even the best transfer method peaks at 27.48 UAS, which is around what can be achieved by attaching all words to their left neighbour.[5] But at the same time, the accuracy of the transferred PoS tagger is 19.49% in Quechua. So, does the low UAS mean that this preprocessing is simply too noisy to even consider parsing? That the state-of-the-art transfer methods fail in Quechua because of some linguistic peculiarities? Or that Quechua syntax differs so much from any other language that high-quality syntactic transfer is impossible to achieve? From this set of experiments, it is hard to say anything on parse quality itself, and for researchers to know if in Quechua there is an improvement margin on the parsing track, or if from now on they should target their efforts on improving tagging only.

Besides, it is more realistic to presuppose small target data, rule-based or data-driven systems based on a quick access to native speakers (Garrette and Baldridge, 2013) for preprocessing tasks, whose manual annotation is faster and can be bootstrapped by e.g. Wiktionary knowledge, than for complex tasks like syntactic parsing. A language of interest can well be truly low-resourced for parsing, but not that much for preprocessing.

In this work, I consider that both settings have their worth. Gold PoS evaluation is relevant for researchers interested specifically in cross-lingual parsing, as it assesses the ability of a given method to transfer syntactic knowledge, while predicted PoS evaluation provides a fair assessment of the state-of-the-art ability to parse low-resourced languages, and it better fits the needs of researchers interested in a given language, or on the effective usability of transfer methods. As my focus is on parsing, most experiments are conducted with gold preprocessing, but I conclude the thesis with a realistic evaluation of the method, to assess its usability.

[5] I measured 25 UAS on the Quechua SQUOIA treebank (Rios, 2015).

4.4 Relations with domain adaptation

Several works mention explicitly cross-lingual transfer as a form of domain adaptation (e.g. Wei and Pal, 2010; Prettenhofer and Stein, 2010; Søgaard, 2011). This perspective implies that both languages are part of the same super-language, which is in fact the underlying hypothesis of both direct transfer and multilingual models. However, I claim that both settings have two major differences.

One concerns input divergences. In intra-language domain adaptation, open class words differ, but one can expect at least shared function words in both domains, and can leverage them as pivot features (Blitzer et al., 2006). This is not the case in cross-lingual transfer, because function words have necessarily different wordforms.[6] Prettenhofer and Stein (2011) summarize this effect as an additional constraint on the cross-domain setting, whereby both domain have non-overlapping features; Fernández et al. (2015) state that this makes many domain adaptation methods inapplicable in the cross-lingual setting. Yet, shared representations (like UPOS or bilingual embeddings) are supposed to alleviate the lack of overlaps.

The second and most important difference is about labels, and the covariate shift assumption (Shimodaira, 2000), whereby both domains have the same label distribution but different distributions of instances (*covariates*), so that domain divergences are only due to the latter. At least in the case of parsing, this assumption seems reasonable in domain adaptation (because despite many OOV words, the underlying grammar is the same[7]) but not in cross-lingual parsing, because typological differences among languages induce differences in the underlying grammar. As such, the vast amount of domain adaptation methods that rely on the covariate shift assumption are mostly inapplicable to cross-lingual parsing. Figure 4.2 shows an example where the same covariate (a delexicalized sentence) has different labels in two Indo-European languages. Diverging dependencies are mostly due to typological differences:[8] genitive and adjective positions, and morphological richness, which implies rarer adpositions.

Additionally, when considering cross-lingual parsing as a special case of domain adaptation, the impossibility theorem of David et al. (2010) applies: it states that transfer will fail unless there exists a parser able to perform well on both. Yet, no delexicalized parser can perform well on both examples of Figure 4.2 (nor on many such pairs). As a result, cross-lingual parsing methods based on domain adaptation insights (i.e. direct transfer and multilingual models) can only perform well if they

[6]There are of course a few specific exceptions, like the preposition 'in' in several Indo-European languages, but overall, exact matching of the wordforms remains rare.

[7]Counter-examples are easier to find in PoS tagging: the same word can have different PoS categories depending on the domain. For instance, the word 'monitor' is in general a verb in news corpora, but a noun in technical documentation (Daume III and Marcu, 2006).

[8]Since both examples follow UD guidelines, the annotation consistency is not in question.

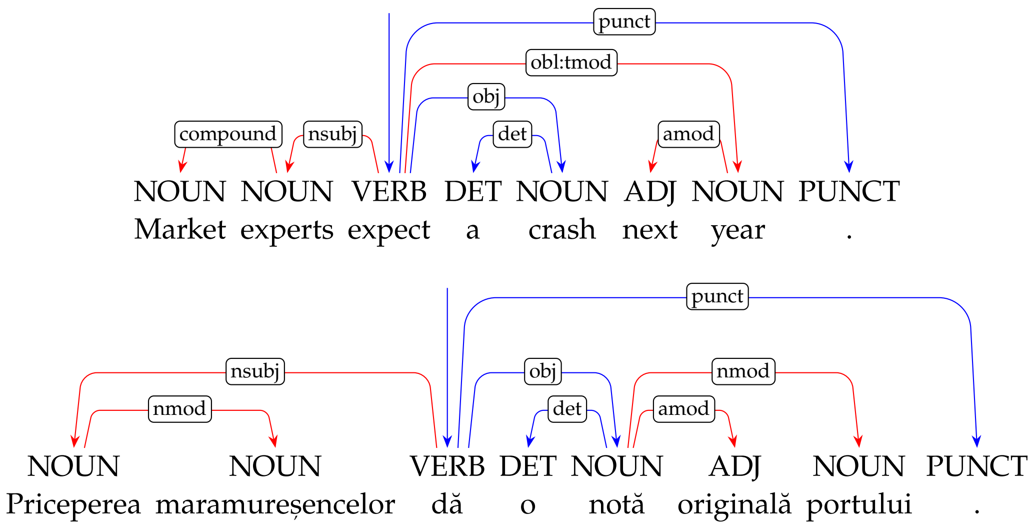

punct
obl:tmod
obj
compound
nsubj
det
amod
NOUN NOUN VERB DET NOUN ADJ NOUN PUNCT
Market experts expect a crash next year .
punct
nsubj
obj
nmod
nmod
det
amod
NOUN NOUN VERB DET NOUN ADJ NOUN PUNCT
Priceperea maramureșencelor dă o notă originală portului .

FIGURE 4.2: Example of a delexicalized sentence which has significantly different parses in English and Romanian. The Romanian sentence, which means *'The skill of the women of Maramureș brings an original note on their clothes.'*, is drawn from UD 2.0. Dependencies that are the same in both sentences are represented in blue, differences in red.

are augmented by knowledge, in the form of linguistic knowledge or target annotated data, in order to bind not only the covariates (Prettenhofer and Stein, 2011) but also the labels. The only alternative is to resort to transfer methods out of the domain adaptation spectrum, like projection, which do not make any assumption on the label distribution like covariate shift does. We will come back to such issues in Chapter 8.

4.5 Effective use of cross-lingual resources

Before wrapping up this chapter, I summarize the survey from the standpoint of how cross-lingual bridges are used in this literature. The previous section has indeed shown that the choice of resources can be crucial for the success of cross-lingual parsing.

Typology Typological information has been leveraged by several works, either to compute language similarity metrics or to guide parameter sharing (Naseem et al., 2012; Søgaard and Wulff, 2012; Täckström et al., 2013; Zhang and Barzilay, 2015; Agić, 2017). Soft parameter sharing is also enabled by typology-based language embeddings (Ammar et al., 2016a), yet in this case it actually degrades the accuracy.

However, to my knowledge, the literature only ever uses typological features as atomic features, to cluster or rank the languages, but the value of the feature itself is not used. To be clear, swapping all 'ADPs are preposed to NOUNs' and

'ADPs are postposed to NOUNs' values would make no difference in the typology-driven algorithms. Täckström et al. (2013) and Zhang and Barzilay (2015) actually use explicitly the *name* of the typological trait, but still not its value. This is under-optimal, since for instance in parameter sharing frameworks, adposition attachment is trained completely separately for languages with prepositions and languages with postpositions: the system is not able to learn only on the former and infer that the latter will have the opposite behaviour.

Language relatedness and universals Specific knowledge on language genealogy is generally used in the same way as typology, i.e. for sharing and to compute language similarity (Berg-Kirkpatrick and Klein, 2010; Søgaard and Wulff, 2012).

As for syntactic similarities that result from relatedness, they are clearly leveraged by all direct transfer methods. Morphological similarities are also used to a lesser extent by these methods, because closely related languages have often the same degree of morphological richness, and hence the same granularity for word segmentation. For this reason, related languages tend to have the same proportion of ADP$\curvearrowleft$NOUN or case dependencies for instance. Agić et al. (2014) also advocate for using rich morphosyntactic tagsets when languages are related; this can now be improved by enriching the shared representations with UD universal features, and thus leveraging morphological similarities transparently at the representation level.

Finally, linguistic universals are implicitly leveraged when performing direct transfer from a distance source, and cross-lingual embeddings rely on extra-linguistic universals.

Bilingual knowledge Parallel data has found two major uses in cross-lingual parsing: through its DCA property and to constrain the learning of shared representations. It has also been used indirectly, as a way to extract bilingual lexicons.

Regarding bilingual lexicons specifically, they have been used in various ways: for treebank glossing, to tie or regularize shared parameters, and also to learn shared representations.

Target knowledge Finally, the sparse knowledge available on the target side has been considered by many works. Among these resources, target raw data is the most emblematic: its uses range from self-training and unsupervised induction, to representation learning and language similarity measures.

Target annotated data also gets its share, as it is widely used for joint training of course, but also as a criterion for heuristic projection or for completing partial projected trees (Shen et al., 2016). This last use attests that target data is complementary to other resources: it uses target knowledge specifically for structures that cannot be projected.

Other kinds of target knowledge, notably human prior knowledge, have been used to prune the search space with constraints. For instance, Hwa et al. (2002) define less than 20 heuristic rules to improve transfer for specific languages, and Ganchev et al. (2009) get big improvements with only 2 rules.

In comparison to the picture of available cross-lingual bridges that I have drawn in Section 2.1, it appears that most of the mentioned resources have found their use in cross-lingual parsing. Only a few resources remain unexplored: the lexical similarity of related sources (cognates are never involved in the considered methods), crowd-sourced target knowledge (typically, Wiktionary article *contents*, not only page links and titles), and the actual value of typological features.

4.6 Conclusion: softer requirements, finer combination

This chapter has shown how the literature has applied the main strategies for cross-lingual transfer to the parsing field. I have exhibited and discussed some specificities that arise in this case: the possibility of late multi-source combination offered by MST reparsing, the issue of partial trees in annotation projection, and the adaptations required by sentence-level predictions which complicate fine-grained transfer.

I have also addressed the vexing topics of data consistency and fair evaluation, which are currently evolving in the right direction (and quite fast) thanks to a large community effort.

Careful comparison with the field of domain adaptation reveals that they are indeed close; but even if some inspiration can be drawn from domain adaptation methods, they do not apply in general, because the cross-domain and cross-language settings face different challenges and cannot rely on the same mathematical assumptions.

This literature review has finally allowed us to survey the actual use of each type of cross-lingual resource. It appears that most of them are used by the parsing literature, but none has provided a framework able to leverage all those resources at once, even less a framework that makes no assumption on the availability of a given resource.

I conclude this chapter with two general comments on the state-of-the-art of cross-lingual parsing. First, many have reported specific difficulties due to peculiarities of the parsing task. In particular, being a structured prediction task implies that the model behaviour through transfer is harder to interpret, and also that the legit transfer operations are much more constrained (see for instance the transfer failures due to DCA shortcomings). For these reasons, parsers in a cross-lingual scenario appear more as black boxes than e.g. PoS taggers, which makes them a harder, but even better study case to achieve my goal of designing generic and flexible transfer methods. This remark adds to my motivations for choosing dependency parsing as a challenging study case.

Second, I am struck by the fact that the main challenge addressed in multi-source settings is the estimation of each language's usefulness, as a whole. Even if the metrics are motivated by fine-grained patterns (for instance the prevalence of a given PoS trigram in both languages), and even if combination procedures have been designed down to the dependency level (Rosa and Zabokrtsky, 2015; Agić, 2017), only a handful of studies have actually proposed fine-grained contribution, adapted at the sentence level (Søgaard, 2011; Søgaard and Wulff, 2012; Bodnari, 2014). I will come back to that topic in the next chapter.

Part II

Cross-lingual parsing for low-resourced languages: an empirical analysis

Abstract

The following chapters investigate the strengths and weaknesses of state-of-the-art cross-lingual parsers, in comparison with monolingual ones. The purpose of this study is to identify in which ways the transfer techniques should, and can, be improved. To that end, extensive empirical analyses are conducted at coarse and fine grains, down to measures at the feature level. Emphasis is put on the usefulness of transfer techniques: how much do we gain when dispensing with building new annotated datasets? The complementarity of monolingual and cross-lingual resources is then demonstrated, both qualitatively and quantitatively, and it is connected with the marked diversity of syntactic contents. In this respect, three aspects of parsing related to treebank heterogeneity are explored: the lexicalization of dependencies and features, the difficulty to learn some dependencies and the effects of class imbalance. All these findings enable to develop new design guidelines for a better transfer framework, based on a cascading architecture.

Chapter 5

How good are we at cross-lingual parsing?

***

Contents

In this chapter and the next, I follow an analytical journey through the state of the art in cross-lingual parsing, in order to fully understand the strengths and weaknesses of this approach. This chapter consists in a general assessment of transfer usefulness, while in the next one I conduct a finer-grained study. The ultimate goal of both is to develop design guidelines for a better transfer framework, which will be built in later chapters.

The experiments and claims presented in this chapter are first motivated by a case study on Romanian in Section 5.1, which raises serious doubts on the efficiency of cross-lingual parsing and confirms them on the tagging task. To generalize these findings for other languages, I investigate the quantitative usefulness of transfer methods in a more systematic manner in Section 5.2. In spite of disappointing results, I exhibit the high potential of complementarity between cross-lingual resources and samples of target data, both empirically (Section 5.3) and linguistically (Section 5.4).

5.1 Assessing transfer usefulness in a case study

To get some hands-on insight on the usefulness of transfer methods, let us start by a case study on Romanian. The complete set of experiments that supports this section has been published in (Aufrant et al., 2016b), to which I refer the reader for further details and discussion.

My purpose in this study is to quantify the actual benefits of manually annotating target data, compared to what can be achieved without extra human work (that is, models transferred by automatic means). To achieve this, I experiment with state-of-the-art methods to transfer Romanian parsers (and taggers, for contrast) from several Romance languages.[1]

This choice is motivated by the belonging of Romanian to the Romance family, a family containing many closely related and well-resourced languages. However, this is not a guarantee for success, as Romanian has rare properties among its family: it is morphologically richer, has kept a case system inherited from Latin and uses more clitics[2] than other Romance languages. The language itself consequently presents several linguistic challenges.[3]

Data I use the French, Italian and Spanish UDT 2.0 as source treebanks. For Romanian I rely on the Romanian Syntactic Annotated Corpus (RSAC, Perez, 2014), which I mapped to the UPOS tagset of Petrov et al. (2012), and from which I randomly sample 2,500 training sentences (RSAC-train) for the supervised models and 500 test sentences (RSAC-test).

Additional resources for transfer consist in Europarl parallel corpora (Koehn, 2005), which are smaller for Romanian than other languages but still of significant size (typically 380,000 sentence pairs in English-Romanian, versus 2,000,000 in English-French), and a tag lexicon automatically induced from crowd-sourcing data. From WIKTIONARY[4] I extract (and map to UPOS) the possible PoS tags of 405,125

[1]I also performed similar experiments with English as a source, but they did not outperform transfer from Romance sources. See the corresponding paper for these results.

[2]Clitics are functional words which cannot stand on their own, but revolve around a host word (e.g. a noun or verb) and depend phonologically and often orthographically (by means of hyphenation) on that word. Romance clitics are mostly pronominal objects, but Romanian has also many clitic determiners and auxiliaries.

[3]An additional motivation for choosing Romanian was also to experiment in a more realistic scenario than with usual test sets; Romanian was indeed low-resourced at the time of these experiments (in 2015). However, this is not the case anymore, notably thanks to the UD project: after an initial release of 633 Romanian sentences in version 1.2 (November 2015), versions 1.3 and 1.4 (May and November 2016) quickly raised the treebank size to 6,347 and then 9,523 trees, thus placing Romanian in the better half of UD languages. In other tasks, Romanian still remains far behind many well-resourced languages, though.

[4]The Romanian WIKTIONARY (`https://ro.wiktionary.org`) covers a large vocabulary but only includes a few inflection tables and consequently does not contain most wordforms. For this reason I complete the data with the English WIKTIONARY (`https://en.wiktionary.org`), which covers a smaller Romanian vocabulary but with complete inflection tables.

wordforms, thereby achieving a coverage of 69% of the types in RSAC, which still remains lower than in other languages (80% to 90% for the languages considered by Wisniewski et al. (2014)).

Cross-lingual parsing In this study, I experiment with the method of McDonald et al. (2011) for cross-lingual parsing. As presented in Chapters 2 and 4, their work combines several ideas: beam parsing, multi-source combination (achieved by corpus concatenation), delexicalized transfer but also annotation projection. The method consists in several steps: it starts with delexicalized transfer (relexicalized by self-training on raw data), and then refines the parser to better fit the target syntax, using parallel data in a way similar to annotation projection. For each sentence pair, the beam hypotheses generated on the target side are compared to the source-side parse, and reranked according to a tree alignment score: the model is updated to rank first the hypothesis that best satisfies DCA with the source parse. This enforces cross-lingual agreement on sentence pairs, while keeping the parser in the neighbourhood of the initially transferred model.

The tree alignment score works as follows: each edge (both on source and target sides) yields a reward ($+1/2$) if the counterparts of its head and child in the other language satisfy DCA, and a penalty (-1) otherwise. In case of multiple alignments, the reward or penalty is received for each combination of head-counterpart and child-counterpart; unaligned words do not contribute. Romanian being morphologically richer than other Romance languages, in this work I use a slightly modified version of this metric that accounts for frequent many-to-one alignments.[5]

I rely on standard parsing and features: the parser is an unlabeled ArcEager with an average perceptron, a beam of size 8 and early update. I only use coarse PoS, and the feature templates are that of Zhang and Nivre (2011) (augmented with the 8 last actions). Relexicalization is done on the sentences of RSAC-train, thus enabling the comparison with the models trained on the same data. Parallel data from Europarl[6] is PoS-annotated by a perceptron-based tagger trained on RSAC-train (with 88.8% accuracy). Otherwise, gold PoS are used, both for relexicalization and evaluation.

The resulting UAS, reported in Table 5.1, show consistent gains when augmenting delexicalized transfer with this transfer method, which partly closes the gap to the supervised UAS. The reported scores are comparable to those obtained by McDonald et al. (2011) and McDonald et al. (2013), including the benefits of using multiple sources. However, looking at the learning curve of supervised parsers for

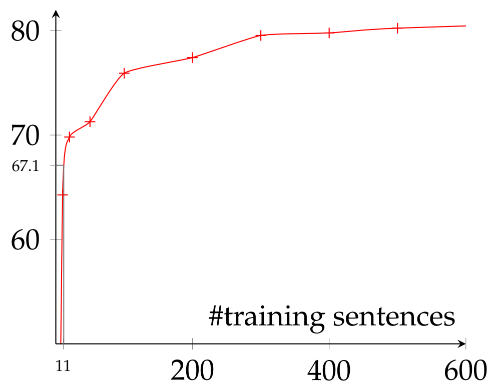
[5]Precisely, I add a small reward ($+1/4$) for edges between tokens aligned to the same token, instead of penalizing them. This models the fact that such tokens typically depend from each other, while keeping their impact on the parse quality low.

[6]In practice, I only use 80,000 sentence pairs: preliminary experiments with unlimited parallel data have proved detrimental, as the cross-lingual parser then moves far away from the initially transferred model.

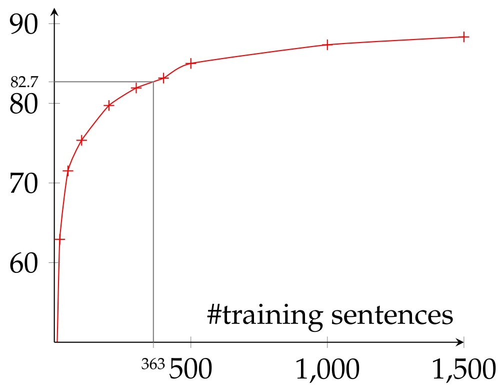
Source	fr	it	es	fr+it+es
Delexicalized	60.8	61.5	61.2	61.7
Full transfer	67.0	66.9	67.1	67.1
Supervised			82.7	

TABLE 5.1: UAS on RSAC-test of supervised and cross-lingual parsers from various sources.

increasing treebank sizes (Figure 5.1a), it appears that the best score achieved by transfer (67.1 UAS) could have been obtained by training a parser on less than 15 annotated sentences (more precisely, 11 sentences). This result is surprising and disappointing: although the technique of McDonald et al. (2011) actually improves the performance of a delexicalized model, on this dataset it only captures very few information and the annotation effort it saves is actually quite small.[7] Is this an issue of this method only, or of transfer in general? To explore this question, I also conduct similar experiments in cross-lingual tagging.



(a) Parser UAS.



(b) Tagger accuracy.

FIGURE 5.1: Monolingual learning curves, compared to cross-lingual best scores.

Cross-lingual tagging For tagging, I use the transfer method of Wisniewski et al. (2014). It is based on Täckström et al. (2013)'s approach that combines two sources of information: *token constraints* extracted from word alignments and *type constraints* extracted from crowd-sourced dictionaries. The type constraints are used to filter out the tags transferred through alignment links. Following Wisniewski et al. (2014), I use an history-based tagging model trained in the ambiguous learning framework. As they advocate, I also add handcrafted rules to describe the possible PoS tags of 15

[7]In general, the work required for manual annotation does not consist only in annotation itself, but also includes the development of annotation guidelines for the considered language. However, in scenarios where cross-lingual transfer is an option, it is unlikely that such guidelines will be designed from scratch: since the guidelines already designed for the source side concern a language with a relatively close syntax, they can be reused for the language of interest. One could also simply rely on generic guidelines like UD.

frequent clitics and function words, the tokenization and annotation of which differ between datasets.

Here again, parallel data is from Europarl, resized to 300,000 sentence pairs for each source setting, and type constraints are those extracted from WIKTIONARY.

The best cross-lingual model (obtained with either French or Spanish) achieves 82.7% accuracy. According to Figure 5.1b, a supervised model reaches this score with 363 sentences. Note that without the 15 handcrafted rules, the accuracy drops to 79.1% (184 sentences).

Discussion These observations strongly challenge the interest of cross-lingual transfer in general, and of cross-lingual parsing in particular: annotating 12 sentences is indeed easier (and performs better) than marked efforts to entertain complex transfer methods, using large amounts of parallel and monolingual data in other languages. Error analysis suggests several explanations for this disappointing finding.

First, an important source of errors is of course the discrepancy between the source and target annotation schemes. Romanian UD was indeed not available at the time of these experiments, so despite PoS mapping the source data does not follow the same guidelines as the test data (but target training data does, hence a gap between the two). The recent UD releases for Romanian should alleviate this, but note however that cross-treebank consistency does not solve the numerous inconsistencies due to the use of external resources like Wiktionary: out of the three Romanian particles, 'să' is annotated as a conjunction by Wiktionary, 'a' as a preposition and 'nu' as an adverb.[8]

Second, some syntactic structures can really not be transferred via the proposed methods, based only on PoS tags, word alignments and syntactic similarity. This is the case for instance of the Romanian preference for subordinates, where other Romance languages use infinitives. Taggers are less affected by this effect, since aligned PoS agreement holds more often than DCA.

Overall, it also appears that the considered transfer methods exploit only part of the available cross-lingual information. In these experiments, source and target have a strong lexical similarity which could prove valuable, Romance languages being partly mutually intelligible, and yet this similarity is never used in the pipeline.

[8]Sometimes even Wiktionary is not cross-version consistent: possessive determiners (in RSAC and UDT) are determiners in the English version and adjectives in the Romanian one.

Finally, these methods oversee the fact that a large part of the predictions concern closed classes (33% of RSAC tokens). They are indeed severely affected by cross-lingual learning,[9] yet such attachments are relatively deterministic (as suggested by their particularly high accuracies, when using full monolingual supervision), so that it is unclear whether these attachments require transfer learning at all, or even data-driven learning; tidbits of target knowledge could suffice. This observation supports the use of target knowledge on a more systematic basis; otherwise it seems like we are wasting large amounts of cross-lingual data to learn the closed-class parameters in an indirect, cumbersome, and yet suboptimal way. Besides, this also sheds light on the tagging results: removing some target knowledge (handcrafted rules) is equivalent to removing half of the annotated data (from 363 to 184 sentences); the method remains more rewarding than for parsing, but in fact it still exploits some target knowledge (albeit noisy), by means of crowd-sourcing. Consequently, this cross-lingual tagging method is also more successful because it exploits more target knowledge.

This hands-on experiment has shown that the usefulness of cross-lingual parsing does not meet our expectations: even 12 sentences can convey more information than what is actually transferred using sophisticated methods. However, despite residual annotation inconsistencies and cases where transfer is impossible, there seems to remain some hope for competitive cross-lingual models. But their success relies on a better use of the available information, such as lexical similarities or weak forms of target knowledge.

5.2 Investigating tiny treebanks as an alternative to transfer

According to the case study measures of the previous section, in Romanian exploiting even 12 annotated trees yields better results than a state-of-the-art transfer method. This finding brings back the idea of processing low-resourced languages with standard systems (with monolingual supervision), thus setting aside methods based on transfer: in the above case, if at least 12 trees are available, it seems preferable to use monolingual parsers over transferred ones. In this section, I conduct a systematic study on many languages to assess whether in the general case parsers trained on tiny treebanks are a good alternative to transfer, as they are in Romanian. This is expressed quantitatively as the minimum trainset size needed to prefer monolingual training over cross-lingual transfer.

[9]For instance, the fully supervised parser achieves 88 UAS on adpositions while this score drops to 81 when using transfer. This drop is notably important for determiners (from 93 to 79 UAS) and for the 3 particles (from 91 to 65 UAS). See (Aufrant et al., 2016b) for full results.

Tiny parsers In the literature, there is no consensus on a quantitative definition of the low-resource scenario, but in this work I choose to target the numerous languages where the expected treebank size is merely that of a sample, in the range of 10-20 sentences: these would typically be example sentences drawn from linguistics papers, or manual annotations resulting from a brief access to a native speaker. As a matter of fact, this is precisely the data size measured in the previous section, hence my doubts on the usefulness of transfer. So, in the following, I denote by 'tiny' the treebanks containing around 10 sentences, and by extension the parsers trained on such data. Thus, parsers trained on even 500 sentences are comparatively well resourced.

This choice is in contrast with the scenarios usually studied in the cross-lingual literature: when monolingual alternatives are considered, they actually start at 100, or even 300 sentences. I believe this is not realistic for truly low-resource parsing though: acquiring 300 sentences is already a significant effort, considering that it is around the size of several published treebanks.[10] In Lhoneux et al. (2017)'s work, treebanks from 1,000 to 14,000 tokens are referred to as 'tiny', but these sizes do not match the real-world needs to process the under-resourced languages I target.

One work stands out though: Spreyer et al. (2010) experiment with baseline parsers built upon 10 sentences and already achieve reasonable scores in comparison to transferred parsers, especially in Italian (57.72 UAS), even though they do not attempt to build a full parser, and restrict their baseline to local attachments. In this regard, the present study is a continuation of their work.

Data preprocessing For the following set of experiments, I use the Universal Dependencies 2.0 dataset (Nivre et al., 2017a,b). As I focus on a low-resource scenario, I first downsize each treebank to the 500 first training sentences,[11] and resample smaller trainsets of increasing sizes.[12] Since training is significantly unstable at that scale, all reported scores are averaged over five random samplings. UAS evaluation is done on gold PoS tags, excluding punctuation as usual.

5.2.1 Selecting a competitive tiny parser

Before performing the actual comparison, I first examine which parsing system is the best monolingual alternative to transfer. Indeed, not all parsers behave the same when data becomes scarce (Lhoneux et al., 2017), and on tiny data some may be more competitive than others with transfer: I search for a parser whose properties

[10]300 Irish sentences (Lynn et al., 2012), 600 for Tamil (Ramasamy and Žabokrtský, 2012), 371 in Slavomír Čéplö's Maltese treebank (Tiedemann and Plas, 2016).

[11]I similarly downsize the validation sets, used only for early stopping, to their first 10 sentences. I do not alter the test sets. I only experiment on the 56 treebanks that are large enough to apply these samplings procedures. `ar_nyuad`, whose complete data is under license, is also excluded.

[12]Resulting trainsets contain 5, 10, 20, 50, 100, 200 and 500 sentences.

allow good exploitation of this amount of data specifically, and to this end I perform a systematic evaluation of several parsers.

I consider three state-of-the-art dependency parsers: UDPIPE (transition-based parser, with a feed-forward neural classifier, i.e. embedding-based), PANPARSER[13] (transition-based parser, with an averaged perceptron classifier, i.e. feature-based), and MSTPARSER (graph-based parser). I use version 0.5.1 of MSTPARSER with default parameters. For UDPIPE, I use version 1.1 (Straka and Straková, 2017) with the same hyperparameters as Straka (2017), but without the word embeddings pretrained on massive monolingual data. For PANPARSER, I use the greedy ArcEager version, with a dynamic oracle and the feature sets of Zhang and Nivre (2011) (coarse PoS, no labels). I additionally consider three variants of PANPARSER: BEAM (with beam search and global training[14]) and delexicalized versions of both the greedy and beam parsers, DELEX and BEAM-DELEX.

On top of scores averaged over all UD languages, which give insights on the general trend across languages, I also report the results for Ancient Greek and Japanese. Ancient Greek is a free order language which typically yields low scores; it generally corresponds to the lowerbound UAS for each system. Japanese, on the contrary, has a nearly fixed word order and is much easier to parse; it serves as an upperbound. Results in French are also mentioned, as an illustration.

Trainset	10 sentences	500 sentences	Full UD
UDPIPE	22.4‖55.5‖66.6‖**42.5**	53.0‖84.8‖90.2‖**74.7**	66.4‖89.0‖92.7‖**83.2**
PANPARSER	41.4‖69.3‖75.6‖**57.7**	53.8‖83.4‖91.6‖**75.0**	58.0‖87.5‖93.4‖**81.2**
DELEX	41.3‖70.6‖75.1‖**57.2**	50.9‖81.7‖85.7‖**71.3**	51.0‖83.8‖87.7‖**74.3**
MSTPARSER	38.1‖62.7‖68.2‖**52.8**	57.6‖81.2‖86.9‖**75.1**	65.8‖86.7‖90.6‖**83.4**
BEAM	42.4‖69.8‖76.8‖**59.0**	56.0‖84.2‖91.1‖**76.1**	61.5‖88.2‖93.7‖**82.6**
BEAM-DELEX	41.5‖70.6‖77.3‖**59.5**	53.8‖83.5‖87.1‖**73.2**	55.9‖85.6‖88.6‖**76.8**

TABLE 5.2: UAS of the 6 parsers, for various trainset sizes; results for Ancient Greek‖French‖Japanese‖**average** over the 56 treebanks. 'Full UD' trainsets contain between 598 and 68,495 sentences.

Table 5.2 reports the UAS of each system, when trained on 10 sentences, 500 sentences and the full UD trainset.[15] At first glance, the tiny scores appear indeed higher than the intuition would suggest: with only 10 training sentences, all perceptron-based parsers achieve around 70 UAS on French, and BEAM achieves an UAS above 50 for 44 out of the 56 treebanks we consider.[16]

[13]This is my own perceptron-based implementation, described in more details in Chapter 7.

[14]The reported results use the restart strategy and global dynamic oracles that will be described in Chapter 7.

[15]Parsers trained on full data also use the downsized validation sets. Hence, they do not correspond to the results published in the state of the art, but underperform them by typically 1-3 points.

[16]When training on a single sentence, the best parser (BEAM-DELEX) achieves an average UAS of 36.1, although such training is highly unstable, with an average standard deviation around 8.3 UAS.

Despite competitive scores on full datasets, UDPipe is clearly outperformed by all non-neural classifiers in low-resource settings. MSTParser has a slight advantage over PanParser for 500 sentences, but Beam outperforms both of them by a large margin, by combining their virtues (perceptron and global predictions) – MSTParser only catches up on much larger treebanks. Comparison among the same paradigm (the four transition-based parsers with perceptron) reveals that global training is always beneficial, regardless of the data size. Finally, PanParser and Beam outperform their delexicalized counterparts for 500 and more sentences, but for 10 sentences the difference is less significant, even in average.[17] I further explore their differences regarding treebank size, by drawing their learning curves (Figure 5.2).

Overall the worst systems on tiny data are UDPipe and MSTParser, and on large data the delexicalized ones are worst; conversely, Beam outperforms all others on most of the size range. But it appears that up to 7 training sentences, Delex parsers are more accurate in average than PanParser, and similarly Beam-Delex parsers outperform Beam parsers up to 13 sentences.[18] For UDPipe, the critical size required to outperform PanParser is closer to 700 sentences (1,000 for Beam), which disqualifies it for low-resource monolingual parsing.[19] As for MSTParser, it needs 434 sentences to outperform PanParser, which also disqualifies it.

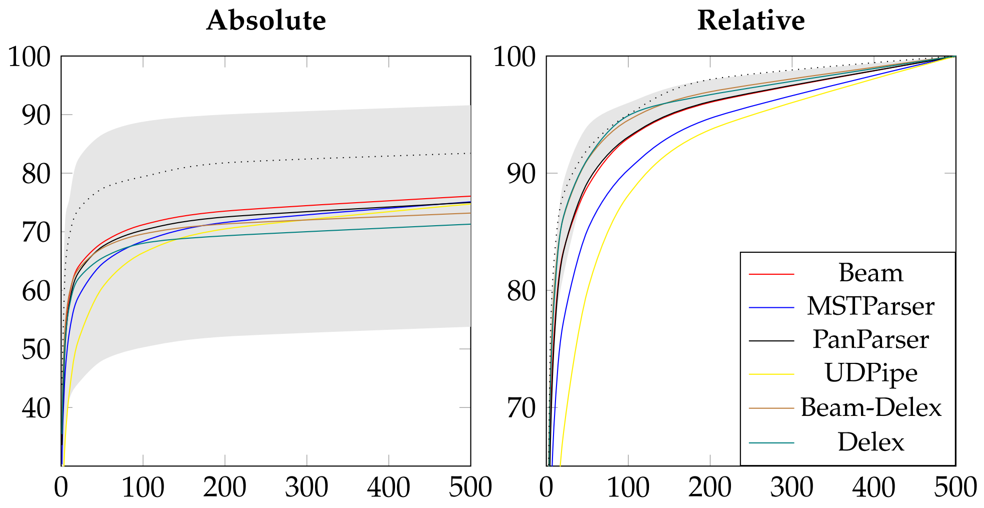


FIGURE 5.2: Learning curves (absolute and relative UAS, averaged across languages; see text for details) for increasing training sizes. The gray areas represent the span between the lowerbound (Ancient Greek) and upperbound (Japanese) curves for PANPARSER. The dotted curves represent French (PANPARSER).

[17]For this data size, PanParser and Delex have typical standard deviations around 2.4 and 3.2 UAS.

[18]One interpretation of this finding is that reducing the feature set prevents overfitting and enforces generalization, which enables to learn general, large-coverage syntactic properties instead of purely lexicalized knowledge. The generalization properties of delexicalized models appear especially valuable in the case of tiny treebanks, where the coverage of lexicalized knowledge is expected to be small.

[19]Three reasons can explain this observation: (a) neural networks tend to have many parameters and are prone to overfitting when hyperparameters cannot be fine-tuned, as is the case in low-resource scenarios; (b) they heavily rely on high-quality word embeddings, which are also not available in this scenario; and (c) neural networks seem to be intrinsically bad at one-shot learning (Santoro et al., 2016; Bertinetto et al., 2016), and in a tiny treebank, most pieces of knowledge typically appear only once.

It is however hard to extrapolate properties for a given language from average behaviour, considering the large score span between Ancient Greek and Japanese (gray area on Figure 5.2, for PANPARSER). Yet, the training differences between parsers are in fact largely characterized by their learning curve shape, which is mostly shared across languages and explains the size range where each system performs best, regardless of the scores obtained on the given language. Overall this curve is always steep, with a quick increase of UAS on the first tens examples, and then a slow convergence; but slight differences remain. Figure 5.2 also pictures the average of *normalized* learning curves, so that for each treebank the curve reaches 100 with 500 sentences, which makes the curve shape more legible.[20] These curves reveal that most of what DELEX and BEAM-DELEX learn is learned through the first few examples, i.e. they generalize well on sparse data but then information saturates, while UDPIPE (and MSTPARSER, to a lesser extent) are slow learners which require a critical mass of data to start extracting more useful information. PANPARSER and BEAM, on the contrary, are able both to extract knowledge quickly, and to go on leveraging any additional data for a long time. For this reason, I choose BEAM as the best monolingual candidate for low-resource parsing in general, and not its delexicalized variant whose high scores are very specific to a narrow size range and which performs poorly elsewhere.

5.2.2 An interpretable quality scale

In PoS tagging, an accuracy of 70% is typically considered low[21] even if it is way above random attachment: indeed, most of the predictions correspond to easy, unambiguous words, and the actual challenge resides in the last few percents (Manning, 2011). So, it is similarly questionable whether a score of 70 UAS is high or low for dependency parsing, which makes absolute UAS values hard to interpret. However, the above study on tiny parsers provides a practical answer to this question.

One way to interpret UAS is indeed by considering the number of training sentences required to achieve a similar score. I call this value the *parsing capacity* of the given UAS: if a parser (for instance, cross-lingual) has a parsing capacity of n, it means that the amount of knowledge it contains is the same as that embedded in a treebank of size n. This notion can also be expressed in terms of annotation cost: Martínez Alonso et al. (2016) evaluate the monetary cost of manual annotations at 3 euros per sentence and layer; let us retain here a sentence cost of 10 euros to include both morphological and syntactic layers and an overhead for small treebanks.[22] The resulting value then states the money that was saved by resorting to

[20]Note how the Ancient Greek–Japanese span is narrowed, indicating that they have similar learning speeds despite important score differences.

[21]For instance, Das and Petrov (2011) doubt on the practical usability of a tagger with 76.1% accuracy.

[22]The overhead is largely reduced by the use of the language-independent UD guidelines and by the reuse of existing annotation platforms, but it remains that annotators' speed increases with treebank size.

e.g. cross-lingual methods instead of manual annotations.

This quality scale is computed with the average UAS over UD languages, using PANPARSER, and reported in Table 5.3. Averaging over this particular set of languages is arbitrary, but in practice it yields results roughly similar to Romanian, whose UAS for size 10 is in fact median among this set of languages.

UAS	30	40	50	60	70	75
Parsing capacity (sentences)	1	2	4	12	77	401
Annotation cost (euros)	10	20	40	120	770	4,010
Romanian trainset size	1	2	3	9	53	410

TABLE 5.3: UAS quality scale: typical number of training sentences needed to reach several UAS levels (measured with BEAM).

From this scale it appears for instance that an UAS of 50 can be considered as 'low', since it has a parsing capacity of only 4 sentences, which is intuitively insufficient to get an accurate description of the language syntax. Similarly, attaching each token deterministically to its left neighbour gives an UAS of 29.8 on average, which has the parsing capacity of a single sentence and confirms that this procedure uses virtually no knowledge; attaching them to one of their neighbours but with a perfect decision function would yield 43.4 UAS,[23] i.e. a parsing capacity of only 2 sentences.

5.2.3 Application: parsing capacity of cross-lingual and unsupervised parsers

The observations made in this section shed a new light on the performance of state-of-the-art cross-lingual and unsupervised parsers.

Indeed, the tiny-resource accuracies reported in Table 5.2 are comparable to other scores reported in recently published papers in those fields. For instance, in unsupervised parsing, Mareček and Straka (2013) achieve 48.7 UAS on average over the CoNLL 2006-2007 treebanks and the average UAS of the rule-based approach of Martínez Alonso et al. (2017) is 57.5 on UD 1.2; in cross-lingual parsing, the multi-source weighted delexicalized transfer method of Rosa and Zabokrtsky (2015) achieves an UAS of 52.5 on HamleDT 2.0 and the projection technique of Tiedemann and Agić (2016) yields 75.43 UAS on UDT 1.0 (COMBG model). These numbers correspond to parsing capacities of, respectively, 4, 9, 5 and 444 sentences.

For fairer comparison on UD 2.0, I reimplement the method of Rosa and Zabokrtsky (2015) with BEAM-DELEX models of all treebanks except those covering the same language (henceforth denoted KL-BEAM). To be fair regarding available target resources, the KL_{cpos^3} metric that weights each source is computed using the whole source treebank but only 10 target sentences. This achieves 66.1 UAS on

[23]Note that this score is in fact the proportion of dependencies of length 1 (on average).

average, which corresponds to a capacity of 32 sentences, or a cost of 320 euros. In other words, if more than 32 target sentences are available (or if the budget available to produce manual annotations is above 320 euros), on average one should prefer monolingual parsing over KL-BEAM transfer.

The study of tiny parsers has shown that even with common annotation schemes, cross-lingual transfer still has a low parsing capacity: it embeds a limited amount of knowledge and does not save much money by avoiding annotation costs. This is less true for methods based on projection, but these already require large data and significant human efforts to find, clean and sentence-align parallel data. Overall, empirical evidence raises serious doubts on the actual effectiveness of many cross-lingual transfer techniques.

5.3 The optimistic approach: opposites attract

In the previous sections, I have reported disappointing results in cross-lingual transfer, and then shown how general they are across languages. Table 5.4a makes up for those bad results: fine-grained evaluation reveals that high scores obtained by tiny parsers vary a lot depending on PoS tags. More striking is the fact that the error reduction when using a cross-lingual parser instead is also unevenly distributed among PoS classes. As a matter of fact, Spreyer et al. (2010) already point out such effects, but between a projected parser and a heuristic baseline built upon tiny data.

	All	ADJ	ADP	ADV	AUX	CCONJ	DET	NOUN	NUM	PART	PRON	PROPN	SCONJ	VERB
KL-BEAM	66.1	72.1	73.4	66.3	72.7	63.7	84.9	59.4	68.3	65.6	72.8	65.0	70.6	55.9
BEAM$_{10}$	59.0	64.5	74.6$_+$	50.6	64.5	55.7	75.2	52.5	52.8	63.5	61.4	47.3	48.6	44.4
BEAM$_{50}$	68.1$_+$	72.9$_+$	82.9$_+$	60.9	75.4$_+$	65.1$_+$	84.2	61.9$_+$	61.9	73.4$_+$	71.6	59.2	65.3	55.6
BEAM$_{100}$	71.2$_+$	75.7$_+$	85.1$_+$	64.9	78.9$_+$	68.6$_+$	86.3$_+$	65.1$_+$	65.5	76.1$_+$	75.4$_+$	62.9	71.2$_+$	59.6$_+$

(A) Scores broken down by child PoS tag.

	All	CORE	NON-CORE	FUN	MWE
KL-BEAM	66.1	70.2	60.1	74.2	45.9
BEAM$_{10}$	59.0	58.3	51.2	71.2	36.9
BEAM$_{50}$	68.1$_+$	68.8	60.6$_+$	79.4$_+$	47.2$_+$
BEAM$_{100}$	71.2$_+$	72.3$_+$	63.9$_+$	81.9$_+$	51.2$_+$

(B) Scores broken down by UD relation subsets.

TABLE 5.4: Average UAS of KL-BEAM and of BEAM parsers trained on 10, 50 and 100 sentences, broken down by fine-grained categories. Averages are computed over all 56 treebanks; for fine-grained scores, only the treebanks with at least 30 occurrences of the category are considered. Scores higher than the cross-lingual one are marked with $_+$.

In light of those results, I hypothesize that transferred models are not intrinsically worse, but simply learn a different kind of knowledge than monolingual systems trained on tiny data. If this hypothesis holds, then both approaches can be combined, and cross-lingual transfer can prove its worth.

The next task I tackle is to characterize the type of knowledge extracted by both types of parsers. As a first attempt, I recompute fine-grained scores along subsets of functional and content-based relations, as advocated by Nivre and Fang (2017), using the same classification. Results, reported in Table 5.4b, reveal that cross-lingual transfer is especially valuable for content-based relations, which are intuitively more consistent across languages (in particular with the UD scheme, whose focus is on those dependencies): Nivre and Fang (2017) define two content-based categories (NON-CORE and CORE), and for those, the transferred parser is at least on par with a parser trained on 50 sentences, and even outperforms it on CORE dependencies. It also appears that despite high accuracies in both cases, functional dependencies (FUN) remain better learned in target data than cross-lingually; so are multi-word expressions (MWE), whose monolingual scores rise way above the cross-lingual ones. This experiment empirically confirms the underlying intuition of delexicalized parsing, whereby it mostly captures the main syntactic features of the sentence, rather than local functional dependencies.

The aforementioned subsets of relations provide a simple criterion which hints at the causes of opposite behaviors and emphasizes on the strong potential of combining both strategies. This criterion remains too coarse, however, to achieve proper fine-grained combination; I consequently turn to a qualitative analysis of the issue.

5.4 Complementarity from a linguistic standpoint

In this section, I tackle with a linguistic perspective the idea that cross-lingual information and tiny monolingual data are above all complementary. In this respect, I consider the syntax of a language not as a whole, but as the combination of various (potentially independent) syntactic phenomena, which all underwent different influences, and do not behave the same with regard to cross-lingual transfer.

5.4.1 A motivating example: Romanian

To apprehend the diversity of syntactic phenomena and similarities, let us come back to the study case of Romanian, with concrete examples drawn from UD 2.0.

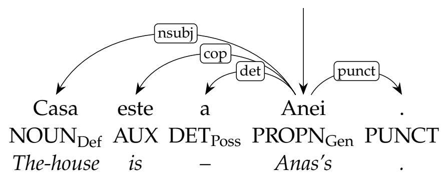
Romanian is an Indo-European language, belonging to the Romance family: as such, it has an SVO word order, like most Indo-European languages. It has inherited a lot of syntactic properties from related languages: for instance, Romanian adjectives (when they are epithets) come mostly *after* the noun they modify, as they do in French and the other Romance languages.

However, its syntax departs from strict inheritance in many ways, which cross-lingual transfer should not overlook. For instance, nominal objects and postnominal possessives have clitic doubling, which is also the case in Spanish but not in French.

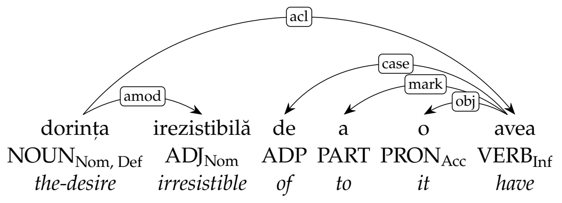
For such sentences Spanish would be a much better source than French, even though both are closely related. Romanian definite determiners also realize as a clitic,[24] as in several other languages (e.g. Arabic) but in no other Romance language, which adds to the difficulty of transferring from Romance sources to Romanian.

On some aspects, Romanian also inherits from another family than its own. It indeed behaves like Slavic languages when deleting articles in prepositional phrases, so that PoS sequences like 'ADP DET NOUN' are as rare as in those languages, and limited to determiners that are not articles. Its preference for subordinate clauses (with a subjunctive introduced by 'să') over infinitives is also inherited from the Slavic family. These similarities are due to specific geographic influences rather than genealogy, and as such their existence could not be inferred from coarse-grain properties of the languages alone, nor from the overall success of cross-lingual transfer: Slavic languages are empirically poor sources for Romanian, and typology-based or trigram-based similarity metrics do not attest of these shared phenomena either.

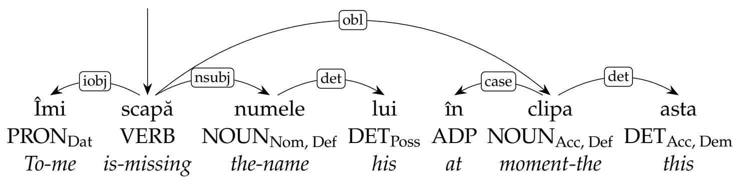
Finally, Dindelegan and Maiden (2013) point out several syntactic patterns which are very specific to Romanian. One such example is the word 'al' (a determiner in UD) that precedes genitives or possessives in predicative position (see Figure 5.3a): this kind of double marking of possession is mostly specific to Romanian and does not exist in any other well-resourced language. Postnominal demonstratives are also constrained to appear just after the modified noun, which is unique as well (see Figure 5.3c). So is the co-occurrence of the preposition 'de' and the infinitive marker (particle 'a') in prepositional infinitive clauses (see Figure 5.3b), while English for instance uses a single particle 'to'. For all these structures, which have direct correspondences in terms of PoS sequences and parsing, cross-lingual processing is hopeless.



(a) Double marking of possession uses both genitive and 'a'.



(b) Preposition 'de' occurs together with the infinitive marker 'a'.

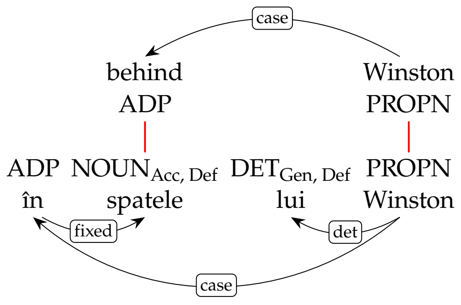


(c) Postnominal demonstrative 'asta' is placed mandatorily just after the noun 'clipa'.

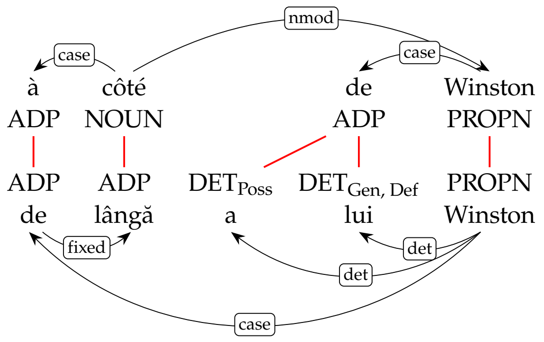
FIGURE 5.3: Syntactic structures that are specific to Romanian.

[24]According to an ungoing debate (Cornilescu, 2016), it may rather be a suffix.

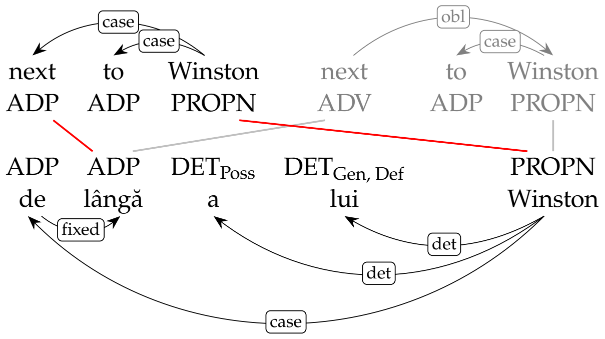
Furthermore, these varying degrees of target specificity also depend on the adopted transfer approach.



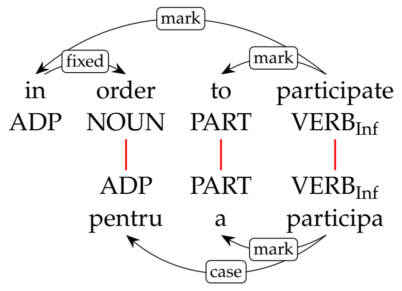
(a) Syntactically inconsistent alignment of a Romanian multi-word expression: preposition 'behind' aligns with noun 'spatele' because their semantic matches, but not their syntax.



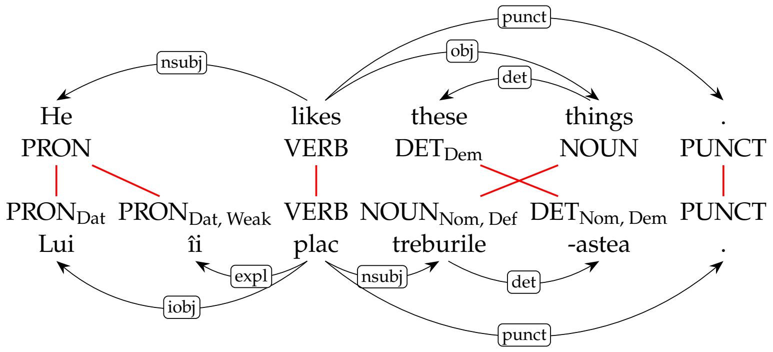
(b) Case where word sequences are semantically similar, but PoS tags and dependencies differ.



(c) 'Next to' expressions have two alternate parses in UD, but none of them align properly with Romanian.



(d) English also has multi-word expressions similar to the Romanian ones.



(e) Sentence pair with perfect semantic, PoS and edge correspondence, but diverging relation labels.

FIGURE 5.4: Romanian syntactic structures which cannot be handled by projection-based methods.

Some Romanian syntactic patterns are indeed doomed to poorly align with other languages, hence failures of projection. This is the case of the numerous multi-word expressions (with `fixed` relations), preferred by Romanian where other languages use prepositions or adverbs. As a result, the corresponding target words often align with the wrong node in tree, likely filtered out because of diverging PoS tags, when they align at all. Figure 5.4a, 5.4b and 5.4c show examples of such expressions, which poorly align with their translations in English or French, hence violating DCA despite straightforward translations. As a result, knowledge regarding these expressions cannot be transferred by any means via parallel data and projection, at least from the examined sources (English, French, but also Spanish, Italian or German for instance) and presumably from many others. Hence, from the viewpoint of parallel data, they are specific to Romanian. Yet, direct transfer could be more successful, because such sequences with `fixed` relations do exist in English (see Figure 5.4d).[25]

Similarly, sometimes despite good alignment and edge correspondence, some sentences cannot get proper relation labels via projection. For instance in Figure 5.4e, semantically the word sequences only differ on clitic doubling ('îi'), but syntactically the subject and object are reversed. As a result, a projected parser would presumably not ignore this information, but instead learn that dative pronouns like 'lui' are indeed `iobj` in general (based on sentence pairs without inversion), but are `nsubj` when modifying verbs like 'plăcea' (including 'scapă' according to Figure 5.3c, and a few others). This knowledge is obviously wrong, and what appears in projected data like an exception to learn results in fact from the incapacity of projection to cover such verbs.[26]

Conversely, for some structures projection is particularly effective, even when direct transfer fails; this is the case of noun-adjective or noun-demonstrative dependencies in English, which are transferred smoothly to Romanian via projection (see Figure 5.4e), despite diverging word order and PoS sequences.

This set of remarks complements the intuition of target specificity with an idea of *querying*: even when the underlying syntactic rule exists on both sides in similar forms (thanks to universals or relatedness), this cross-lingual knowledge may or may not be accessible from the target-side viewpoint (for a given transfer approach).

5.4.2 A bilingual typology of knowledge

Inspired by the study of Romanian syntactic idiosyncrasies, I expect other source-target language pairs to suffer from similar discrepancies (or transfer failures). I thus consider that, in the general case, cross-lingual parsing handles four types of knowledge:[27]

- TARGET-SPECIFIC knowledge, which is absent from cross-lingual resources;

[25] See (Mititelu and Ion, 2005) for other examples of projection failures in Romanian.

[26] Here again, correct knowledge for this sequence of PoS tags and features could be acquired by direct transfer: for instance in Spanish, '*Me queda la casa*' (*to-me remains the house*: I still have the house) behaves the same. Note that this Spanish sentence could not contribute to projected knowledge either, because 'I' is subject in its Romanian translation.

[27] In this context, a 'piece of knowledge' refers to any underlying rule in the syntax of the language. 'Determiners depend on nouns' is one such piece of knowledge, and so is 'sentences rooted by the *plac* verb follow an OVS word order'.

- SOURCE-SPECIFIC knowledge, which does not apply to target input;
- PSEUDO-SHARED knowledge, i.e. what is intuitively similar in both languages but that the considered transfer method fails to query;
- SHARED knowledge, i.e. what is similar in both languages, but also easy to query.

The work presented in the following chapters is mostly based on the following claims:

1. PSEUDO-SHARED knowledge can often be turned into SHARED knowledge with very simple means, as soon as tidbits of linguistic knowledge are available, thereby greatly improving the usability of transfer.
2. TARGET-SPECIFIC knowledge is where tiny parsers have a role to play: SHARED knowledge can already be acquired in source data, and learning it in target data would be redundant, so tiny parsers should focus on extracting TARGET-SPECIFIC knowledge to avoid wasted efforts. However, this does not hold if learning SHARED knowledge facilitates itself the extraction of TARGET-SPECIFIC knowledge.
3. SOURCE-SPECIFIC knowledge brings at worst noise and at best nothing to the task at hand, so it should be left out from source training (again, except if it helps extracting SHARED knowledge).

Understanding the interactions between heterogeneous classes is the subject of Chapter 6; this will provide insights on how SHARED knowledge may improve TARGET-SPECIFIC knowledge extraction, and how SOURCE-SPECIFIC knowledge may improve or degrade SHARED knowledge. Chapter 8 tackles the question of turning PSEUDO-SHARED knowledge into SHARED knowledge; in the following I consequently assume that this conversion has been performed.

5.4.3 Target-oriented relevance

At that point, I have established that in each cross-lingual transfer setting, and with scopes that depend both on the source language and on the type of resource, *some part* of the available syntactic content is 'relevant' for *some part* of the target syntax. By 'relevant' I refer to content whose knowledge deserves to be extracted and transferred, because the corresponding syntactic rules apply similarly to the target. What remains to be done is to associate the conceptual notion of SHARED knowledge with an empirical measure of data *relevance*.

The idea of relevance for target processing exists in various forms in the domain adaptation field. Because of the covariate shift assumption, such works generally associate this intuition with *data similarity* measures (Ben-David et al., 2007; Plank and

Noord, 2011). However, such measures are based only on the words, ignoring the annotations: for instance, a dataset with numerous occurrences of the 'NOUN$\curvearrowright$VERB' pattern would be considered as similar to a target containing many 'NOUN$\curvearrowleft$VERB' sequences. But for my purpose I need to identify this pair of patterns as dissimilar: to be relevant, a piece of data must convey knowledge that is easily queried during target processing, but also leads to correct annotations. This is stronger than mere data similarity. So, it is more appropriate to rather consider *treebank similarity*, i.e. to include the evaluation of annotation similarity.[28] However, to my knowledge, no such metric has been published.

Daume III and Marcu (2006) stand out as an exception, as they argue in favour of taking the annotations into account for e.g. PoS tagger adaptation, based on the idea that the word 'monitor' is likely a verb in news corpora, but a noun in technical documentation. In their work, where they have access to large out-of-domain data and a sample of in-domain data, they improve upon a series of baseline strategies (including the interpolation of the out-of-domain and in-domain models, as well as training a single model on the concatenation of both datasets) by considering each data point as a mixture of general knowledge and domain-specific knowledge. Hence, they end up splitting the available knowledge into three parts, corresponding in fact to the aforementioned distinction between SOURCE-SPECIFIC, TARGET-SPECIFIC and SHARED, and train separate parameters for each. This split is performed using an indicator variable z, stating whether the example belongs to the domain-specific or the general distribution; this variable models in fact relevance. It is clear that they follow the same motivation, but in their work they do not try to actually measure relevance, but resort to trained parameters only. Besides, direct application of this framework to the cross-lingual parsing task faces model restriction issues: Daume III and Marcu (2006) mention that for per-word z (which is intended here), only local classifiers avoid the large incurred computation costs; it is consequently not appropriate for structured prediction like transition-based parsing.

In a nutshell, in order to implement the ideas described above, I need to design either a direct measure of relevance, or a treebank similarity measure. Concrete metrics will be proposed in Chapter 9.

[28]Conversely, it is not desirable either to consider only annotation similarity, regardless of the actual data (e.g. measuring that both languages have the same proportion of left attachments). For instance, data points like 'NOUN$\curvearrowright$CONJ$\curvearrowright$NOUN' are irrelevant to process target inputs like 'VERB$\curvearrowright$NOUN$\curvearrowright$ADJ': this similarity has no linguistic grounding and any accuracy gain related to these patterns would only result from chance. Hence, by 'treebank similarity' I mean in fact the annotation similarity of common data patterns.

5.5 Wrap-up: not so good, but hope prevails

The case study that opens this chapter yields disappointing results: only 12 target annotated sentences suffice to outperform even sophisticated cross-lingual parsers. Yet, this low number may have several straightforward explanations, among others the differences in annotation guidelines, or peculiarities of the Romanian language. However, a more systematic study over a wide and diverse array of languages (with consistent treebanks) only confirms this finding: it establishes that multi-source direct transfer yields the same accuracy, on average, as a monolingual parser trained with 32 target sentences. The monetary benefit of avoiding manual annotations (around 320 euros) is consequently too low to support the use of cross-lingual methods on a systematic basis for low-resourced languages.

As a by-product, the study also reveals that beam perceptron parsers should be preferred in low-resource monolingual scenarios, as they better exploit very small treebanks than their neural, delexicalized, greedy and graph-based counterparts.

However, despite this negative picture of the reality of cross-lingual parsing, a series of fine-grained measures suggests that it can still be valuable when rather *combined* with monolingual parsing, because they do not convey the same knowledge: transfer is particularly useful for content-based relations, while even small target treebanks provide much more information on functional dependencies and lexicalized expressions. This is consistent with the intuition, and I further analyze the topic by drawing a linguistic picture of how language-specific a given piece of syntactic knowledge can be. I summarize these findings by introducing the notion of relevance of a piece of data for the task at hand, which guides subsequent work.

In light of this chapter, I propose to improve cross-lingual transfer by extracting from each resource the task-relevant part of its syntactic content; this would make it possible to combine several cross-lingual and monolingual views, in a selective manner. The next chapter further analyzes state-of-the-art parsers, to investigate how this strategy fits with the inner workings of transition-based parsing.

Chapter 6

Heterogeneity in parsing: monolingual and cross-lingual issues

***

Contents

This chapter dives deeper into the inner workings of parsers, both in cross-lingual and monolingual[1] settings, and completes the picture drawn in the previous chapter, with a finer-grained analysis. Section 6.1 motivates this new perspective,

[1]Many analyses in this chapter focus on monolingual parsing, but this work has a larger scope: analyzing monolingual parsers contributes to improving cross-lingual transfer, in two ways. First, it exhibits general effects which are at work during training – and which hold in both settings but are easier to study in a monolingual condition. Second, it helps in identifying more precisely the shortcomings of tiny monolingual parsers, which is where cross-lingual transfer comes into play.

by showing that a study limited to coarse grain does not faithfully render the actual usefulness of a tiny parser, and would thereby prevent me from getting the most out of low-resourced parsers in concrete scenarios – which remains my primary goal.

This chapter deals with the notion of 'treebank heterogeneity': by 'heterogeneity' I refer to the fact that the dependencies contained in a treebank are typologically very diverse (i.e. involve very different PoS, syntactic roles, prevalence in data, etc.), to the point that annotating tokens in one category or another can present completely different challenges. Studying such typology-related phenomena is of importance for fine-grained evaluation of course, but also when considering the notion of cross-lingual relevance I introduced in the previous chapter: fine-grained typologies of dependencies, based e.g. on PoS tags, uncover marked differences in how shared or universal these dependencies are, but they are also often easy to relate to linguistic properties. For instance, DET$\curvearrowleft$NOUN dependencies are obviously irrelevant for a language without determiners, while case relations will probably be relevant for languages with the same degree of morphological richness. Section 6.2 exhibits a number of such heterogeneous phenomena and their link with linguistic properties; it notably highlights their imbalance – some types of dependencies are by far more frequent that others.

One recurring interrogation in this study, illustrated in Figure 6.1 (hypothesis B), is whether all shared knowledge should really be learned preferably in the source language. For instance, in many languages, determiners attach almost deterministically to the first subsequent noun; learning such dependencies seems so straightforward that it should be possible to learn them directly in the target language, with simple means, while exploiting large cross-lingual resources for that purpose appears computationally expensive, and probably also less reliable. So, what can be

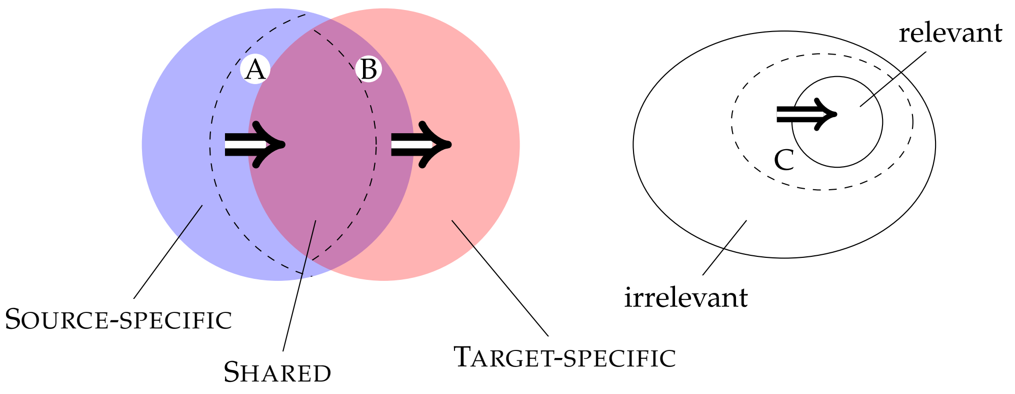


FIGURE 6.1: Typology of syntactic knowledge (shared or language-specific) and of annotated data, depending on their relevance for the task at hand. The three main hypotheses examined in this chapter are (A) whether the cross-lingual model should be trained to extract shared knowledge alone, or also pieces of source-specific knowledge, (B) whether some shared knowledge is better learned on the target side, and (C) whether in general, a parser should train only on the relevant dependencies of a treebank (that is, those whose prediction is expected in the end) or it also needs information about the irrelevant ones.

learned in a tiny treebank and what cannot? Conversely, following hypotheses A and C in Figure 6.1, is the cross-lingual knowledge deemed irrelevant for the task at hand really useless? By attempting to understand how dependencies interact at the feature level during training, I provide elements to answer these questions, and thereby to offer more quantitative insights on the complementarity of cross-lingual and low-resource parsing.

The purpose of this chapter is thus threefold: (a) designing better tools to study heterogeneous patterns, (b) identifying precisely in which ways a tiny treebank can prove useful, and (c) examining interactions between the various types of knowledge to refine the frontier, sketched in the previous chapter, between relevant and irrelevant data.

I fulfill these objectives with a series of independent experiments, focused on various aspects. Section 6.3 tackles issues related to lexicalization and shows how lexicalized and delexicalized parameters interact in subtle ways during training: for instance, as soon as lexicalized parameters are tuned accurately enough, the corresponding delexicalized parameters cease to be updated, an effect which can be viewed as an instance of *explaining away* (Pearl, 1988). Section 6.4 introduces a specific aspect of heterogeneity, *class hardness*, whose measure itself is not straightforward, but which is essential to ascertain what must be learned in which language. The impact of class imbalance is investigated in Section 6.5; it also reports a series of attempts to control this imbalance, whose results further enlighten the mechanisms at work. Hypotheses A, B and C are respectively validated in those three sections.

Each one of these studies has its own worth, and they unveil a few effects whose implications exceed the scope of this work; but once put together, by cherry-picking some of their findings, they also provide an answer to the questions which are my focus. Section 6.6 summarizes those findings, clarifies how they impact the understanding of the inner workings of transfer, and proposes additional guidelines to design a better transfer framework.

The experimental setup remains the same throughout this chapter: when not otherwise specified, measures are performed on UD 2.0, with an ArcEager parser trained with averaged perceptron, greedy search and a dynamic oracle. However, the generality of the findings is assessed by repeated comparisons with neural and graph-based parsers.

6.1 Parser usefulness: heterogeneity matters

Let us come back to the study of tiny parsers, first addressed in the previous chapter, but with a new point of view: downstream tasks. Does cross-lingual parsing produces trees that are useful for downstream tasks, or at least that are more useful than a tiny parser? As will be shown, this property cannot be assessed with UAS only, and it requires fine-grained measures: UAS in general appears indeed subject to much criticism, but my experiments show that the low-resource setting makes its drawbacks even worse.

Literature criticism A converging body of evidence has indeed shown how global attachment scores (UAS and LAS) poorly correlate with task-oriented extrinsic evaluation (Yuret et al., 2010; Popel et al., 2011; Gómez-Rodríguez et al., 2017). Indeed, attachment scores are computed as a percentage over all tokens and thus reward all predicted dependencies equally, even though many of them are actually completely irrelevant to the task at hand (Volokh and Neumann, 2012). There is a consensus in the parsing community that attachment of punctuation tokens is never relevant and as such they are usually skipped when computing scores, but this is by far not the only issue.

For instance, many have mentioned the subject-verb and verb-object dependencies, i.e. the general structure of the sentence, as especially valuable for downstream tasks like polarity classification or machine translation (Ng et al., 2006; Popel et al., 2011; Volokh and Neumann, 2012). Counter-intuitively, tree leaves may also be quite useful in specific cases. In syntax-based machine translation (Yamada and Knight, 2001; Cowan et al., 2006; Gimpel and Smith, 2014; Eriguchi et al., 2016), prediction of target morphology benefits, at least in theory, from knowing source dependencies as local as determiners (for instance, English definite articles are realized with morphology in Romanian) or `case` dependencies, which denote case-marking elements like adpositions (for instance, the English preposition 'at' translates into a locative case in Czech). As for Gómez-Rodríguez et al. (2017), they report good improvements when leveraging local dependencies like `amod` and `neg` (adjectival and negation modifiers). Unfortunately, standard intrinsic evaluation makes it hard to identify the best parser with respect only to the useful dependencies.

To achieve fine-grained parser evaluation, and thus enable research on making the parses more useful, some resort to construction-focused evaluation, narrowing the measures to a subset of complex linguistic phenomena, carefully chosen for their relevance (Rimell et al., 2009; Nivre et al., 2010; Bender et al., 2011). Others simply prune the testset with respect to relations deemed more relevant. Bosco et al. (2014) thus conduct their shared task evaluations on a subset of 18 semantically-loaded relations, a proposal which is close in practice, although not in spirit, to Nivre and Fang (2017)'s recommendation to focus evaluation on content-word dependencies, for the sake of cross-lingual fairness. However, Volokh (2013) reminds that the important relations depend entirely on the downstream task, and that there may be cases where '*the simplest ones are actually the relevant ones*'.

Empirical assessment In the following set of experiments, I focus on differences in parser behaviour related to depth in tree, and oppose the high-level dependencies (close to the root) to low-level ones (close to leaves). This choice is notably motivated by the recent UD guidelines, which have become the *de facto* standard, and in which root attachment matches by design predicate identification. Therefore, identifying the root and its children comes down to finding the predicate and its arguments (*who* did *what* to *whom*), whose high usefulness has been repeatedly noted for downstream tasks like question answering, information extraction or textual entailment (Rimell et al., 2009; Nivre et al., 2010; Volokh and Neumann, 2012).[2] Low-level dependencies are comparatively less useful, but it does not mean either that they are completely useless and should be removed from score computation, as is the case for punctuation marks.

Using the same experimental setup as the tiny parser study in Section 5.2, I measure Spearman's correlations between scores computed on roots only, leaves only, and on the whole test set. Since the number of leaves is large, it comes as no surprise that the variations of overall UAS are mostly representative of its variations on leaves (ρ(overall UAS, leaves UAS) over .95 for all 4 parsers and 3 sizes). But as shown by Table 6.1, the smaller the data gets, the less correlated are learning accurate roots and learning accurate leaves. And thus, for tiny parsers, the overall UAS fails to provide information on root accuracy (with correlations under .300 except for BEAM). In other words, when the goal is to perform a task which benefits more from accurate roots than from accurate leaves, at least in a low-resourced language, the overall UAS *cannot* be used to select the best parser.

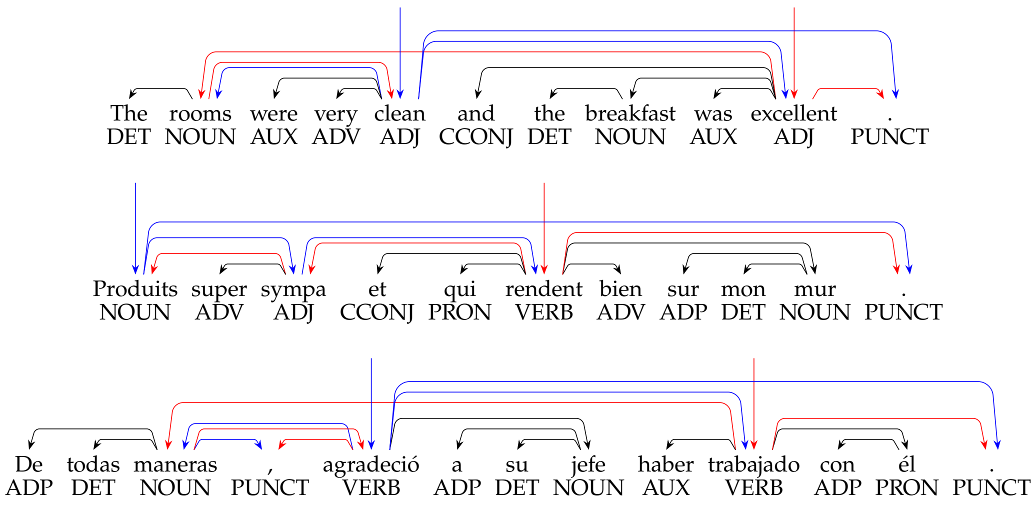
	ρ(root UAS, leaves UAS)			ρ(overall UAS, root UAS)
	10 snt.	500 snt.	Full UD	10 snt.
UDPIPE	.134	.519	.709	.249
PANPARSER	.146	.382	.595	.293
MSTPARSER	.017	.159	.475	.152
BEAM	.360	.577	.716	.477

TABLE 6.1: Spearman's correlation of UAS on roots, leaves and all tokens, for several parsers with various training sizes. Among tiny parsers, accurate predictions appear significantly more correlated for BEAM, which takes the more global decisions.

This is confirmed by a per-sentence analysis. It is indeed technically possible and in fact quite common that for a given sentence, the UAS on high-level dependencies (the root and its children) is at least 30 points lower than the UAS on the whole sentence (for instance, at least 70 UAS on the whole sentence but less than 40 for the top level). This is the case for 15% of sentences in average over the languages (computed with BEAM_{10}, using previous chapter's notations), but this number can reach 42% (Galician, worst case). Figure 6.2 pictures examples of such parse trees. For these sentences, the high UAS is not representative of the usefulness of the parse:

[2]According to Plank et al. (2015)'s experiments, this perspective is also consistent with human judgement, which favours proper attachment of high-level words.

for instance in the first sentence, the output states that the rooms are excellent (which they are not: they are clean) and would thus mislead the downstream system.



The rooms were very clean and the breakfast was excellent .
DET NOUN AUX ADV ADJ CCONJ DET NOUN AUX ADJ PUNCT
Produits super sympa et qui rendent bien sur mon mur .
NOUN ADV ADJ CCONJ PRON VERB ADV ADP DET NOUN PUNCT
De todas maneras , agradeció a su jefe haber trabajado con él .
ADP DET NOUN PUNCT VERB ADP DET NOUN AUX VERB ADP PRON PUNCT

FIGURE 6.2: English, French and Spanish examples of parse trees (predicted by BEAM$_{10}$) whose UAS is over 70 while higher dependencies are mostly incorrect. Correctly predicted dependencies are in black, for others the blue edges represent the reference dependencies and red edges the predicted ones.

Finally, I assess empirically the claim that aggregated metrics make some important parsing challenges 'invisible' (Bender et al., 2011). In the case of BEAM$_{10}$, reference roots (resp. leaves) *with incorrect attachments* represent 3% (resp. 18%) of all dependencies. This means that improvements focused on roots (resp. leaves) are bounded to improve overall UAS by at most 3 (resp. 18) points.[3] Therefore, in practice, even a major improvement on roots, with 100% error reduction, would still be overshadowed by only 20% error reduction on leaves: this renders the efforts towards more useful parsers unrewarding, when only UAS is considered. And this is in fact what happens here: the UAS equivalence of KL-BEAM with BEAM$_{32}$, measured in the previous chapter, hides the fact that they do not provide the same syntactic knowledge. Compared to BEAM$_{10}$, exploiting other languages (KL-BEAM) yields a 33% error reduction on roots, while acquiring more target data (BEAM$_{32}$) yields only a 22% decrease. As far as roots are concerned, the parsing capacity of KL-BEAM is not 32 but 84.

Such considerations affect obviously the accurate evaluation of state-of-the-art parsers and of new proposals, but even the quality of the trained parsers can be impacted. Typically, when cross-validation is used to tune the hyperparameters, it generally involves the same metrics: UAS and LAS computed on the whole dataset. Therefore, even if a given parsing system is able to tune specifically long-tail dependencies, in practice it will lack the opportunity to do so, because the model retained

[3]Roots constitute a much smaller fraction (7%) of evaluated dependencies than leaves (57%), and as such are mechanically doomed to have a smaller impact on micro-averaged measures. But since leaves have much higher baseline accuracies, they could have yielded comparable improvement margins.

in the end will systematically be the one performing best on just the most frequent dependencies (e.g. determiners, whose usefulness is questionable), regardless of the rarer ones.

For all these reasons, I claim that comparison of tiny and cross-lingual parsers should not restrict to the overall picture drawn in the previous chapter, but requires further investigation at a finer grain.

6.2 Empirical evidence of treebank heterogeneity

In order to perform this fine-grained investigation, I move down to the level of classes. In the following, the word 'class' is used for any typological criterion describing a subset of dependencies in a treebank. It can be based on length, depth, edge direction, relation label, PoS tags, empirical values like frequency, etc. The notion extends to the child: in this context, the class of a token is the dependency class of its attachment.

This section illustrates, with empirical measures, what heterogeneity means in practice; it notably reveals to which extent the size of a class can vary depending on its typology, and hints at some consequences that this imbalance can have on monolingual but also on cross-lingual parsing.

To gain quantitative insights on how diverse a treebank can be, I consequently start by examining the properties of several corpora. Figure 6.3 presents a series of measures performed on classes defined by combining two straightforward criteria: child PoS and dependency direction (left), type frequency and tree depth of the child (middle), and head PoS and dependency length (right). Dependency length is defined as the distance *in the sequence* from the dependency child to its head (neighbouring attachments have length 1), and depth is the distance *in the tree* from the dependency child to the ROOT token. As for the classification based on type frequency, it specifies as 'frequent' the words which belong to the 1/8 of types with lowest frequency rank; intermediate categories are similarly defined for 1/4 and 1/2, and the 'rare' category corresponds to all other tokens, i.e. the half of types with largest rank.

The measures of class sizes are reported as heatmaps: each cell pictures the prevalence of the (combined) class, using a logarithmic color scale to better distinguish values in the long tail range. Percentages indicate the numerical value of the neighbour cell. For instance, in the top-left heatmap, the first line states that 8% of English tokens are adjectives, among which most (6%) have their head on the right. Figures also include aggregated values (referenced as 'all'), computed over all classes of the corresponding row or column – including the rarest classes, which for legibility reasons are not represented.

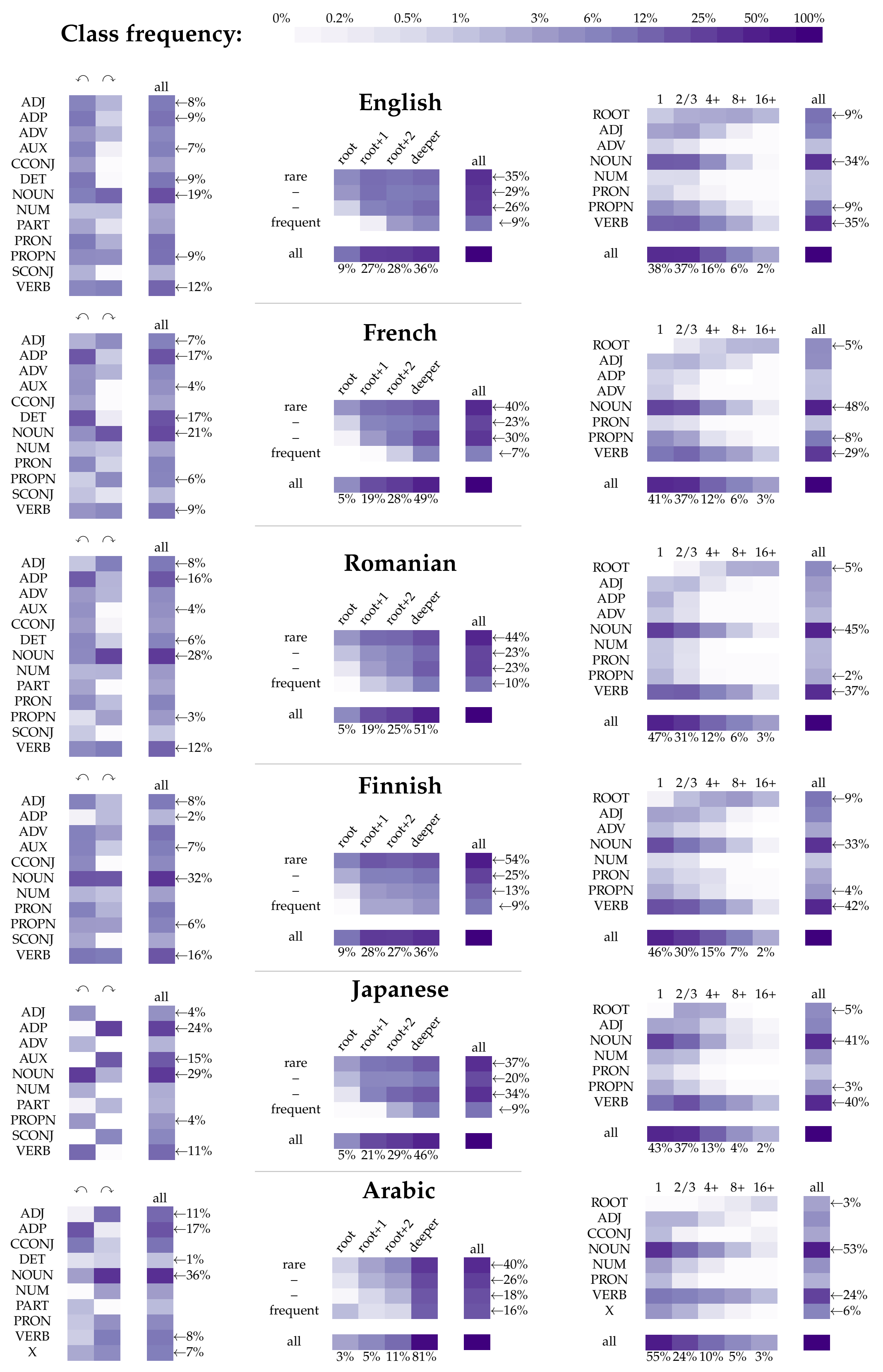


FIGURE 6.3: Measures of heterogeneity along various criteria, in 6 languages. See text for details.

These values are computed for languages from various families (using UD test sets), so that divergences between families are additionally highlighted, while annotation consistency is not in question.

At first glance, Figure 6.3 already confirms several universal properties: the distributions are Zipfian in several respects (with a long tail and the few most frequent words or PoS gathering most of the occurrences) and they denote a marked preference for short dependencies,[4] which is a well-known fact in the linguistic field (Hawkins, 1994; Temperley, 2007).[5]

A number of comments can be done on the results for specific languages, among which several linguistic facts are retrieved.

It appears for instance that English has few adpositions compared to e.g. French; this is consistent with its use of genitive to mark possession, and with its tendency to create compounds. The high number of auxiliaries in Japanese accounts for the complexity of its aspectual and modal system. Arabic and Romanian also seem to underuse proper nouns, but this is not a linguistic property: in Arabic they have in fact been annotated as X, in Romanian it seems due to the treebank genre, containing many legal and scientific documents. Still, regardless of the cause of underuse, this peculiarity will surely affect transfer accuracy in both cases: if a system is transferred from a treebank with virtually no PROPN to a treebank containing many of them, it will have no clue on how to process them.

Regarding attachment direction, most languages present an overall bias, but first this does not prevent many PoS classes from having both, and second the bias is not in the same direction for e.g. English and Arabic. Japanese stands out as an exception, with no dominant direction, and yet the direction is indeed deterministic *with respect to* each PoS – this property likely explains the high accuracies observed previously in Japanese parsing.

Looking at frequency-depth results, the English and Finnish trees appear relatively flat, with fewer high-depth tokens (36%) than other languages (around 50%). Arabic structures on the contrary are particularly deep (81%), but it is unclear if this relates to linguistic properties or another annotation bias; it may only result from

[4]Actually, in most UD treebanks (highly non-projective languages like Ancient Greek and German have sometimes lower rates), between 40% and 50% of dependencies are of length 1, and 60% to 70% are of length 2 at most. This notably raises a strong interrogation on current parsing systems: if most attachments are to a neighbour, or at most a neighbour's neighbour, are complex parsing models worth it? Eisner and Smith (2005) argue the negative, and attempt to simplify their model by bounding dependency length, which yields promising results. Chapters 7 and 9 will pursue on that track.

[5]It is interesting however that these works are based on a dependency formalism that differs from that of UD guidelines, and yet the result holds. For instance, the content-head convention of UD creates a long NOUN-NOUN dependency on the phrase 'NOUN ADP NOUN', while Temperley (2007) mentions this sequence as an example of length minimization in English, precisely because this word order allows to have two short dependencies (NOUN-ADP and ADP-NOUN). Similarly, Japanese is purely left-branching in traditional schemes, which is consistent with Temperley (2007)'s analysis, but UD made it mixed branching.

the high number of CCONJs. The overuse of rare words in Finnish is also obvious (54%), and likely linked to Finnish being agglutinative, with a large vocabulary. In most languages, the combined heatmaps reveal that rare words are evenly distributed across the whole structure, while frequent words become less and less used when going up the tree: the root is in fact never a frequent word.[6] This emphasizes on the need of good generalization to get the high-level dependencies right. This is not true however for Finnish, which has a similar frequency profile at all depths below the root.

The last series of heatmaps presents the distribution of heads and how far they fetch their dependents. For all languages, consistently 80-85% of heads are either nominal or verbal,[7] and in noun phrases the dependencies remain quite local (most of length 1-3, some up to 7), while in verb phrases they are longer in general (up to length 15, or even more). Interestingly, the SOV word order of Japanese is noticeable in the fact that the ROOT token (located at the end) is particularly close to its dependents, i.e. the actual roots.

In a nutshell, even in monolingual settings the treebanks appear very diverse, with strong differences in both typology (for instance in Japanese, all verbs attach on the right, but auxiliaries on the left) and frequency (for instance in French, NOUN heads are 6 times more frequent than PROPN heads), depending on the PoS tags or position in tree; this can already be troublesome for accurate parsing. But bigger issues arise when considering that these discrepancies vary again across languages, which has of course substantial effects on the efficiency of transfer.

6.3 Lexicalization and explaining away

Bearing in mind the motivations and illustrations provided above, I now tackle the first of three studies dedicated to specific aspects of heterogeneity, and start with the topic of lexicalization. In the syntax of a language, there are indeed two kinds of rules underlying the dependency structures: those involving only PoS categories (unlexicalized) and those which depend on the actual words (lexicalized). For instance, in English, the 'VERB NOUN NOUN' sequence parses generally as an `obj` dependency between a verb and a `compound` noun (e.g. 'induce thyroid cancers'); this piece of knowledge is unlexicalized. But when the verb is *give*, *tell*, *show*, etc., it parses as a verb with two dependents, one `iobj` and one `obj` (e.g. 'give patients hope'): this is lexicalized knowledge. At the same time, standard feature templates consist in lexicalized features (referring explicitly to the words, e.g. 's_0=give') and

[6]This is the case for Arabic though, but its frequent roots correspond in fact to two specific words (meaning 'Sir' and 'and'), which appear in so many sentences that it biases the corpus.

[7]Compared to the others, Arabic overuses noun phrases at the expense of verb phrases, but this is again due to a treebank bias.

unlexicalized features (e.g. 's_0=VERB'). Intuitively, and that is the motivation underlying such templates, the parser should learn and predict lexicalized dependencies based on their lexicalized features, and similarly for the unlexicalized ones; I show however in the following that, for lack of hard constraints on that matter, in practice the intuition does not hold, and parsers end up using lexicalized parameters to learn unlexicalized knowledge, or approximating lexicalized knowledge with unlexicalized parameters. Hence, in my analysis, I study in fact lexicalization along two dimensions: whether a feature (and the corresponding parameter, or weight) is lexicalized, and whether a given piece of syntactic knowledge (like identification of nominal subjects, or epithet attachment) needs lexical information to be correctly acquired.

The topic of lexicalization is obviously related with cross-lingual transfer, since lexicalized knowledge cannot be acquired directly in another language (with pure direct transfer), for lack of common wordforms. But it is also tightly linked with my concerns in a second regard, the ability to exploit tiny data. Coming back to the 'VERB NOUN NOUN' example, a tiny parser is presumably able to learn the interpretation as compound noun, but would fail to handle the case of 'give': as an exception to the PoS-based underlying rule, the natural way to encode it in feature weights is by enumerating the handful of verbs triggering that structure – but this is only possible for those encountered in training data, i.e. probably none in tiny data. Acquiring a given piece of lexicalized knowledge (like that exception) thus appears to require large amounts of data (to cover all word types); hence, the knowledge contained in a tiny treebank seems to correspond mostly to unlexicalized knowledge, for which generalization based on a few examples is doable.

With that motivation in mind, this section consequently studies interactions between both types of parameters and thereby reveals that enabling lexicalization *at training time* affects how unlexicalized parameters are learned: lexicalized parameters prevent generalization by offering a more direct way to learn a dependency; but at the same time, by taking care of lexicalized exceptions, they let the model learn more general rules, which has then different implications in monolingual and cross-lingual parsing. Since lexicalized exceptions are language-specific, this finding supports hypothesis A of Figure 6.1: SOURCE-SPECIFIC information can prove useful for cross-lingual transfer.

Exhaustive experiments on all UD languages are out of the scope of this specific study and I focus on English and French, but for the sake of generality, I still start with a case study in a different experimental setup (English-Finnish transfer) to sketch some interesting properties, before investigating them (and confirming them) in a more systematic manner on English-French. The reason underlying this choice of languages is that English and Finnish have relatively similar word orders (SVO), with the exception of nominal adpositions (mostly prepositions in English, postpositions in Finnish): their main difference is that Finnish is an agglutinative language with much richer morphology than English, and consequently has a much larger vocabulary (and many more lexicalized parameters). English-French transfer is a slightly different paradigm: the pair is more similar than English-Finnish, and French is a synthetic language with a much smaller vocabulary than Finnish (but still larger than English).[8] So, contrasting the study with Finnish also enables us to verify that the effects I uncover hold even when the number of lexicalized parameters explodes.

6.3.1 Hands-on measures

I first illustrate the impact of lexicalization on English and Finnish, by evaluating various lexicalized and delexicalized parsers at a fine-grained level (i.e. with UAS computed per class) and highlighting their differences in accuracy.

Figure 6.4 compares three parsers: LEX (a standard lexicalized parser), DELEX (a parser trained on delexicalized data) and DELEX(LEX) (obtained by deleting all lexicalized parameters from a LEX parser, or equivalently by applying LEX on delexicalized test data). Figures 6.4a and 6.4b represent UAS measures in monolingual and cross-lingual settings, while Figures 6.4c-6.4f picture UAS *differences* in the monolingual settings, and Figures 6.4g-6.4h in the cross-lingual setting.

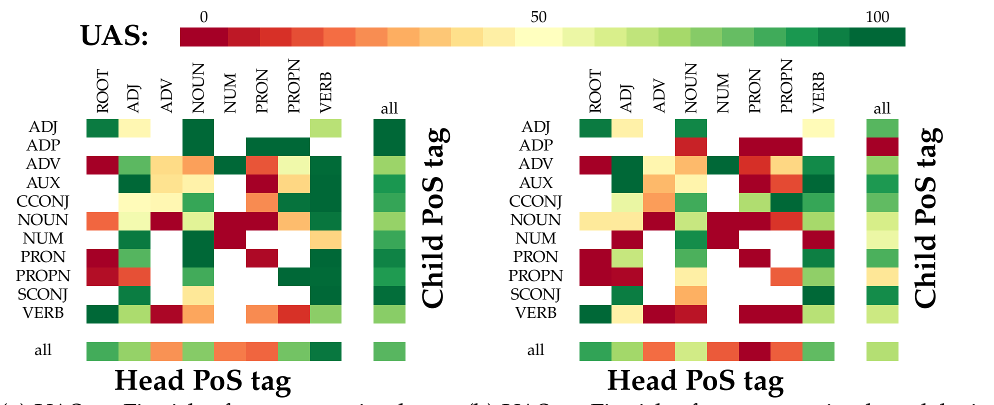
Fine-grained scores are computed by PoS pairs, child and head. For instance, the cell in row ADJ and column NOUN represents the proportion of correct attachments for adjectives whose gold head is a noun; in Finnish its value is 92.4 UAS (Figure 6.4a) and it undergoes a decrease of 1.3 UAS when lexicalized features are accessible during training (Figure 6.4f).[9]

Monolingual analysis My first series of comments concerns the monolingual measures. On the one hand, comparison of LEX with DELEX(LEX) (Figures 6.4c and 6.4e) indicates how much information is embedded in the lexicalized parameters. As it turns out, the prediction of some dependencies by LEX relies a lot on lexicalized features: attachment of ADVs, NOUNs, NUMs (and of Finnish VERBs) is severely hindered when the information contained in lexicalized parameters is lost (which is indicated by dark blue cells). This is not the case however for finding the root and the dependents of VERBs (light blue cells), whose knowledge appears encoded mostly in unlexicalized parameters.

On the other hand, comparison of DELEX(LEX) and DELEX (Figures 6.4d and 6.4f) reveals that lexicalized features can also have a significant effect on how

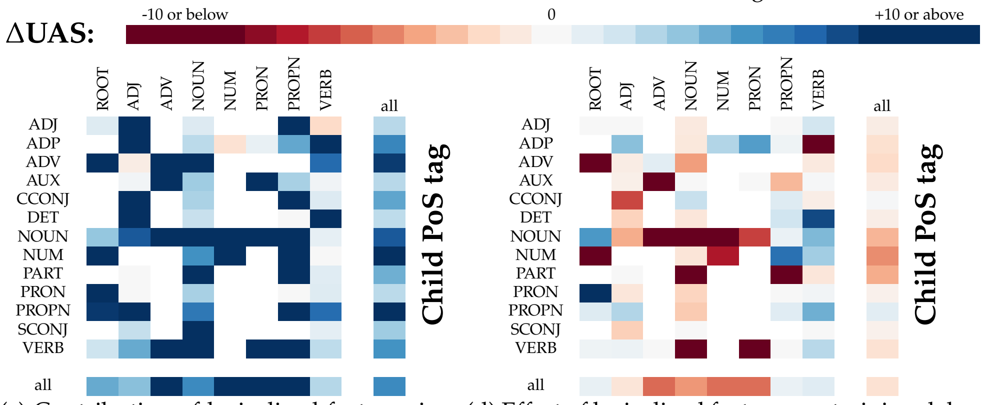
[8]In UD 2.0, measures of the type-token ratio give 0.10 in English, 0.12 in French, 0.30 in Finnish.

[9]Only cells with at least 20 occurrences are represented, but overall scores ('all' values) are based on the whole corpus (except for punctuation tokens, as is customary).



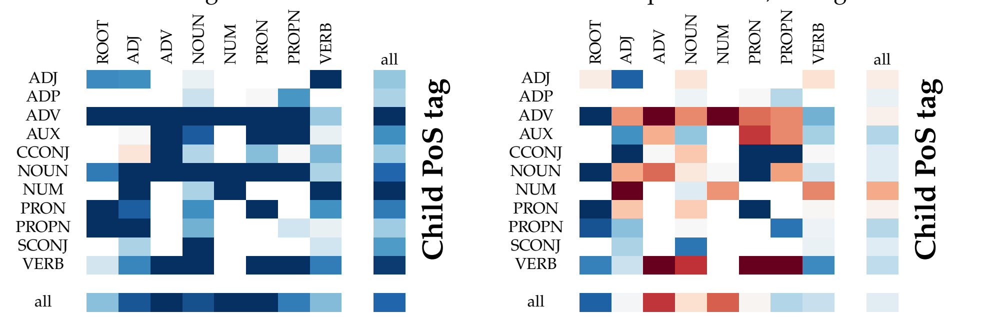
(a) UAS on Finnish of a parser trained on delexicalized Finnish.

(b) UAS on Finnish of a parser trained on delexicalized English.



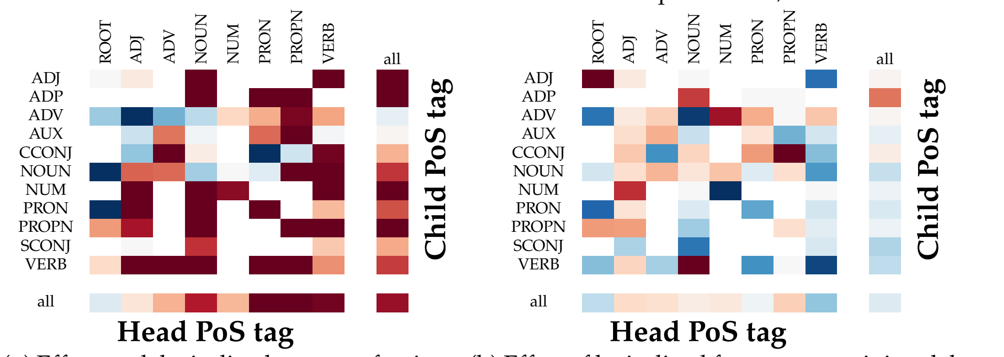
(c) Contribution of lexicalized features in English.

(d) Effect of lexicalized features on training delexicalized parameters, in English.



(e) Contribution of lexicalized features in Finnish.

(f) Effect of lexicalized features on training delexicalized parameters, in Finnish.



(g) Effect on delexicalized parsers of training on English instead of Finnish.

(h) Effect of lexicalized features on training delexicalized parameters, for English-Finnish transfer.

FIGURE 6.4: Fine-grained comparison of LEX, DELEX(LEX) and DELEX parsers in monolingual and cross-lingual settings. Heatmaps are drawn along linear color scales. See text for details.

the parser is trained, even if not used at test time: having *access* to lexicalized features during training improves the way that the delexicalized parser learns some classes (blue cells), but degrades others (red cells). In both languages, the predictions which benefit the most from this effect pertain to roots and to dependents of PROPNs and VERBs. But overall the effect depends a lot on the language. In Finnish it is particularly beneficial for roots, but also for several classes across the board, and hence it is beneficial *overall*: DELEX(LEX) is more accurate than DELEX. For English however, the impact on attachments of nouns and verbs is so negative that the overall impact is negative. An explanation to this surprising interaction will be proposed in §6.3.2.

Impact on transfer Now examining the cross-lingual results, a few more comments can be made. For clarity, I denote X-LEX the English LEX model applied on Finnish data (and similarly for X-DELEX and DELEX(X-LEX)). Figure 6.4h then compares DELEX(X-LEX) with X-DELEX (which shows the impact of lexicalized features on delexicalized transfer), while Figure 6.4g illustrates the cost of transfer, i.e. the UAS differences between X-DELEX (Figure 6.4b) and DELEX (Figure 6.4a).

Visual comparison reveals that the cells with high UAS (green) in Figure 6.4b are those with small UAS differences (light) in both Figures 6.4c and 6.4e. In other words, transfer accuracies result from two concurrent effects: on one hand, the English dependencies which are very lexicalized contribute less to cross-lingual transfer, because for them not much unlexicalized knowledge can be extracted, and transferred (see ADJ-ADJ or PROPN-PROPN attachments); on the other hand, the Finnish dependencies which are very lexicalized are predicted with lower accuracy by cross-lingual transfer, because they require information that is very specific to Finnish and cannot be acquired cross-lingually (see ADJ-VERB or NUM-VERB attachments).[10]

However, the fact that few information is transferred regarding lexicalized dependencies does not mean that lexicalized features are useless for transfer. Considering that the exact same models have been evaluated in Figures 6.4d and 6.4h, it appears indeed that depending on the test conditions only (and thus on the use case, monolingual or cross-lingual), access to those features can overall be either detrimental (monolingual case) or beneficial (cross-lingual case). At fine grain, this difference is particularly visible for NOUN and VERB attachments, as well as attachments to NOUNs. Hence, lexicalized features can induce information losses from the monolingual viewpoint, but simultaneously improve generalization or cross-lingual consistency.

My last remark concerns the dependencies involving VERBs, which are severely impacted by transfer itself (red in Figure 6.4g), and at the same time significantly

[10]This is of course not the case for ADPs, whose dependencies are rather unlexicalized in both cases, but which behave differently in both languages, hence low transfer accuracy.

benefit from generalization improvements (blue in Figure 6.4h). This suggests another approach to explain the bad accuracies obtained by vanilla delexicalized transfer (apart of course from actual divergences like English-Finnish ADPs): the transferred models may simply lack some additional help (e.g. by adapting feature representations) to achieve the particularly high degree of generalization required in cross-lingual settings.

This case study has ascertained that some dependencies need lexicalized information to be predicted, while others use purely unlexicalized parameters, and it confirms the intuition that the most transferable ones are the latter: they contribute more to training and have better cross-lingual accuracies. But it has also shown that lexicalized features have an effect on how unlexicalized parameters are trained, which is either positive or negative depending on the class and the language, but is in general more beneficial to cross-lingual transfer.

6.3.2 Generalization and explaining away effects

I now examine more carefully the interactions of lexicalized and unlexicalized parameters, and thereby confirm the above findings on another language pair, by transferring from English to French, and with more systematic measures.

Having shown that both types of parameters convey information on overlapping pieces of knowledge, I uncover an explaining away effect[11] between features that hinders proper generalization. Yet, I conclude by exhibiting positive effects of explaining away: because their use also simplifies the boundaries of dependency classes, lexicalized features can actually prove useful to train delexicalized parsers.

Redundancy of lexicalized parameters I first examine whether lexicalized and unlexicalized parameters can capture the same knowledge, which would make them prone to explaining away effects. I achieve this by changing the initialization value (usually, 0) of some parameters, before performing a lexicalized training as usual. In a first strategy, denoted 'Delex + Lex', a pre-trained Delex model is given, and all unlexicalized parameters are initialized to their final value in Delex. This is equivalent to training first the unlexicalized parameters, and then tuning the whole set by adding lexicalized representations; intuitively the initial values already contain most knowledge from the dataset, enough to build already accurate parses, and there should not be much tuning, only to cover the cases left unresolved by delexicalized

[11]In the sense of Pearl (1988), the explaining away effect concerns Bayesian networks specifically. It states that when two independent events (A and B) can cause the same event C, observing C can make them *conditionally* dependent: if A is observed as well, it is already a good explanation for C, so that B becomes less likely (it has been *explained away* by A). This work borrows this term but uses it in a broader sense, in a context of error-based training: if both features A and B can be used to predict dependency C, having already tuned the weight of A *explains away* any possible update on B, because A is already a good predictor for C.

	UAS	Norm		Dist. to Lex		Dist. to Delex	Significant features	
		delex.	lex.	delex.	lex.	delex.	delex.	lex.
Lex	88.31	1,054	3,193	0	0	1,118	5,034	34,148
Delex	85.44	1,517	0	1,118	3,193	0	8,122	0
Delex(Lex)	83.73	1,054	0	0	3,193	1,118	5,034	0
X-Delex	69.68	1,403	0	1,460	3,193	1,729	7,558	0
Delex(X-Lex)	70.10	1,094	0	1,206	3,193	1,557	5,537	0
Delex + Lex	88.50	1,131	3,572	502	1,863	1,129	5,824	50,804
Delex(Lex) + Lex	88.73	1,354	2,490	491	1,824	1,126	8,202	14,640
X-Delex + Lex	88.82	1,545	3,006	1,160	1,753	1,444	9,099	27,511
Delex(X-Lex) + Lex	88.84	1,315	2,898	884	1,752	1,289	7,329	24,178

TABLE 6.2: Accuracy, model norm and model size of lexicalized and delexicalized parsers, with training modified by several initialization strategies. Parsers are trained on French, English or both, and evaluated on French.

models. In the 'Delex(Lex) + Lex' strategy, again only the unlexicalized parameters are initialized, but now to the final value of a pre-trained Lex model. In that case the initialization point corresponds to a truncated model, much knowledge has been lost compared to the convergence point, so much tuning is expected to restore the value of the discarded lexicalized features. But this experiment mainly assesses how timing influences training, i.e. whether reaching earlier the convergence point of a given parameter affects the updates on other parameters. To explore potential applications to transfer, similar initializations are performed using cross-lingual models (strategies 'X-Delex + Lex' and 'Delex(X-Lex) + Lex').

Table 6.2 reports several measures for each model: UAS, Euclidean norm of both subvectors (unlexicalized and lexicalized weights) and Euclidean distance to the baseline Lex and Delex models. It also contains the number of significant features in both categories: I consider a feature significant when it has a weight above 5 (or below -5) for at least one class, in which case it strongly biases the chosen action.

At first glance, results seem incompatible with the aforementioned intuitions: 'Delex + Lex' trains indeed a lot of significant features, even more than 'Lex', and its lexicalized norm is also higher. However, this is easily explained by looking at delexicalized distances between parsers: indeed, on one hand Delex(Lex) is far from Delex (with a larger distance than its own norm), and on another hand all lexicalized models with partial initialization (including those initialized with Delex) are closer to Lex than to Delex. It ensues not only that the delexicalized subvectors of lexicalized and delexicalized models converge to very different points, but also that this is a property of lexicalized training itself, which is not altered by any initial bias for a given region. As a result, to reach the lexicalized convergence point, 'Delex + Lex' has more training to do: first forgetting its initial bias, then updating towards this point. It is thus not comparable to other models, and ignored in the following.

Other results show that when unlexicalized parameters are tuned first, the lexicalized features are actually much less exploited. Their norm is smaller, and their weights are condensed on a much smaller feature set, which is compensated by the unlexicalized ones. Less than half of lexicalized features significant in Lex are reacquired by 'Delex(Lex) + Lex', meaning that the others have indeed become redundant, and were only trained because the corresponding knowledge was not yet available in unlexicalized parameters: they have been explained away. Hence, unlexicalized and lexicalized contents overlap, but the former uses much less features because each one combines the weights of a large lexicalized inventory. The same effects hold with cross-lingual initialization, to a lesser extent (a bit more when using X-Lex); but it is not clear how the knowledge embedded in X-Lex and X-Delex relates to the monolingual weights.

Finally, it is worth mentioning that all initialization strategies improve the UAS over the Lex baseline, by a small but consistent margin (+0.2 to +0.5). Notably, the best models are those based on cross-lingual initialization, which benefit from additional knowledge (English syntax) compared to purely monolingual models. Fine-grained evaluation of 'Delex(Lex) + Lex' reveals that the gains are mostly due to attachments involving only nouns and verbs. The learning curves of the corresponding classes (Figure 6.5) confirm that this is due to good initialization: it starts with a bonus accuracy on those classes, but it does not stagnate while other parameters are tuned, and it stays ahead of the baseline all along, performing better updates, which are not accessible to the models without prior knowledge.

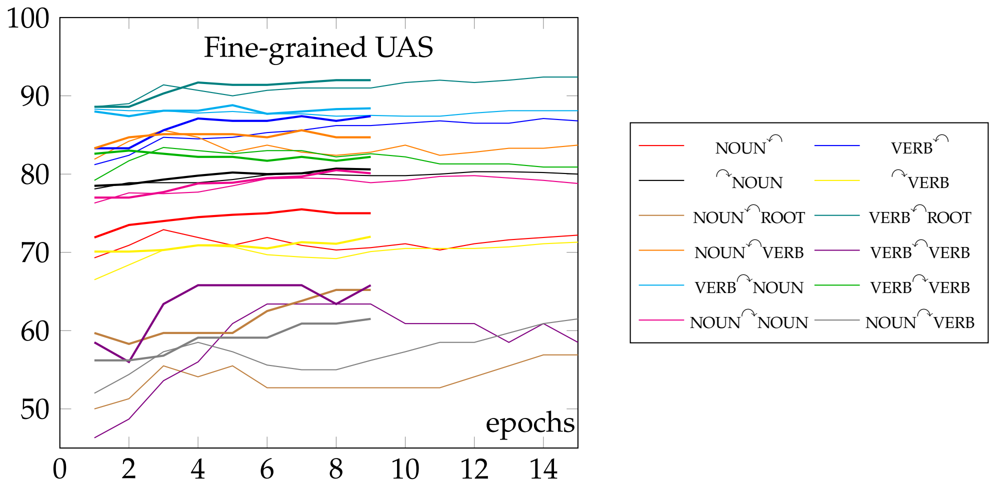


FIGURE 6.5: Learning curves of Lex and the 'Delex(Lex) + Lex' initialization strategy (respectively thin and bold curves), at fine-grained level.

Hence, lexicalized and unlexicalized parameters cover (partly) the same knowledge, and timing matters: letting one parameter or the other learn that knowledge first has an impact on how the other parameters are trained, and can even give access to better updates. However, parameters from both categories are not on an equal footing: unlexicalized parameters have good generalization properties, while purely lexicalized decisions have low coverage and make learning more tedious. Since it controls the difficulty to learn, timing appears to have high stakes.

Detrimental effects on generalization Further exploring the ideas sketched in §6.3.1, I now compare more carefully the accuracies of Delex and Delex(Lex) in Table 6.2. As it turns out, for the same inventory at test time, giving access to more features at training time actually lowers UAS (-1.7): when the parser has access to lexicalized features, it learns to rely on them, instead of generalizing what can be generalized with unlexicalized parameters. In that sense, lexicalized features explain away unlexicalized knowledge.

According to previous measures on Finnish and English, this seems less true for the morphologically richest languages though (which Finnish is, compared to French and English), presumably because the vocabulary is too large to even consider fully-lexicalized knowledge: each time a new word is encountered during training, purely unlexicalized knowledge is necessarily solicited,[12] which lets it be fine-tuned in the long run.

Table 6.3 shows fine-grained scores for the most impacted classes. For multi-word expressions, only a small amount of knowledge can be generalized on PoS tags (38.2 on `fixed` for Delex, while Lex achieves 51.1), but even that is hindered by lexicalized features, which instead make the parser focus on enumerating the known multi-word expressions. The same happens to other closed classes (SCONJ, `mark`). For open classes like nouns or verbs, there is no obvious reason to prefer lexicalized knowledge, but as they are particularly ambiguous, learning them takes time, which gives lexicalized features an opportunity to take precedence.

Coming back to earlier measures performed on English (Figure 6.4d), this now explains the large penalty on ADP attachment to VERB: as there is roughly a dozen of verbal postpositions (*up*, *out*, *to*, *of*, *off*, *in*, *on*, *down*, *with*, *at* and a few others), with lexicalized training it is tempting to rely entirely on these few features for better handling, which prevents generalization.

Benefits for class boundaries However, a last comparison reveals that explaining away due to lexicalized features can also be beneficial (as already measured on English-Finnish): Delex(X-Lex) is 0.4 UAS more accurate than X-Delex overall, but the differences are more striking at fine-grained level, as shown in Table 6.4. Performance on multi-word expressions is degraded for the same reasons as above, and nominal modifiers presumably suffer from the fact that English uses lexicalized parameters to distinguish both genitive placements (which is irrelevant in French). But among all other classes, many present substantial improvements.

[12]This is at least the case with the standard feature templates, containing only wordforms and unlexicalized features: for instance, unknown Finnish words could be handled on the basis of known morphemes (which would be lexicalized), if such features were used in the templates.

	Child PoS						
	ADV	NOUN	PROPN	VERB	SCONJ	Others	
Delex	84.0	73.8	81.1	69.9	86.4	92.8	
Delex(Lex)	79.6	70.2	76.2	66.7	82.6	92.6	
Δ UAS	-5.5	-3.6	-4.9	-3.2	-3.8	-0.2	
	CORE	NON-CORE			MWE	FUN	
	nsubj	acl	advmod	nmod	fixed	mark	Others
Delex	89.0	60.0	85.2	81.9	38.2	92.2	87.7
Delex(Lex)	83.3	51.8	80.6	70.9	31.5	87.4	88.5
Δ UAS	-5.7	-8.2	-4.6	-11.0	-6.7	-4.8	+0.8

TABLE 6.3: Fine-grained UAS of Delex and Delex(Lex), for the most impacted classes.

For the concerned dependencies, PoS-based generalization is intuitively appropriate, but not done in practice because they have many exceptions, and delexicalized models try in vain to handle them. But when access to lexicalized features is granted, the exceptions can be encoded in lexicalized parameters, which lets the unlexicalized ones focus on the more general rule, i.e. it enforces proper generalization. Thus, even if they are discarded at last, the lexicalized features themselves simplify class boundaries, making the extracted unlexicalized knowledge more general, and consequently more transferable.

	Head PoS			Child PoS							
	NOUN	VERB	Others	DET	ADV	ADP	SCONJ	PRON	NOUN	PROPN	Others
X-Delex	74.5	74.0	55.7	93.5	68.1	81.9	51.5	79.6	60.8	43.4	60.0
Delex(X-Lex)	70.1	79.4	56.2	94.7	71.8	84.2	56.1	86.5	54.1	36.6	60.3
Δ UAS	-4.4	+5.4	+0.5	+1.2	+3.7	+2.3	+4.6	+6.9	-6.7	-6.8	+0.3
	CORE			NON-CORE				MWE	FUN		
	xcomp	nsubj	obj	advmod	advcl	obl	nmod	flat	mark	Others	
X-Delex	82.2	69.7	88.4	70.7	45.7	62.5	67.7	28.6	57.6	71.7	
Delex(X-Lex)	93.3	76.4	89.9	75.1	51.1	69.8	44.7	16.0	68.8	72.4	
Δ UAS	+11.1	+6.7	+5.5	+4.4	+5.4	+7.3	-23.0	-12.6	+11.2	+0.7	

TABLE 6.4: Fine-grained UAS of X-Delex and Delex(X-Lex), for the most impacted classes.

To conclude, this study has provided evidence of explaining away effects between lexicalized and unlexicalized features in parsing, and shown that they can be either detrimental or beneficial depending on the context.[13] In monolingual settings they let the model wrongly rely on features that will not be used at test time, while in cross-lingual parsing they help simpler and thus more general rules to emerge; of course this enforced generalization can also be helpful in some monolingual cases, and vice versa.

This series of experiments has only scratched the surface of the implications of feature redundancy for training parsers. First, the measures have been performed

[13]In contrast, Bartenlian et al. (2016) also study the impact of lexicalization on out-of-domain PoS tagging, but they exhibit only detrimental effects.

with a perceptron-based parser, whose update criterion is error-based, and generalizing the results to other training procedures would require further investigation.[14] Second, it seems reasonable to expect that explaining away effects concern all kinds of redundant pairs of features, not only the lexicalized and unlexicalized ones.[15] An extensive exploration of the topic is out of the scope of this work, and I do not pursue on this track; but I retain from this study that there are concrete cases in which source-specific features help the extraction of cross-lingual knowledge, by taking care of source-specific exceptions. This validates hypothesis A of Figure 6.1.

6.4 Class hardness

As a second study, this section investigates the intuition that some classes are harder to learn that others. Indeed, despite the picture opposing target-specific and shared knowledge (drawn in Section 5.4), sometimes shared knowledge may still be much more easily (and directly) learned in target data than by cross-lingual means. In that case, it would be better to consider this piece of knowledge as target-specific, and let it be learned in tiny target data: this proposal corresponds to hypothesis B in Figure 6.1. The focus here is then to characterize difficult classes: based on detailed fine-grained analysis, I introduce concrete measures of class hardness, with a focus on complexity. Cross-lingual and tiny parsers are then reevaluated in accordance with this criterion, which confirms that to learn simple classes, tiny target data should be preferred. Additional results and analyses can be found in the publication associated with this study (Aufrant et al., 2018b).

However, the idea of 'difficulty' corresponds in fact to many different phenomena. Classes with high accuracies are typically considered easy. But difficulty also arises from having a large number of exceptions, in which case the easy classes are rather those for which the systems can generalize well. The previous section has notably mentioned the case of purely lexicalized knowledge, whose acquisition requires large amounts of data: is the task more difficult just because more data is needed? Besides, some sentences are truly ambiguous, so that there is no way to annotate them with full confidence; is this also a kind of difficulty? This multiplicity of meanings has been formalized by Ho et al. (2006), according to whom the difficulty

[14]For instance, when using an objective function that includes L_2 regularization, training is supposed to balance the weights between two equivalent features, regardless of timing. In such case, knowing the lexicalized features does not prevent from tuning the unlexicalized parameters; but on the other hand, some weight mass is systematically held by the lexicalized part of the model, so that it inevitably relies on lexicalized features. Therefore, intuitively, the aforementioned effects on parsing accuracy hold, but they cannot be exhibited using the same timing tricks.

[15]For instance, in English, learning to attach a DET to the first subsequent NOUN (that is, based on the feature that joins both PoS tags) is roughly equivalent to learning to predict the DET-NOUN dependency when the DET-NOUN distance is 1 (which is indicated by a specific feature in standard templates), and also when their distance is 2 (to cover the DET-ADJ-NOUN case); depending on timing, one way may overcome the other. Yet, the former is a better generalization than the latter, rather language-specific; similar effects as for lexicalization can thus ensue.

of a given task can arise from three factors: complex decision boundaries, information scarcity and class ambiguity.[16] Their work belongs in fact to a large body of research on the causes of prediction errors,[17] which has already proposed a number of concrete measures to assess the difficulty of a task, along those criteria.[18]

6.4.1 Existing metrics to assess task difficulty

Learnability Schwartz et al. (2012) have already explored ways to quantify the difficulty to learn *from a given treebank*. Although their purpose (improving annotation schemes) differs from mine, I believe that their work can be used to measure, at the same time, both the complexity of boundaries and the class ambiguities.[19] Indeed, among other metrics, they define the accuracy-learnability (henceforth, learnability) of a treebank, which is simply the accuracy of a parser trained on a large dataset (thereby alleviating the information scarcity effect), and represents the maximal amount of knowledge that can be extracted from annotated data. They measure it at the level of the treebank, but I propose to also compute that metric (which is in fact an UAS) over a given class only, because then learnability indicates *how separable*[20] that class is.

Information scarcity Li and Abu-Mostafa (2006), on the other hand, focus on the third cause of errors, information scarcity, and aim at quantifying the knowledge embedded in a given piece of data. They propose to measure directly its Kolmogorov complexity, i.e. its minimum description length, but show that this measure is incomputable in general.[21]

[16]They consider as ambiguous a class whose boundaries cannot be ascertained based on the available features. The straightforward solution to this issue is to use more features. But sometimes it happens that the very same input results in different annotations (either because of annotation errors, or because of a true linguistic ambiguity), in which case improving the features does not help.

[17]An early work is for instance that of Domingos (2000), who decomposes them into three factors: bias, variance and noise. With a more practical approach, Ho and Basu (2000) investigate a series of metrics for task difficulty, like feature overlaps, separability and topology of the dataset.

[18]Actually, this interest has mostly focused on standard classification tasks, which is not the frame of my work: I am rather interested in understanding erroneous attachments than in the prediction of incorrect actions. Nevertheless, one way to view parsing is that it also involves dependency classification at some point: if a parser is supposed to predict which word a given DET (belonging to the DET$\curvearrowleft$NOUN class) depends on, then it must at least be able to predict that this DET actually belongs to the DET$\curvearrowleft$NOUN class. So, the difficulty to annotate words belonging to a particular class is at least that of classifying them. On the basis of this classification subtask, I consequently consider that the ideas and metrics drawn from that literature also apply to parsing.

[19]Since my low-resource scenario rules out the option of dedicated feature engineering, the distinction between complex boundaries and ambiguous classes is in fact less relevant.

[20]Here separability is not to be understood as the formal property stating whether a dataset is separable (for a given classifier), but as how far the class is from being separable, i.e. the minimum number of misclassified points under perfect training (due either to the classifier being too simple for the overly complex boundaries, or to actual ambiguities).

[21]They still approximate it by the number of support vectors produced by a SVM trained on that data, but this measure is very specific to SVMs and does not render difficulty in general.

Zubek and Plewczynski (2016) thus take a more empirical approach and they rather consider the task difficulty from the perspective of *information saturation*: a dataset is complex if it is poorly approximated by its subsets, it is simple if with just a few examples one can already extract most of the knowledge embedded in the dataset. They measure this with the Hellinger distance between the probability distributions learned on the whole dataset (P^*) and on subsets of increasing sizes (P_{size}).[22] In the discrete case, Hellinger distance has two equivalent definitions:

$$H^2(P_{size}, P^*) = \frac{1}{2}\sum_x \left(\sqrt{P_{size}(x)} - \sqrt{P^*(x)}\right)^2 = 1 - \sum_x \sqrt{P_{size}(x)P^*(x)}$$

Zubek and Plewczynski (2016) simply define *data complexity* as the Hellinger distance, averaged over several random samples. They also define *conditional complexity*, in which they replace $P(x)$ by $P\left(x|\text{class}(x)\right)$ and then average H^2 over the classes. Supposedly, data complexity captures the complexity of the data structure as a whole, while conditional complexity assesses specifically classification difficulty, by taking the differences among classes into account. With these metrics, Zubek and Plewczynski (2016) draw complexity curves, estimate the dataset complexity based on the curve shape, and aggregate this information as a single measure: the area under the complexity curve (AUCC).

Data complexity versus learning complexity With the perspective I have adopted in this work, and which I will further describe in §6.4.2, the metrics proposed by Zubek and Plewczynski (2016) seem to satisfy my needs: they characterize how complex a piece of data is, and as far as conditional complexity is concerned, the criterion explicitly refers to the classes, and thus to the dependencies (not only to bare words). However, my interest in characterizing difficulties lies in the classifier standpoint: for a given piece of data, how hard will it be to learn *how to classify it*? Zubek and Plewczynski (2016)'s proposal, on the other hand, is more about the structure of the data itself and as I show below, it does not faithfully represent the classifier point of view. Consequently, as appealing as they are, I cannot apply directly their metrics in my case; their approach remains however qualitatively interesting, and a good inspiration for my own work.

The reasons for that mismatch are illustrated in Table 6.5, on a toy example where two features (child PoS and wordform) are available to predict only the direction of the dependency. The AUCC[23] metric states that dataset 2 is simpler than dataset 1. Indeed, in that case the attribute independence assumption loses some probability mass to (DET, noir) and (ADJ, la) instances, which lowers the probabilities and smooths their divergence. However, from the classifier standpoint, both

[22]Thanks to an attribute independence assumption, they compute the distributions easily, using only the feature frequencies in the dataset: $P(x) = \prod_{f_i \in \phi(x)} \text{frequency}_i(f_i)$.

[23]Following its definition, AUCC is computed as the integral of Hellinger distance for growing datasets. But $H(P_0, P^*)$ and $H(P_2, P^*)$ are necessarily 1 and 0 for both, so only $H(P_1, P^*)$ matters.

datasets have the same complexity, because in both cases there are two informations to learn;[24] so the data complexity metric is not satisfying in my case. Conditional complexity is not more suitable: because probabilities are now conditioned on the class, its value is exactly the same for datasets 3 and 4. Indeed, in both cases the PoS tag is uniform among each class, thereby removing the only difference between these datasets. However, the classifier has three informations to learn in dataset 3 but only two in dataset 4,[25] so in my perspective the metric should deem dataset 3 more complex than dataset 4. I interpret this mismatch as follows: Zubek and Plewczynski (2016) focus on $P(\text{features}|\text{class})$, i.e. the (generative) description complexity of *class contents*, while I am rather interested in the (discriminative) description complexity of *decision boundaries*, which would arise from $P(\text{class}|\text{features})$.

Dataset 1:	(NOUN, chat) ⇒ Left; (NOUN, souris) ⇒ Right
$\hookrightarrow H^2(P_1, P^*)$	$= \frac{1}{2}\left[\left(\sqrt{1\times 1} - \sqrt{1\times 1/2}\right)^2 + \left(\sqrt{1\times 0} - \sqrt{1\times 1/2}\right)^2\right] = 2 - \sqrt{2} \approx 0.6$
Dataset 2:	(DET, le) ⇒ Left; (ADJ, noir) ⇒ Right
$\hookrightarrow H^2(P_1, P^*)$	$= \frac{1}{2}\left[\left(\sqrt{1\times 1} - \sqrt{1/2\times 1/2}\right)^2 + \left(\sqrt{0\times 0} - \sqrt{1/2\times 1/2}\right)^2\right] = \frac{1}{2} = 0.5$
Dataset 3:	(NOUN, chat) ⇒ Left; (NOUN, chien) ⇒ Left; (NOUN, souris) ⇒ Right; (NOUN, rat) ⇒ Right
Dataset 4:	(DET, le) ⇒ Left; (DET, la) ⇒ Left; (ADJ, noir) ⇒ Right; (ADJ, verte) ⇒ Right

TABLE 6.5: Examples of datasets for which data complexity does not match the classifier point of view. Note that all examples are linearly separable, which allows to focus on complexity.

Instance-level measures Finally, the active learning literature also addresses these topics to some extent, in the sense that it often involves outlier detection, and classes containing many outliers are typically harder to learn. For instance, Mirroshandel and Nasr (2011) search for sentence parts that are especially difficult for the parser. In this perspective, Smith et al. (2014) propose a definition of *instance hardness*, which in their view corresponds to its expected accuracy (over a pool of various classifiers). Prudêncio and Castor (2014) then introduce the notion of *class hardness* (how hard a member of a given class is, overall) and compute it as the average instance hardness over the corresponding instances; in my opinion however, this bottom-up perspective does not match Smith et al. (2014)'s finding, whereby the hardness of an instance remains mostly due to class-level factors, class imbalance and overlap. I consequently seek a metric to evaluate class hardness directly at the class level.

6.4.2 Characterizing difficulties: class hardness

In my attempt to characterize how hard a given class is, I draw from all the works mentioned above, reusing conjointly the notions of learnability (Schwartz et al., 2012), information saturation (Zubek and Plewczynski, 2016) and class-level hardness (Prudêncio and Castor, 2014). I thus distinguish two complementary measures

[24] For example: chat⇒Left and souris⇒Right versus DET⇒Left and noir⇒Right.
[25] For example: NOUN⇒Left, souris⇒Right and rat⇒Right versus DET⇒Left and ADJ⇒Right.

of class hardness: the learnability (i.e. the UAS of a parser trained on massive data) of that class, and its Kolmogorov complexity, approximated by the minimum number of examples needed to extract *the available knowledge* (i.e. with respect to learnability, in order to single out their effects).

Let us illustrate both notions on an example. In the French UD 2.0, DETs almost deterministically attach to the first noun on the right (88%). As such, very high accuracies (above 95%) on their attachment seem guaranteed,[26] and the DET class can be said ‘highly learnable’. Besides, high scores can be achieved very fast, because this knowledge generalizes well, so the class is also simple. However, even though DETs normally attach on the right, their head is always on the left in the specific case of multi-word expressions (`flat:name` and `fixed` relations, as in ‘Pline$\curvearrowleft$l’ Ancien’ or ‘en$\curvearrowleft$tout cas’). As this concerns a handful of words and cannot be generalized, the only solution to learn the left-attaching cases is to learn the list of such expressions, which is a long description and requires a lot of sentences to see them all. But that list is finite and with very large data, high accuracies are achievable. The class of left-attaching DETs is consequently complex, but also highly learnable. Complexity corresponds in fact to the difficulty to learn *what can be learned*.

Intuitively, both metrics I consider can be read on a learning curve for growing datasets: learnability (the asymptote) is roughly the accuracy achieved with full data training and complexity (the learning speed) is the curve slope, with appropriate normalization to single out both effects. Hardness is the combination of low learnability and high complexity, and thus corresponds to the slope of the unnormalized curve.

Figure 6.6 presents the per-size learning curves of the UAS broken down by class (defined here as child depth in reference tree), without (‘Absolute’) and with (‘Relative’) normalization by the score achieved with 500 sentences, considered in this context as massive data. The experimental setup is again the same as in Section 5.2, including the fact that the displayed curves are averaged over all languages (after class-level normalization, for the relative curves). Unsurprisingly, all fine-grained curves have the same steep shape as overall curves: they rise quickly to high accuracies, and then continue to increase but much more slowly. This means that, *for each class*, most knowledge is extracted from the first few examples, and subsequent examples only marginally contribute to accuracy. However, there are significant speed differences between classes. Given that the focus is on tiny data and that the learning speed for the smallest treebanks is thus as much relevant as that for large treebanks, turning to a logarithmic scale (‘Log-relative’) gives a better readability of both speeds. Regarding interpretation, the slope of the relative curve shows the marginal utility of each new example, while the slope of the log-relative curve indicates the marginal utility of doubling the treebank size. Figure 6.7 reproduces the same kind

[26]In the course of my experiments, and looking only at trainsets over 50 sentences, I have trained 310 different parsers on French UD: 308 of them have an accuracy over 95% on DETs.

of measures for BEAM, but with a different class criterion, based on both the child PoS and its attachment direction.

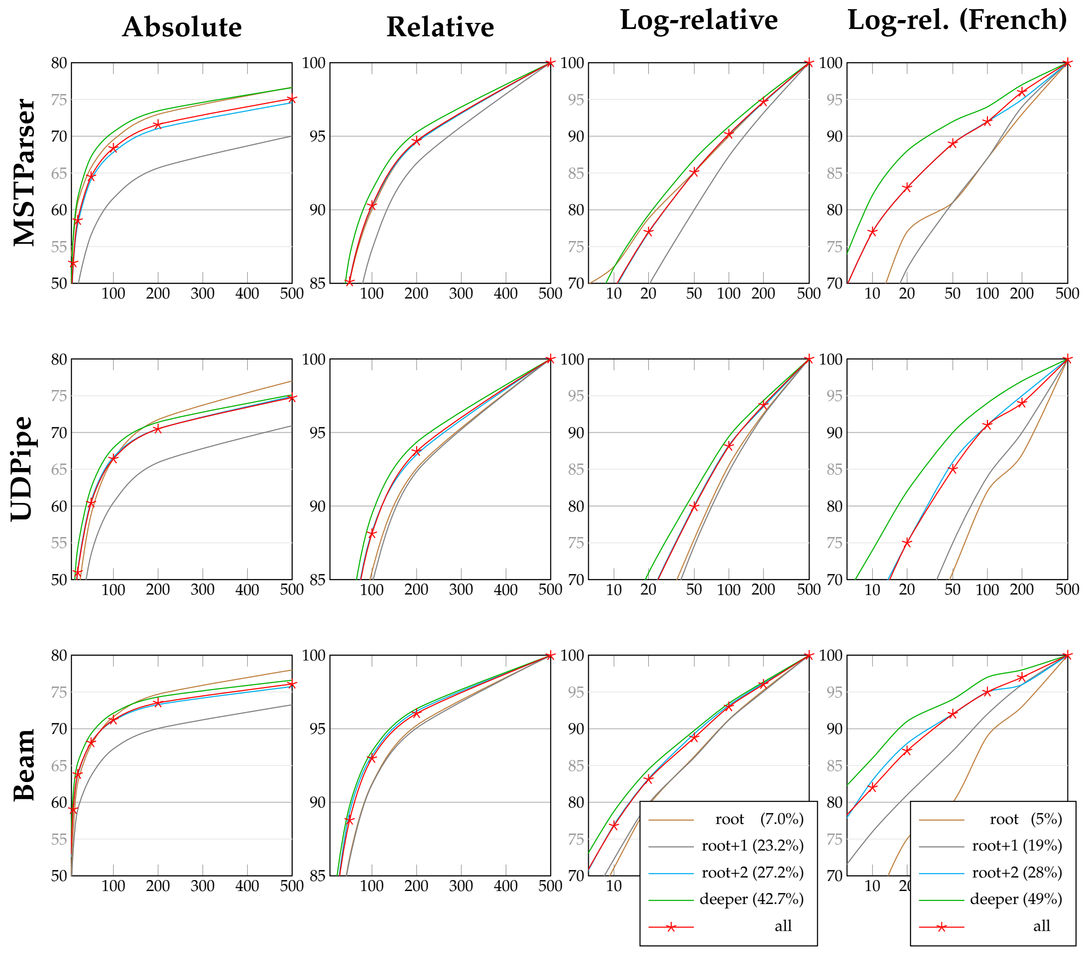


FIGURE 6.6: Learning curves (along data size) of the UAS, computed overall and separately on each depth class. Percentages indicate the size of each depth class. 'Absolute' views picture UAS explicitly, 'relative' views display $\frac{\text{UAS}_{size}}{\text{UAS}_{500}}$, and 'log-relative' ones follow a logarithmic scale.

What is intriguing in both Figures 6.6 and 6.7 is that (i) the relative curves can clearly be partitioned into two groups,[27] quickly-converging curves and slower-increasing ones, separated by a significant margin, and that (ii) the overall score clearly belongs to the former group, rather than being in the middle of both.[28]

While in Figure 6.6 the ranking of the relative curves is consistent with the size of the corresponding classes, in Figure 6.7 it is not: $\overset{\curvearrowleft}{\text{DET}}$ has a steeper curve than $\overset{\curvearrowleft}{\text{ADJ}}$, which is itself steeper than $\overset{\curvearrowleft}{\text{NOUN}}$. This hints at an interpretation of curve

[27]Because Figure 6.7 has 3 outliers, it contains in fact 4 groups, but such detailed distinctions are out of the present scope, and the two retained groups are the curves which start at 70% of their final value, and those which start around 50%.

[28]Notably, in all settings of Figure 6.6, the overall UAS follows very closely the 'root+2' curve, even though this class represents less than 30% of dependencies. This surprising fact remains to be investigated.

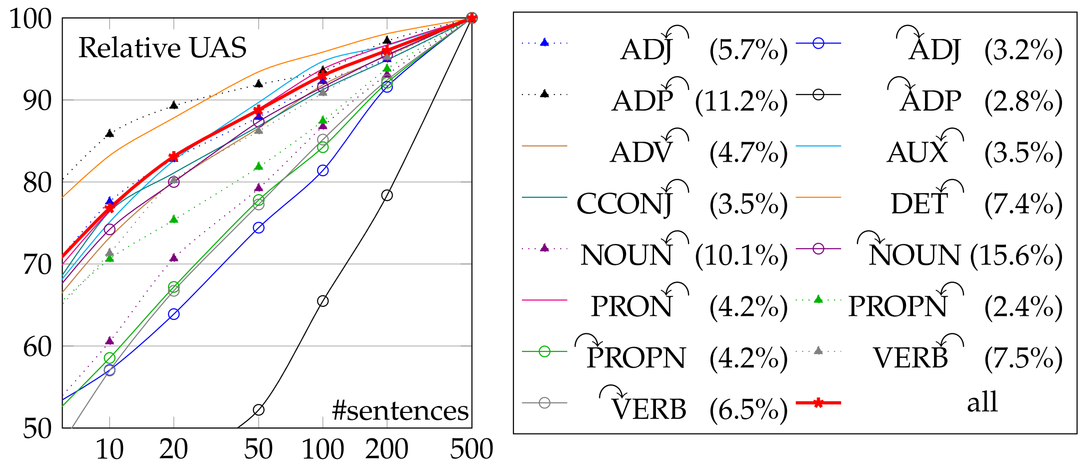


FIGURE 6.7: Log-relative learning curves of UAS by PoS/direction class (computed with BEAM and averaged over all languages with at least 30 occurrences in the class). $\overset{\curvearrowright}{\text{P}}$ (resp. $\overset{\curvearrowleft}{\text{P}}$) means all dependencies whose child PoS is P and reference head is on the left (resp. right, including roots). Percentages indicate the size of each class; the rarest ones are not represented.

groups which is rather based on interactions between classes. Indeed, in Figure 6.7, competing classes like $\overset{\curvearrowright}{\text{ADJ}}$ and $\overset{\curvearrowleft}{\text{ADJ}}$ (i.e. pairs of dependency classes involving attaching the same PoS in both directions) have 'opposite' learning curves, each one belonging to a different group (and even among such pairs the behaviors are not always consistent with frequency, as shown by $\overset{\curvearrowright}{\text{PROPN}}$ and $\overset{\curvearrowleft}{\text{PROPN}}$). Similarly, for depth-based curves, the groups correspond to the upper (the root and its children) and lower (all others) parts of the tree; and the upper dependencies consist precisely in identifying relations between subtrees that have been singled out by the lower dependencies. It would make sense that the classes requiring more training data are in fact those for which knowledge extraction can only be really efficient whenever another class has reached sufficiently high accuracies: this may be to use the other class as building blocks (like reduction of DETs before considering NOUN-VERB dependencies) or as a rule from which exceptions will be learned (like $\overset{\curvearrowleft}{\text{ADJ}}$ and $\overset{\curvearrowright}{\text{ADJ}}$). This hypothesis will be investigated at the level of features in Section 6.5.

A systematic evaluation of complexity requires to go beyond visual reading of curve shapes. But their slope is not properly defined, considering that for several classes, learning speed shows significant variations between 5 and 500 sentences. Like Zubek and Plewczynski (2016), I consequently compute areas to aggregate the curve shape into a single metric, but for cross-treebank comparison I calibrate it on the average curve. Formally, I define the COMPLEXITY of a class as the signed area[29] between the overall log-relative learning curve and that of the class.[30] A negative

[29]This can be computed easily using any numerical integration method. In the following experiments, I use Simpson's rule.

[30]The logarithmic scale is kept to ensure that the slope for tiny data contributes to the metric. In these experiments the area is computed between 5 and 500 training sentences, but another low-resource range could be chosen, provided that it is the same for all treebanks.

COMPLEXITY (curve above average) means that training on this class is simple (most knowledge is embedded in the first few examples), a positive value denotes a complex class (compared to the typical complexity of the treebank). HARDNESS can be defined similarly on logarithmic curves without normalization. From now on, the terms *simple* and *complex* are used to denote classes with negative and positive COMPLEXITY, while *easy* (resp. *hard*) classes are those with negative (resp. positive) HARDNESS.

LEARNABILITY	$\text{DET}^{\curvearrowleft}$	$\text{ADP}^{\curvearrowleft}$	$\text{AUX}^{\curvearrowleft}$	$\text{PRON}^{\curvearrowleft}$	$\text{SCONJ}^{\curvearrowleft}$	$\text{ADJ}^{\curvearrowleft}$	$\text{CCONJ}^{\curvearrowleft}$	$\text{ADV}^{\curvearrowleft}$	$\text{V}^{\curvearrowleft}$	$\text{PN}^{\curvearrowleft}$	$^{\curvearrowright}\text{PN}$	$\text{N}^{\curvearrowleft}$	$^{\curvearrowright}\text{N}$	$^{\curvearrowright}\text{ADJ}$	$^{\curvearrowright}\text{V}$	$^{\curvearrowright}\text{AUX}$	$^{\curvearrowright}\text{ADP}$
	91.3	89.0	83.9	82.4	80.2	80.0	77.1	76.1	75.1	69.0	68.4	68.2	67.9	60.6	56.4	52.8	48.0
COMPLEXITY	$\text{ADP}^{\curvearrowleft}$	$\text{DET}^{\curvearrowleft}$	$\text{PRON}^{\curvearrowleft}$	$\text{AUX}^{\curvearrowleft}$	$\text{ADJ}^{\curvearrowleft}$	$\text{CCONJ}^{\curvearrowleft}$	$^{\curvearrowright}\text{N}$	$\text{ADV}^{\curvearrowleft}$	$\text{V}^{\curvearrowleft}$	$\text{PN}^{\curvearrowleft}$	$\text{SCONJ}^{\curvearrowleft}$	$\text{N}^{\curvearrowleft}$	$^{\curvearrowright}\text{PN}$	$^{\curvearrowright}\text{V}$	$^{\curvearrowright}\text{ADJ}$	$^{\curvearrowright}\text{AUX}$	$^{\curvearrowright}\text{ADP}$
	-18.8	-18.7	-0.6	0.2	1.9	6.3	7.6	9.6	12.6	23.4	35.0	42.0	49.5	52.5	57.7	68.0	131.2
HARDNESS	$\text{DET}^{\curvearrowleft}$	$\text{ADP}^{\curvearrowleft}$	$\text{AUX}^{\curvearrowleft}$	$\text{PRON}^{\curvearrowleft}$	$\text{ADJ}^{\curvearrowleft}$	$\text{CCONJ}^{\curvearrowleft}$	$\text{ADV}^{\curvearrowleft}$	$\text{SCONJ}^{\curvearrowleft}$	$\text{V}^{\curvearrowleft}$	$^{\curvearrowright}\text{N}$	$\text{PN}^{\curvearrowleft}$	$\text{N}^{\curvearrowleft}$	$^{\curvearrowright}\text{PN}$	$^{\curvearrowright}\text{ADJ}$	$^{\curvearrowright}\text{V}$	$^{\curvearrowright}\text{AUX}$	$^{\curvearrowright}\text{ADP}$
	-79.6	-72.4	-33.2	-27.2	-20.1	-0.4	7.5	11.0	13.3	35.8	45.4	59.9	62.6	90.8	108.7	110.0	159.6

TABLE 6.6: Difficulty rankings (from easy to hard) measured by LEARNABILITY, COMPLEXITY and HARDNESS, based on the BEAM curves averaged over all treebanks. N, PN and V stand for NOUN, PROPN and VERB.

Beyond a sign-based grouping criterion, these quantitative measures also provide an interesting difficulty ranking. Table 6.6 compares the rankings based on LEARNABILITY (UAS for 500 training sentences), COMPLEXITY and HARDNESS. The results are consistent with intuition: the easiest classes are closed PoS classes with very deterministic attachments ($\text{DET}^{\curvearrowleft}$, $\text{ADP}^{\curvearrowleft}$), the hardest ones are rare ($^{\curvearrowright}\text{ADP}$) or semantics-driven attachments ($^{\curvearrowright}\text{VERB}$), while classes of average difficulty are mostly NOUNs and other open classes. Besides, these measures clearly show that HARDNESS is a trade-off between LEARNABILITY and COMPLEXITY (see the ranks of $^{\curvearrowright}\text{NOUN}$).

While some classes emerge as consistently simple or easy across languages, these properties can depend on the language, as revealed by separate computations for each treebank. For instance, $\text{ADJ}^{\curvearrowleft}$ is simple/easy and $^{\curvearrowright}\text{ADJ}$ is complex/hard in English, and in French it is the opposite, which is consistent with the respective frequencies of those classes (and with the interpretation as rule and exception). Similarly, $^{\curvearrowright}\text{ADP}$ is complex for all but 6 languages (from which the competing class $\text{ADP}^{\curvearrowleft}$ is virtually absent), and DETs, whose attachments are highly deterministic, are easy for all but 3 languages (Arabic, Latin-PROIEL and Old Church Slavonic). Also, several languages present as expected some complex but highly learnable classes (typically PARTs or SCONJs, but also the Japanese $^{\curvearrowright}\text{NOUN}$ for instance).

6.4.3 Comparative evaluation

These metrics can now be used for large-scale computation of fine-grained evaluations. Table 6.7 reports the average UAS when computed separately on simple and complex classes (that is, depending on the sign of COMPLEXITY), for all 6 parsers introduced in previous chapter. It appears that all systems present a significant score difference between both categories, in favour of the simple ones, in particular for tiny data (around 30 points).

	UAS_{10}		UAS_{500}		$\text{UAS}_{\text{full UD}}$	
	simple	complex	simple	complex	simple	complex
UDPIPE	56.4	28.0	82.1	66.8	88.0	78.1
PANPARSER	70.6	40.1	82.2	65.2	86.3	74.3
DELEX	69.1	41.8	78.5	62.0	80.8	66.2
MSTPARSER	68.0	36.9	83.5	66.1	89.1	77.4
BEAM	71.1	42.7	82.9	67.1	87.3	76.4
BEAM-DELEX	70.5	44.2	79.9	64.1	82.6	68.9

TABLE 6.7: UAS of the 6 studied systems, broken down by simple and complex PoS/direction classes. The partitions and corresponding scores are computed separately for each language and system.

It is important to note that this result was not expected: it is true that the computation of COMPLEXITY is based on measures of the UAS achieved using tiny training data; but these scores are normalized by learnability. Therefore, the simple classes are not *by design* those with high accuracies on tiny data, despite what the results suggest.[31] In this regard, the score difference between both groups remains a finding. Besides, the score gap is mostly maintained for full datasets, which means that the classes that have been singled out using tiny data have truly different properties (or at least are handled differently by the systems), whatever the data size.

Figure 6.8 pictures the learning curves of BEAM on simple and complex classes, as well as the split UAS for KL-BEAM on the same categories. It appears that the score gap is less pronounced for the cross-lingual parser, which results in differences in parsing capacity. Indeed, transfer is much more useful for complex classes (capacity of 60 sentences) than for simple classes, for which target data should be preferred much more often (starting from 14 sentences).

Unfortunately, with such definitions the proposed split cannot be performed in low-resource scenarios. Indeed, in order to decide for which dependencies tiny data should be preferred, one needs access to 500 annotated sentences, which is obviously out of the question. Since this size range is arbitrary, it is possible though to estimate simple and complex classes using only tiny data, by sampling 1, 2, 5 and 10 training sentences, and based on 10 validation and 10 test sentences (that is, all in all only 30 sentences are used).[32] The second subfigure of Figure 6.8 displays the resulting

[31]There remains however an indirect impact of absolute scores on COMPLEXITY: the classes that reach already high scores (over 80 UAS) with only 10 sentences are doomed to be simple classes, because being complex would imply achieving more than 100 UAS with 500 sentences. So, there is indeed a score bias toward simple classes. But such classes with early high scores represent a minority of the dependencies (19% on average), and there are many other kinds of simple classes (for a total of 57% of dependencies), including poorly learnable ones.

[32]Classes with less than 5 occurrences in test are considered complex by default, in order to alleviate the risk of throwing away useful cross-lingual knowledge because of poor estimation.

curves: differences remain, but they are much smaller and this evaluation is not so informative (curves are too close). However, it appears that using HARDNESS instead of COMPLEXITY (that is, distinguishing easy and hard classes depending on the sign of HARDNESS) provides quite similar results when 500 sentences are available, and this split is correctly approximated when using tiny data (third and fourth subfigures). Consequently, in low-resource scenarios, it is recommended to split between easy classes (learned in target) and hard classes (transferred), and estimated using tiny data. It is notably striking to see that only 10 training sentences allow to predict which classes will be annotated by KL-BEAM with an accuracy equivalent to as many as 60 sentences.

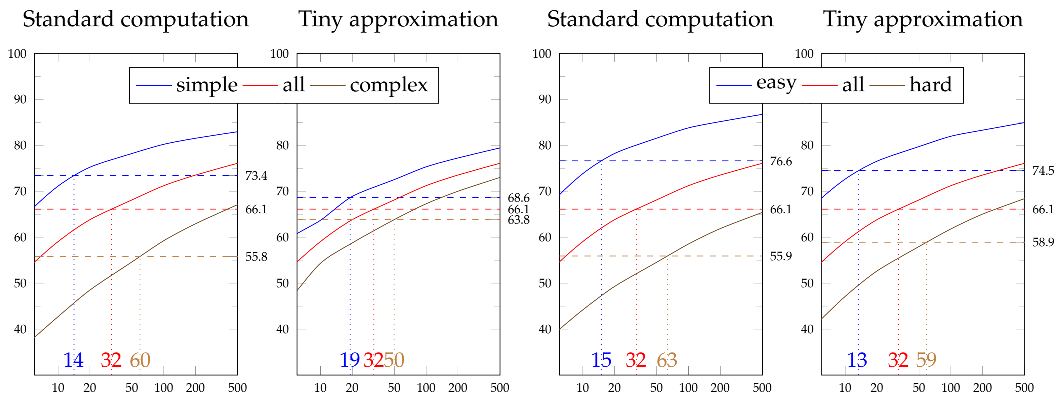


FIGURE 6.8: Overall, simple/complex and easy/hard UAS of KL-BEAM and BEAM parsers with increasing training sizes. COMPLEXITY and HARDNESS are computed with either 500 or 10 training sentences for each language; the displayed curves are averaged over the languages.

As a result of this investigation, for the purpose of combining cross-lingual and tiny parsers, hypothesis B of Figure 6.1 is validated and part of the SHARED knowledge is reassigned to the TARGET-SPECIFIC category: all easy classes, exhibited using HARDNESS values inferred from tiny data. Still, even though COMPLEXITY has been shown less adequate for this purpose, this notion remains crucial for understanding the learning mechanisms at work, as detailed below.

6.5 Interactions of unbalanced classes

This section completes this series of studies, by investigating in which ways the dependency classes can interact during training, in particular in case of strong imbalance. This broad topic is explored in a purely monolingual context, with the aim of evaluating hypothesis C of Figure 6.1: when only part of the dependencies are relevant for the task at hand, is it better to train the parser only on that part, or a bit more?

A particular emphasis is put on pairs of very similar classes, like DET$\curvearrowleft$ and $\curvearrowright$DET, ADJ$\curvearrowleft$NOUN and NOUN$\curvearrowright$ADJ, or NOUN$\curvearrowleft$NOUN and NOUN$\curvearrowright$NOUN. It has already been suggested (§6.4.2) that interactions between classes may be the main cause of complexity, so that $^{\curvearrowright}$DET for instance would be a complex class *because of* DET$^{\curvearrowleft}$. This study explores the idea of such interactions. Does the learning of one class hinder, or help, that of its main counterpart? Besides, does it affect other classes?

These interactions are analyzed thanks to two control experiments. First, all occurrences of one class are removed from the dataset, before training a parser. The impact on the other classes (in particular the most similar class) is then measured, both in terms of accuracy and of the values of learned parameters. Another experiment investigates whether the described interactions are due to timing only (typically, to explaining away effects), or specifically to the coexistence of both classes in the dataset: several parameters, which clearly relate to simple attachments, are initialized to their final value before training, using the same procedure as in §6.3.2.

This section presents and analyzes the results of these experiments, regarding accuracy (§6.5.1) and feature-level monitoring (§6.5.2). It concludes by reporting a series of attempts to integrate the findings into the training procedure, by means of word-level subsampling (§6.5.3).

The primary purpose of this work is to get better feature-level insights of the inner workings of parsers, to exhibit some mechanisms which are at work during training, and thereby to guide the design of a new transfer framework. Assessing whether these effects concern all languages, or even all classes in a given language, is consequently out of the scope of this work, so that I focus again on French and English, and on a handful of interesting classes.

6.5.1 Accuracy experiments: knowledge does not flow between similar classes

An English parser predicts ADJ$\curvearrowleft$NOUN attachments with a 93.9 UAS, but yields only 45.1 for NOUN$\curvearrowright$ADJ ones. One way to explain this difference is the marked imbalance between both classes (the former is 20 times more frequent than the latter). But the result remains counter-intuitive: in English, adjectives depend on nouns (at least when close to a noun), so why should it be more difficult to learn the dependency in one direction than in the other?

Figure 6.9a pictures the scores of an English parser on several PoS pair classes, and on top of the ADJ-NOUN discrepancy it also reveals large differences between the scores on NOUN$\curvearrowleft$VERB and VERB$\curvearrowright$NOUN (81.5 and 88.9 UAS), as well as NOUN$\curvearrowleft$NOUN and NOUN$\curvearrowright$NOUN (86.1 and 69.6 UAS). It is tempting to ascribe these differences to class size only (similarly to the ADJ-NOUN case, there are more occurrences of VERB$\curvearrowright$NOUN than of NOUN$\curvearrowleft$VERB), but the NOUN-NOUN pattern invalidates this hypothesis: the NOUN$\curvearrowleft$NOUN attachment is less frequent than NOUN$\curvearrowright$NOUN and still much more accurate.

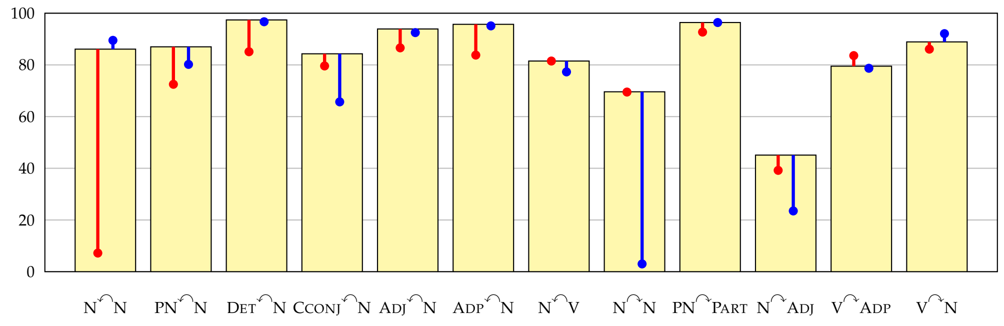
To further investigate this, I experiment with filtering of specific dependency classes: does the UAS on a given class change, depending on whether its counterpart is present or absent from training data? In practice, I simulate its absence with quite naive means: I remove from the dataset any sentence containing an occurrence of the counterpart class.[33] As expected, when removing all NOUN$\curvearrowleft$NOUN instances (Figure 6.9a), their prediction accuracy drops (-7.2 UAS). But what is more interesting is that this removal does not affect NOUN$\curvearrowright$NOUN at all; it would have been reasonable to expect its score to either drop (because some information is lost on the fact that NOUNs can depend on NOUNs) or increase (because when processing a pair of NOUNs, it becomes easy to dismiss at least one of the options, predicting a left dependency), but an unchanged score is surprising. Similar experiments with ADJ$\curvearrowleft$NOUN and NOUN$\curvearrowright$ADJ in English (Figure 6.9b) and French (Figure 6.9c) yield similar results.[34] It is clear then that knowledge does not flow between comparable classes, so that when seeking to improve a given class, the model fails to exploit knowledge on its counterpart.

However, Figure 6.9 also displays the effect on a few other classes presenting either substantial or interesting differences, and it reveals that a big change on the size of a class has a significant impact on other classes as well; for instance the deletion of NOUN$\curvearrowleft$NOUN occurrences reduces the accuracy of dependencies other than NOUN-NOUN from 86.5 to 82.8 UAS. Notably, deleting a left dependency class significantly degrades several other left dependency classes (gathered on the left), and removing NOUN$\curvearrowleft$NOUN has a pervasive impact on most left children of NOUN. So, there is in fact some knowledge sharing among comparable classes, presumably through overall direction biases. And yet, this is not the case for CCONJ$\curvearrowleft$NOUN, which is much more impacted by the disappearance of NOUN$\curvearrowright$NOUN (or NOUN$\curvearrowleft$VERB by NOUN$\curvearrowright$NOUN, or NOUN$\curvearrowright$NOUN by ADJ$\curvearrowleft$NOUN in English).

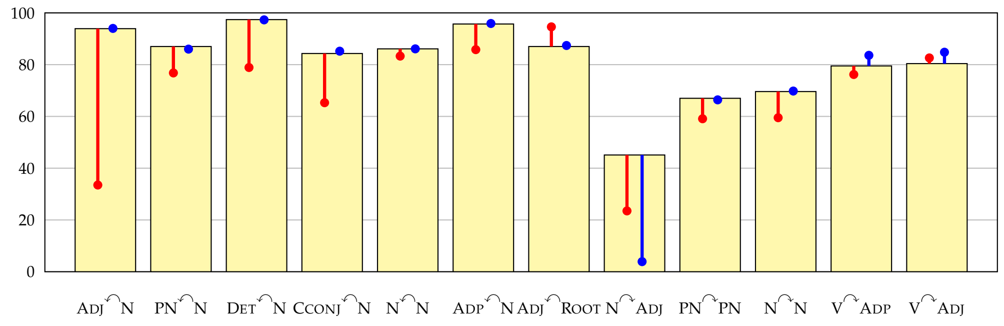
Degrading accuracy overall is expected, because this sentence filtering downsizes a lot the dataset (down to 50-70% of its original size, in most cases), and such data size reduction has a cost; but this does not explain why the classes are differently impacted. Of course, removing for instance all English sentences containing PROPN$\curvearrowright$PART (genitive marker) would also remove most occurrences of

[33]For soundness, the validation set undergoes the same filtering.

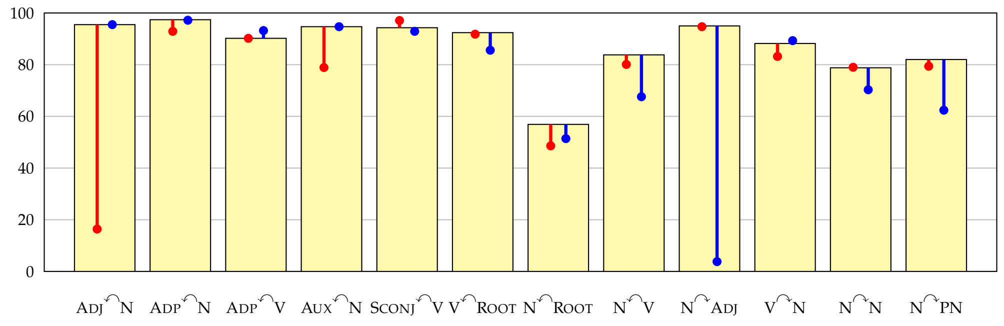
[34]There is an exception though: in English removing ADJ$\curvearrowleft$NOUN hurts significantly NOUN$\curvearrowright$ADJ. But the impacted class is so infrequent in training data that it is hard to conclude: a number of NOUN$\curvearrowright$ADJ dependencies are also deleted when removing that many sentences, and the resulting dataset may have a too small coverage of this kind of dependencies.



(a) Fine-grained UAS in English. Red marks (on the left) picture the effect of deleting NOUN$\curvearrowleft$NOUN sentences, and blue marks (on the right) that of deleting NOUN$\curvearrowright$NOUN sentences. Respective data sizes are 12,543, 9,108 and 8,671 sentences.



(b) Fine-grained UAS in English. Red marks (on the left) picture the effect of deleting ADJ$\curvearrowleft$NOUN sentences, and blue marks (on the right) that of deleting NOUN$\curvearrowright$ADJ sentences. Respective data sizes are 12,543, 7,703 and 12,112 sentences.



(c) Fine-grained UAS in French. Red marks (on the left) picture the effect of deleting ADJ$\curvearrowleft$NOUN sentences, and blue marks (on the right) that of deleting NOUN$\curvearrowright$ADJ sentences. Respective data sizes are 14,553, 10,718 and 7,136 sentences.

FIGURE 6.9: Fine-grained UAS of English and French parsers, and impact of deleting all training sentences containing a given class. N, PN and V stand for NOUN, PROPN and VERB.

PROPN$\curvearrowleft$NOUN (possession), and the naive approach of whole sentence deletion has surely an effect on the dependencies that often co-occur with the deleted class. But there is not an obvious link for all concerned classes, so maybe there is more here than meets the eye.

Notably, the UAS *increases* which occur with shrinked datasets are particularly intriguing: in English, removing NOUN$\curvearrowright$NOUN improves NOUN$\curvearrowleft$NOUN and

VERB$\curvearrowright$NOUN by +3.5 and +3.2 UAS (both being direct alternatives), removing ADJ$\curvearrowleft$NOUN improves ADJ$\curvearrowleft$ROOT (a competing attachment) by +7.6 UAS, but removing NOUN$\curvearrowright$ADJ also improves VERB$\curvearrowright$ADP by +4.1 UAS, which seems unrelated. Although this last increase remains unexplained, the others have straightforward interpretations in terms of reduction of ambiguity (which simplifies the decisions).

This experiment has uncovered a series of counter-intuitive measures and inconsistent effects, and has rather raised questions than solved them. But at that point, one can already formulate three findings: first, classes are mostly learned independently and knowledge hardly flows between them, even the most similar ones; as a result, the minority classes have often poorer accuracies than expected, because their prediction does not exploit all the information that is intuitively available; finally, the absence of a given class has a mixed influence on competing classes, but also on seemingly unrelated ones. These findings will not be further developed here, but more will be said in Chapter 8 regarding the lack of knowledge sharing between similar classes, and in particular on its impact on transfer.

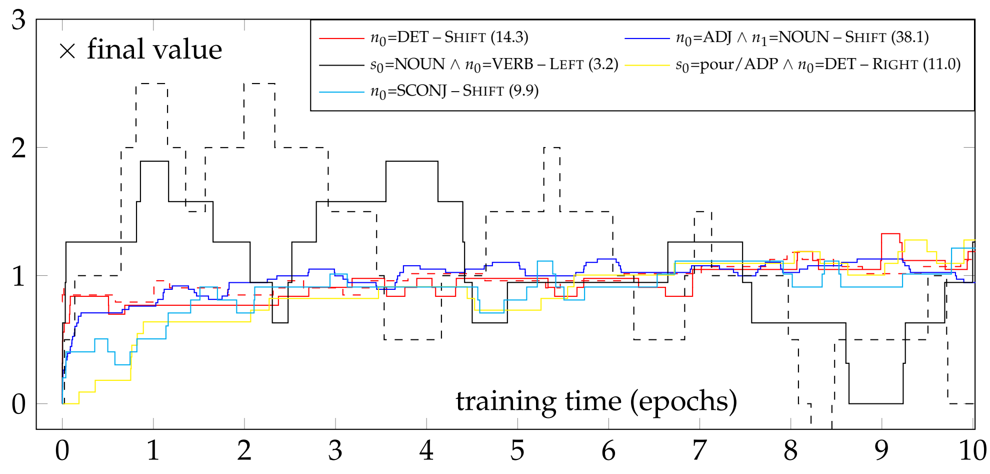
6.5.2 Feature-level experiments: complex classes result from unstable parameters

The following investigates more specifically the interactions between simple and complex classes, and for that purpose it moves to the level of feature weights. A French parser is trained as usual, except that during training each feature weight is monitored. Figure 6.10 pictures a few of the resulting weight curves, normalized by their final averaged value. It reads as follows: for feature s_0=NOUN $\wedge$ n_0=VERB, the averaged model has a weight of 3.2 for the LEFT action, and this parameter reaches a value of $1.9 \times 3.2 \approx 6$ at the end of epoch 1.[35]

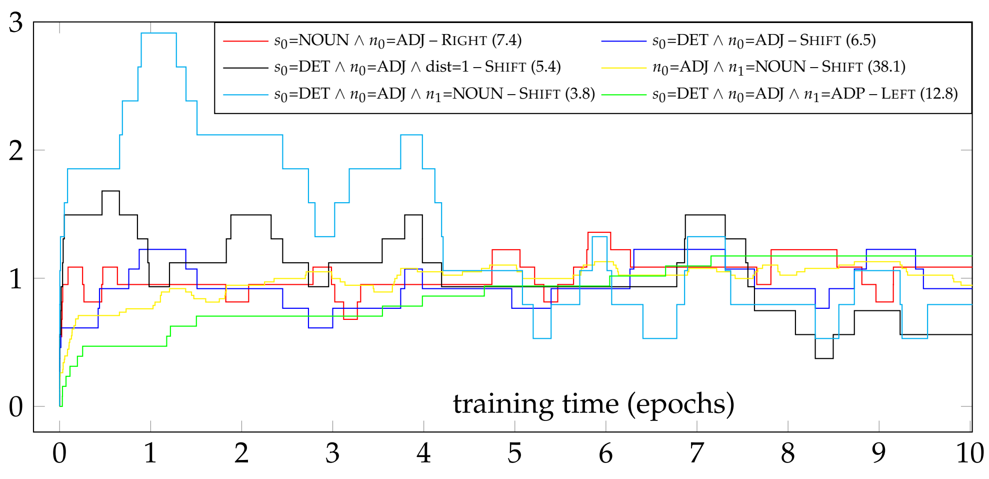
It appears that simple classes (DET$\curvearrowleft$ and ADJ$\curvearrowleft$NOUN) are learned quite straightforwardly on a single generic feature: the feature weight quickly reaches the neighbourhood of its final value and sticks there. On the other hand, features associated with complex classes (NOUN$\curvearrowleft$VERB and $\curvearrowright$DET) present big oscillations all along: such ambiguous classes can indeed not be fully learned with simple delexicalized features, and there is no convergence for these weights (their final value results solely from averaging). Meanwhile, for rare but simple classes (like SCONJ$\curvearrowleft$), feature weights have a slower but monotonic growth, paced by their rare occurrences.[36] These curve shapes are consistent with intuition, and I assume in the

[35]For legibility, the smallest oscillations (1-point oscillations or peaks of less than 3 points during less than 0.2 epoch) are not pictured.

[36]It is interesting to note that training-time curves do not behave the same as data-size learning curves: the difference between simple and complex classes is not on convergence speed, but on convergence itself. Indeed, the above data-size considerations were about the *amount of knowledge* extracted from the data, and never stated *how fast* this knowledge could be extracted.



(a) Feature weights corresponding to several classes: respectively DET$\curvearrowleft$, ADJ$\curvearrowleft$NOUN, NOUN$\curvearrowleft$VERB, pour$\curvearrowright$DET, SCONJ$\curvearrowleft$. There are two kinds of curves here: monotonic curves with smooth convergence (simple classes) and unstable ones which do not converge (complex classes; see also $\curvearrowright$DET in Figure 6.11). The dashed curves show the effect on DET$\curvearrowleft$ (simple) and NOUN$\curvearrowleft$VERB (complex) classes of preinitializing several parameters associated with simple classes: in both cases, the general shape of the curve remains unchanged.



(b) Significant feature weights corresponding to the attachment of ADJ. Parameter tuning for the 'DET ADJ NOUN' pattern uncovers an interesting interaction. The SHIFT curves for s_0=DET ∧ n_0=ADJ ∧ dist=1 and s_0=DET ∧ n_0=ADJ ∧ n_1=NOUN (parameters stating that ADJ is not head of DET) do converge, but painfully: after an initial over-estimation (first epoch), and then exchanges of weight mass between both features (epochs 2-4), they stabilize simultaneously when a feature describing the competing class (s_0=DET ∧ n_0=ADJ ∧ n_1=ADP ⇒ LEFT, corresponding to DET$\curvearrowleft$ADJ) reaches its final weight (during epoch 5).

FIGURE 6.10: Evolution of several feature weights during training. Each curve is normalized by the final averaged value of the corresponding parameter, reported in parentheses in legends.

following that instability of unlexicalized parameters characterizes complex classes, i.e. those which would require much more data to be less ambiguous.

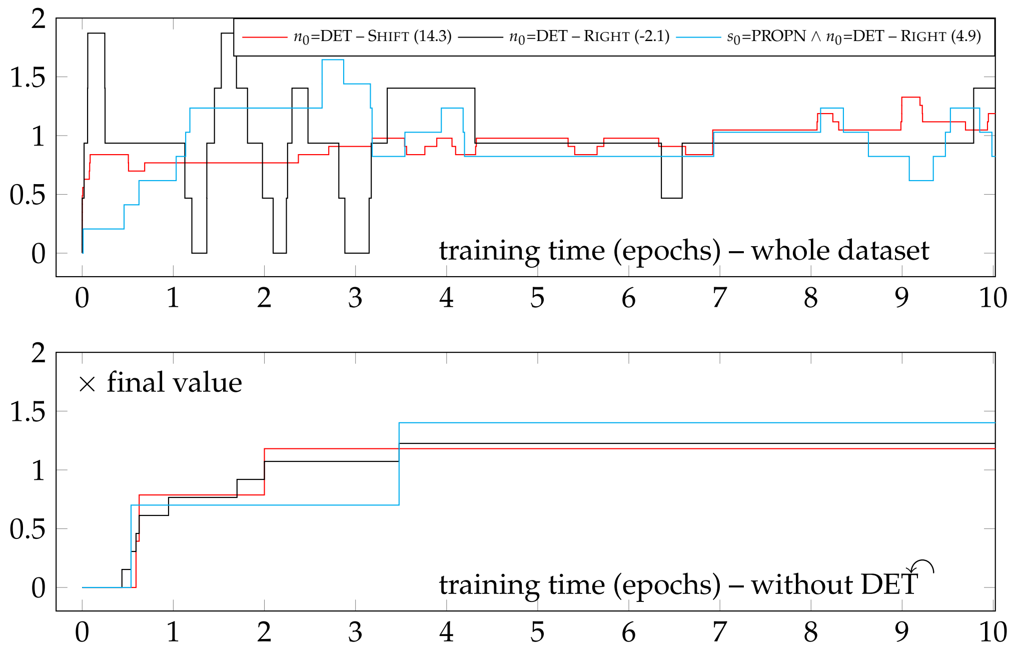


FIGURE 6.11: Removal of sentences containing preposed DETs: impact on parameters associated with the simple and complex classes. All curves become monotonic, with a smooth and slow convergence, usually characterizing rare simple classes. Note that for legibility, the smallest oscillations of the upper curves are not pictured, but all updates are represented for the lower ones.

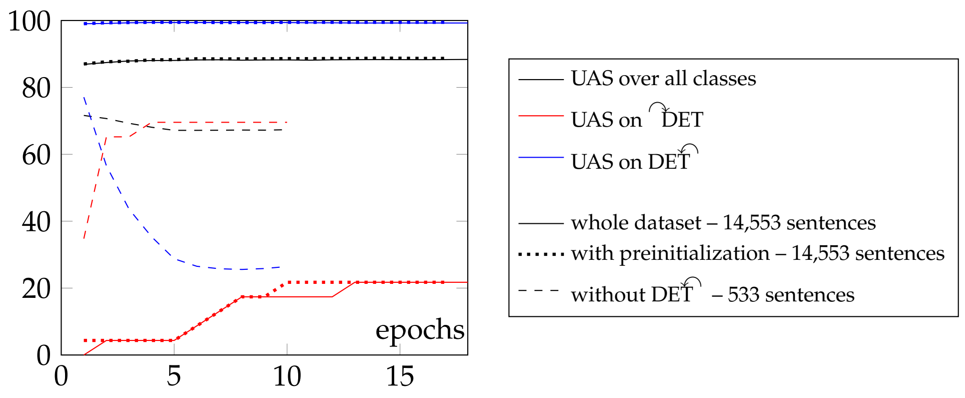


FIGURE 6.12: Impact on accuracy (overall and fine-grained) of both experiments, removing training sentences and preinitializing some parameters.

To investigate possible interactions between simple and complex classes (following the hypothesis formulated in §6.4.2), I experiment again with filtering of the training data. In the case of DETs,[37] all sentences containing the simple class $\text{DET}^{\curvearrowleft}$ are removed from the dataset. Figure 6.11 reports the effect on parameters associated with $\text{DET}^{\curvearrowleft}$ and $^{\curvearrowright}\text{DET}$. It appears that both curves corresponding to the complex

[37]Similar experiments have been conducted for $\text{ADP}^{\curvearrowleft}$ (n_0=ADP ⇒ SHIFT) and $^{\curvearrowright}\text{ADP}$ (s_0=ADV ∧ n_0=ADP ⇒ RIGHT and s_0=ADV ∧ n_0=ADP ∧ n_1=NOUN ⇒ RIGHT), yielding very similar results.

class ${}^{\curvearrowright}\text{DET}$[38] become much smoother when the simple class disappears. The shape of the new curves is in fact that of simple classes (almost monotonic, flat after a few epochs); has ${}^{\curvearrowright}\text{DET}$ become a simple class? The effect is also visible on UAS (Figure 6.12): ${}^{\curvearrowright}\text{DET}$ is learned much better and much faster when $\text{DET}^{\curvearrowleft}$ is absent.[39] These curves are also remarkable in another regard: it is the first time in this work that a French parser is accurate on ${}^{\curvearrowright}\text{DET}$; even though it happens here at the expense of $\text{DET}^{\curvearrowleft}$, this dismisses the idea that such dependencies are just impossible to learn and should be given up.

These observations suggest that the big weight variations noticed for complex classes are not an intrinsic property of these predictions, but are only due to their coexistence with the corresponding simple class, which complexifies the decision boundaries and thus makes its counterpart complex (by making it a mere exception to a simpler rule).

One way to explain that interaction would be that some parameters wait for the final value of another parameter, and cannot converge before the other one has stabilized. The timing interaction uncovered in Figure 6.10b supports this hypothesis: parameter tuning relies on a careful balance between the weights of co-occurring features, so that those representing exceptions must continuously be reestimated, whenever the parameter representing the rule is updated. It is similarly hard to believe that parameters associated with $\text{NOUN}^{\curvearrowleft}\text{VERB}$ dependencies can be accurately tuned before the noun phrases to attach have been properly identified (typically, before determiners have been attached to nouns). To verify this hypothesis, I alter the timings with a preinitialization trick, similarly to what has been done in Section 6.3: the weights of several features which are characteristic of a handful of relatively simple classes (chosen based on the ranking of Table 6.6: $\text{P}^{\curvearrowleft}$ for P=DET, ADP, AUX, PRON, CCONJ; ${}^{\curvearrowright}\text{P}$ for P=ADJ, NOUN)[40] are preinitialized to their final averaged value, then training unfolds as usual. As shown in Figure 6.10a, no significant effect is visible: having good $\text{DET}^{\curvearrowleft}$ predictions right away does not make the $\text{NOUN}^{\curvearrowleft}\text{VERB}$ curve any smoother. Moreover, the impact on accuracy of this preinitialization (Figure 6.12) is only marginal: ${}^{\curvearrowright}\text{DET}$ dependencies are not better learned when the $\text{DET}^{\curvearrowleft}$ class is known early.[41] The instability of weights associated to complex classes is consequently not due to timing interactions.

Hence, for complex classes, the issue seems really to be the existence of a (more frequent) competing class, and not the time required to learn it: large weight variations result from feature-level contradictions of both classes, and will thus occur

[38]These are n_0=DET $\Rightarrow$ RIGHT and s_0=PROPN $\wedge$ n_0=DET $\Rightarrow$ RIGHT.

[39]Test data is unchanged: it includes the $\text{DET}^{\curvearrowleft}$ sentences in all cases.

[40]For $\text{P}^{\curvearrowleft}$, the chosen features are those containing references to n_0=P and whose final weights prefer the SHIFT action, and those containing s_0=P and whose final weights prefer the LEFT action. For ${}^{\curvearrowright}\text{P}$, they are those containing n_0=P and whose final weights prefer the RIGHT action.

[41]The overall score is still improved in the same proportions as in Section 6.3, i.e. +0.4 UAS.

whenever the model is required to optimize both, even if one is tuned before the other. For this reason, I can see only one strategy to avoid such non-converging classes: learning both kinds of classes separately, instead of mixing both objectives in the training material. With this approach, parsers are required to learn *less classes*, which is a simpler task, with simpler class boundaries – and hopefully better accuracies, like those obtained for $\overset{\curvearrowright}{\text{DET}}$. The rest of this section further supports this idea by reporting a series of attempts to boost rare and hard classes, whose modest results (at least compared to their apparent potential) are precisely due to the fact that hard classes remain mixed in training data with the easy ones.

6.5.3 Word-level subsampling

In accordance with its traditional use on imbalanced training sets (Kubat and Matwin, 1997), subsampling has been shown above to yield significant improvements on specific classes. But entertaining it at the sentence level has revealed a major downside: it often shrinks dramatically the dataset, which incurs a huge loss of knowledge for even unrelated classes. I thus turn now to experiments with subsampling *at the word level*.

In practice, such sampling can be done by a few adaptations on the update procedure: not learning from a given dependency corresponds straightforwardly to not performing the updates it triggers. After such a canceled update, I choose to pursue training from the gold action (as if the model was already better and there was no need to update).[42] So, the subsampling procedure simply consists in enforcing a gold action whenever the model attempts to violate the considered dependency.[43]

I experiment on French with 3 subsampling criteria: shrinking the frequent, the most learnable and the easiest classes. In contrary to the previous experiments, here several classes undergo subsampling at the same time, and I do not delete but only shrink them: before exploiting each example, each dependency is removed from training material with a probability that depends on its class. This probabilistic procedure ensures that the affected dependencies are not removed from the available knowledge (the parser can still learn from them in the next epoch), but will only contribute less to training.

[42]The alternative, pursuing from the predicted action, would imply that the parser's decision was correct and that the considered dependent was indeed attached elsewhere. Here I only want to extract less update material, not to validate the parser's choices regarding tokens it has not seen a lot, therefore I choose the gold action.

[43]In the next chapter, this kind of adaptation will be described as a 'dependency constraint', whereas pursuing from the predicted action would correspond to 'partial training'.

	all	ADJ		ADP		ADV		AUX	CCONJ	DET		NOUN		NUM		PRON		PROPN		SCONJ		VERB	
		↶	↷	↶	↷	↶	↷	↶	↶	↶	↷	↶	↷	↶	↷	↶	↷	↶	↷	↶	↷	↶	↷
Size ($\times$1,000)	317.1	5.8	14.4	55.2	2.0	9.1	3.6	12.2	9.0	54.4	0.3	13.9	52.5	4.8	4.6	14.5	1.5	3.9	23.3	1.9	0.8	11.7	16.1
Baseline UAS	88.3	91.1	93.0	96.6	40.3	89.0	81.3	96.7	88.1	99.3	21.7	72.2	80.0	93.5	74.7	96.5	77.0	80.8	86.3	88.9	75.8	86.8	71.4
Freq.-based	88.3	91.7	93.0	96.2	**48.1**	87.8	80.7	97.0	89.3	98.4	30.4	**76.9**	78.4	**95.7**	75.8	96.3	80.3	**86.3**	85.8	**91.9**	72.7	87.7	71.0
Acc.-based	87.5	91.7	90.1	94.1	**61.0**	88.1	82.0	**97.5**	86.9	95.5	**65.2**	73.8	79.3	92.8	75.8	95.5	75.4	**87.7**	85.4	88.9	75.8	84.5	**75.4**
Dyn. acc.-based	88.5	91.7	92.5	96.4	**49.4**	89.6	**83.3**	97.0	88.9	98.7	34.8	**74.0**	79.9	94.2	73.7	96.3	77.0	**89.0**	85.6	88.9	72.7	**86.0**	72.7

TABLE 6.8: Effect on fine-grained UAS of various strategies for word-level subsampling, measured on French. Sizes written in red denote classes impacted by frequency-based subsampling (above 12,685 occurrences in trainset). Blue (resp. red) scores correspond to exceptionally high gains (resp. drops) with respect to the class frequency.

Frequency-based subsampling In the first set of experiments, the size of each class is capped to 4% of all dependencies: classes with a higher frequency are probabilistically shrinked to this size. Here I consider classes based on child PoS and attachment direction;[44] this leads to removing around 40% of all dependencies. With this subsampling, convergence is much longer (46 epochs instead of 15), but the final accuracy is the same. According to Table 6.8, several minority classes benefit a lot from a reduced imbalance, in general at the expense of the competing class (see ADP or PROPN for instance). The overall UAS is unchanged, but this is because a few very frequent classes (ADP↶, DET↶, ↷NOUN) class suffer decreases which are comparatively low but weight a lot in the overall score.

Accuracy-based subsampling Subsampling the most learnable classes is done using the test accuracy of the fully trained baseline parser. All classes with an UAS over 70 (and enough occurrences in test set to be evaluated reliably, here 30) are downsized according to their score, at a linear rate: classes with score *s* are reduced to $\frac{100-s}{100-70}$% of their initial size. Thus, the classes which require more training are favoured. Using the same class criterion as above,[45] this removes around 56% of dependencies. As reported by Table 6.8, it induces a significant drop of overall score. Indeed, the few best classes (above 95 UAS) are particularly impacted by this subsampling, become suddenly quite rare and are much less trained, which overshadows the numerous improvements achieved on other classes.

However, this experiment has a very interesting effect on ↷DET, which now rises to 3 times its baseline UAS. This confirms in a more realistic setting the previous finding (Figure 6.12) that the ↷DET class is not impossible to learn, as usual results suggest, but is only impeded when DET↶ is 200 times larger (40 times larger with

[44]Using only child PoS tags as a class criterion inflicts an undue penalty to minority classes and is detrimental overall (-0.4 UAS): for instance the accuracy on ↷DET drops to 4.3 UAS, because of the size penalty incurred by DET↶. I also consider relation labels, but here they yield similar results to child PoS with direction.

[45]Here again, classes based on PoS only are detrimental to e.g. ↷DET (whose UAS drops to 0), and relation labels present similar effects, but with an even worse overall score (87.1 UAS).

frequency-based subsampling, only 4 times with accuracy-based subsampling). Notably, here both classes achieve relatively high scores at the same time, which was not the case in Figure 6.12.

Another effect worth mentioning is that even though the convergence time does not significantly increase (19 epochs instead of 15), the learning of individual classes appears tangibly much slower. For instance, at the end of first epoch, DET$^{\curvearrowleft}$ is 5 points below its final score, while in the baseline it is already 0.2 below an even higher final score. Also, if a threshold is set at 1 point below final score, DET$^{\curvearrowleft}$ does not reach it before epoch 10 (epoch 19 instead of 8 for VERB$^{\curvearrowleft}$ which is less impacted, and 7 instead of 2 for overall score). Interestingly, this delay concerns also classes which are much less affected by the subsampling process (NOUN$^{\curvearrowleft}$: epoch 7 instead of 3; $^{\curvearrowright}$NOUN: epoch 14 instead of 5): this supports the idea that they use easy classes (typically DET$^{\curvearrowleft}$) as building blocks, in which case they need more time here because easy classes have not stabilized yet.

Dynamic accuracy-based subsampling To prevent the initial delay that impacts the best but also other classes, this last set of experiments adopts a progressive approach: subsampling is applied to accurate classes, but only as soon as they achieve high scores *during training*. These are the easy classes; they have taken some advance (with respect to the hard ones) in extracting the available knowledge, so they do not need as much learning efforts and are slowed down. In practice, the good predictions made during training inference are tracked (ignoring of course the enforced ones), and exploited at the end of each epoch to re-estimate the current (cumulative) score for each class, to be used in the $\frac{100-s}{100-70}$ coefficient as above. Note that the first epoch is thus unaffected. This criterion appears indeed progressive: empirically, 46% of dependencies are removed on second epoch, 62% on 10th.

Convergence is now faster than in baseline (10 epochs), and Table 6.8 additionally reports that this subsampling improves overall UAS. The first (unchanged) epoch is indeed leveraged as usual by easy classes to quickly reach high accuracies, and the next epochs focus on the harder classes, which require dedicated efforts; as a result, drops on accurate classes are alleviated, while preserving gains on others. Analysis reveals that learning delays have disappeared. It thus seems possible to have the model pursue two separate objectives, only by adapting the training procedure.

However, it appears that focusing on hard classes in later epochs is still detrimental to the easy ones, because the learned parameters that contribute to their good predictions (specific features, but also shared biases) are not frozen: they are continuously affected by training of hard classes, but without sanity check on easy classes. Typically, DET$^{\curvearrowleft}$ quickly reaches an UAS of 99.3 (epoch 2), but then it contributes only rarely, which does not leave enough room to adapt its weights to the new third-party parameters, and it slowly decreases down to 98.7 UAS.[46]

This exploration of subsampling confirms several interaction mechanisms mentioned earlier, but it also presents concrete attempts to handle classes which are over-represented (either because of natural imbalance, or just because they are not so relevant). However, results highlight the unavoidable – and harmful in this context – effects of such interactions. Class subsampling incurs delays on specific classes which in turn slow down the whole learning process; and this delay can easily be avoided with a progressive subsampling, but only at the cost of making the easy classes vulnerable to harmful third-party updates. These findings support the strategy of *freezing* once for all the learned parameters corresponding to previously extracted knowledge (typically, on simple classes), except that this set of parameters is not known exactly. On another hand, I have already mentioned the idea of *splitting the task* at the class level (in which case the model takes care of identifying the appropriate features), thereby making the parser learn less classes at a time. It ensues that there is a way to apply all those ideas together: to split the traditional catch-all model into a series of independent submodels, each one focusing on a subset of classes and with no interaction during updates.

Coming back to the main hypothesis of this section (C in Figure 6.1), whether the parser of a given source should train only on the relevant part of the dependencies, there is no definite answer. On the one hand, some classes can be noxious to the rest of the predictions, so that if those are in fact irrelevant, they have no reason to be kept in training data. But on the other hand, training only on the relevant dependencies overlooks the fact that the irrelevant ones can correspond to basic syntactic mechanisms (like determiners) which are essential to learn the other classes. As a result, the best strategy seems to be in between: reducing the task of each parser to the relevant dependencies, but augmented with a few irrelevant ones. Hopefully, these basic dependencies correspond mostly to the easy ones, for which estimation techniques have already been developed in Section 6.4, even in low-resource scenarios.

6.6 Wrap-up: more is both more and less

The purpose of this chapter was to better understand the inner workings of parsing, and how heterogeneity affected transfer efficiency. Among other findings, it has uncovered several ways in which classes interact and provided better insights on feature-level interpretation. Drawing firmer conclusions on those broad topics is out of the scope of this work, but a number of properties revealed along this study

[46]In an attempt to preserve the accurate classes from the negative impact of third-party updates, I have also experimented with more dynamic criteria, like the score on previous epoch only, but the results were not significantly better.

(root-leaves discrepancy, impact of lexicalized features, generalization ability, class complexity, feature-level convergence, as well as the importance of timing during training) deserve to be investigated in future work.

Regarding specifically the cross-lingual application, this series of experiments has also been very instructive for refining the design of my transfer framework.

Section 6.1 has pointed out the deficiency of available tools for intrinsic evaluation: they fail to assess the usefulness of tiny or cross-lingual parsers (that is, how informative the resulting parses are). Section 6.4 fills this gap by proposing a formal distinction between classes containing much and little knowledge. Based on the intuition that there are two ways to learn a piece of knowledge, one simple (with a few generalizing rules, typically unlexicalized) and one complex (by learning a precise behaviour for each concerned word in turn, which requires much more data), it introduces concrete measures to evaluate the behaviour for each class in this regard. These tools have allowed me to exhibit how classes can be handled differently by parsers, and to characterize the classes for which cross-lingual transfer is not worth it, compared to tiny parsers. This study has thus led to reassigning some knowledge considered as SHARED, as if it was TARGET-SPECIFIC; in other terms it not only validates hypothesis B of Figure 6.1, but also proposes an empirical way to identify some of the concerned dependencies.

Coming back to the questions left open at the end of the previous chapter, this chapter has unveiled several concurrent effects which exemplify how learning even *irrelevant* pieces of knowledge may become necessary for proper transfer (which corresponds to hypothesis C). While it is true that there is not much knowledge sharing between close or related classes (§6.5.1) and as such irrelevant dependencies are not explicitly used to annotate target data, they may still have a significant effect on the quality of training. Indeed, Section 6.3 uncovers how learning SOURCE-SPECIFIC knowledge can in fact improve generalization on what is SHARED (thereby validating hypothesis A): by means of explaining away effects (§6.3.2), granting access to lexicalized features during training of a delexicalized parser makes the model rely on such parameters. This may prevent generalization, and hurt monolingual accuracy because of the resulting loss of knowledge, but in cross-lingual settings it can also help for distinguishing the rule from the language-specific exceptions, and thereby enable better transfer with simpler class boundaries.

Furthermore, Section 6.5 exhibits cases where learning relevant knowledge can only be done by first learning some other pieces of knowledge, independently of their relevance. Indeed, some easy classes (typically, determiners) act as building blocks for others (typically, nouns): not having access to them can delay or event prevent learning of other classes (§6.5.3). For instance, even if the target does not have determiners, cross-lingual parsers must be able to attach them in source data, because nouns must first be singled out (without determiners to pollute their neighbourhood) before learning their attachment to verbs (SHARED needs SOURCE-SPECIFIC). Conversely, regarding the nouns whose attachment is specific to the target, even if determiners are cross-lingually consistent, the parser trained specifically on target data must also be able to attach them, or again it could not learn to attach the nouns (TARGET-SPECIFIC needs SHARED).

Keeping these dependencies can be crucial, because they affect the *representations* of the configurations evaluated by the parser. For instance on the French sentence '*cet appartement minuscule me suffit*' (literally, *this flat tiny to-me suffices*, corresponding to PoS sequence 'DET NOUN ADJ PRON VERB'), the PRON$\curvearrowleft$VERB dependency is theoretically scored with only 'NOUN' on the stack (which can help to detect that the VERB has already a candidate subject, so that the PRON will be annotated as `iobj`); but if the parser is not supposed to attach ADJs, even in gold space the stack can be 'NOUN ADJ' at that time, so that the parser may see only the ADJ and thus decide that the PRON is `nsubj`.

However, when the focus is put on a handful of classes, incorporating more knowledge to learn within the same model can have an adverse impact on the ability to extract knowledge from the desired dependencies. Feature-level analysis reveals that including some simple classes makes others much more complex, by preventing their parameters from converging (§6.5.2). But this effect cannot be alleviated by timing tricks, considering that it results from an objective function that combines two different purposes at the same time and thus advocates for an under-optimal trade-off.[47]

Thus, I rather propose to simplify the task assigned to the parser: thanks to simpler class boundaries, deleting one class can have a large beneficial impact on competing classes, even despite data reduction (§6.5.1), and a few manipulations at the word level (to reduce their contribution to the objective function) can yield huge improvements on specific classes whose learning seemed impossible until then (§6.5.3). So, even if the parser needs access to a given piece of knowledge acting as building block, it is better to just provide it than to make the parser learn it altogether. Providing it as initialization parameters would fail, because it would be vulnerable to pervasive third-party updates (§6.5.3). One way to preserve that knowledge is to provide it *at the data level* instead of the parameter level. For my transfer framework, it means that the prediction is split between several submodels, each one trained separately, and each one taking over the annotation of some tokens in the sentence (and forwarding the information to others).

As a matter of fact, the literature already has a meta-learning framework for this

[47]This statement appears in contradiction with the success of multi-task or multi-view learning. However, such frameworks assume some kind of covariate shift: they consider two different ways to learn the *same* knowledge, while here it is about learning two *unrelated* pieces of knowledge on the same data.

setting: *cascading* (Alpaydin and Kaynak, 1998).[48] The idea here is to train separately (but not independently) a series of partial parsers, each one in turn, and to enrich the input at each step with their predictions, in order to incrementally build the full parse – that is, without reannotating previously annotated words, which would rather be *stacking*. The first submodel would typically be easy shared classes, presumably used as building blocks by all subsequent models, then would follow cross-lingual parsers from various sources (and based on various transfer techniques), so that the last submodel could focus on extracting from sparse target data what could not be recovered yet, i.e. target-specific knowledge. For a sentence like '*Her boyfriend broke up on February 14*', the first step would be to attach '*Her*' and '*on*' (DET$^{\curvearrowleft}$ and ADP$^{\curvearrowleft}$ are easy), after which a parser projected from a close language like French would take care of the main structure of the sentence (identifying the `root` '*broke*' and its `nsubj` and `obl` dependents, '*boyfriend*' and '*February*'); the verbal postposition ('*up*'), which is a property of Germanic languages, could then be attached by a delexicalized parser trained on, say, German, and the very English-specific attachment of '*14*' would finally be completed by an English parser.

Now that its specifications have been established, the remaining chapters describe step by step the development of this framework.

[48]Interestingly, the method was primarily presented as 'learning a rule and exceptions' (Alpaydin, 1997), which has precisely been one major focus in this chapter.

Part III

A new framework for cross-lingual transfer

Abstract

In this part, practical steps are taken to implement the cascading transfer framework designed in the previous chapters, which combines seamlessly multiple sources and resources. First, the parsing framework is extended to reach a higher degree of flexibility regarding inputs and outputs, needed to perform cascading in practice. The whole extension is based on a generalization of dynamic oracles; it leads to the release of the PanParser software, whose main functionalities are described and empirically evaluated. Second, typological knowledge is used to address some word order divergences between the source and target languages, which often hinder effective transfer. This proposal improves the applicability of the proposed transfer approach, by making even distant sources worth exploiting. Finally, as the cascading framework is parameterized by a series of metrics and architectural choices, a set of those is proposed to enable its effective use in a cross-lingual context, but also for monolingual applications. All contributions of this part are evaluated in the frame of a shared task, with realistic low-resource conditions.

Chapter 7

Developing a more flexible parsing system

***

Contents

In Chapters 5 and 6, I have designed and described the main ideas of a transfer framework with desirable properties, based on a cascading architecture. However, that framework is impossible to apply with state-of-the-art transition-based dependency parsers. Indeed, in the literature, the parsing task is mostly considered from the perspective of its standard use cases: given a treebank, how to train a parser? And given raw data, how to produce a tree for each sentence? No need to produce partial trees, for instance, has arisen, and as such, the main algorithms of this literature have been developed with a focus on the standard use cases; so, even if departing from some aspects of those conditions is just a matter of implementation, the possibility has not necessarily been included in existing implementations. However, my cascading proposal implies precisely to stray from the standard parsing conditions, therefore I need to extend the framework of transition-based parsing to accept much more diverse settings; the behaviour of the parsers has also to be analyzed in this new context, in order to check that they perform well even then.

This chapter is consequently a necessary preliminary step before actually tackling cascading transfer, where I propose ways to make parsing frameworks more flexible, and I put those ideas into practice by releasing my own parser implementation: PanParser.[1]

Section 7.1 describes a few non-standard parsing tasks which are required by my cascading framework, but out of the scope of state-of-the-art parsing; I then provide hints at how dynamic oracles can be used to reach this higher level of flexibility. In Section 7.2, I propose several extensions of the dynamic oracle framework, in order both to increase its applicability and to make it accessible to beam parsers: these contributions allow me to sketch a unified framework which reconciles various parsing approaches and opens new research perspectives.

PanParser is built upon those ideas, as explained in Section 7.3, and it includes the new functionalities I advocate, among others. The rest of the chapter addresses more closely three of those functionalities (partial training in Section 7.4, partial parsing in Section 7.5 and constrained parsing in Section 7.6), providing additional implementation details, but also empirical insights on their accuracy and behavioural properties.

7.1 An increased need for flexibility

7.1.1 Non-standard parsing tasks

In contrary to the standard use cases, the transfer framework sketched in previous chapters involves *parts* of parsing models or data. Putting this framework into practice thus assumes access to or implementation of several non-standard parsing tasks based on partial trees; most of them however are not possible with the state-of-the-art algorithms and implementations, which are tailored to the standard use cases of parsing.

Partial training In cross-lingual settings, the training data may contain tokens which are unannotated by default: in this work it concerns some dependencies considered as irrelevant for the target syntax (for instance source-specific exceptions) but it can also happen for other reasons, like unaligned tokens during annotation projection, as mentioned in Section 4.1. To properly handle such tokens, the parser must then be able to *train on arbitrarily partial annotations*.

[1]`https://perso.limsi.fr/aufrant`

Partial parsers While partial training does not affect the prediction task (the parser is still supposed to annotate any input token, without restriction), the previous chapter has also suggested to build parsers with a more specific task (§6.5.2), i.e. which are supposed to learn only part of the syntax. For instance, a model specialized on function words (used early in the cascade, to help all subsequent models which focus on content words) should learn to reject content words, i.e. leaving them unannotated. Such *partial predictions* are not possible in the state of the art: the only way to achieve this currently is by performing a full annotation, and then filtering the output; but this strategy has no impact on the classifier and consequently misses the potential beneficial effects mentioned in Section 6.6, whereby the decision boundaries can be simplified at training time. Besides, such filtering is only possible when the typology of unannotated tokens is well known, which is not the case when they result from partial projection or confidence filtering. For instance if, for any reason, training data turns out to have annotations for only 10% of ADPs, it is not clear whether ADPs annotations should be filtered out from the output, or if the parser has enough material to learn them accurately. Using a partial parser in that case lets the parser decide *itself* whether that few occurrences are enough to generalize on ADP attachments, or it is safer to refrain from annotating them.

Constrained parsing In the cascading architecture I propose, information flows from one submodel to the other by means of enriching the input with the already predicted dependencies. These annotations act as constraints, in the sense that they must then be included in the output. Hence, to build my transfer framework as described, the underlying parser needs the ability to *parse under constraints*, at prediction time, which a few implementations have in fact started to allow, but also *at training time*, which to my knowledge is unprecedented. Indeed, if all determiners are preannotated by another parser, the current parser does not need to learn how to attach them (thus redundantly acquiring previous knowledge), because it will never actually annotate a determiner. Providing the constraints at training time thus signals what the parser can avoid to learn, letting it focus on how to complete them.

Besides, there are also issues of train-test consistency, whereby the model performs best if it is trained in conditions similar to those it encounters at prediction time,[2] because it has been optimized to discriminate hypotheses in a specific search space, which should not be tampered with. So, if parsing is constrained at prediction time, it should be at training time as well.[3]

[2]This effect is illustrated in Chapter 3 by the score drop when beam size changes between training and test.

[3]In Pécheux et al. (2015)'s experiments, constraining the search space is beneficial at prediction time, but detrimental at training time: indeed, the constrained tokens do not contribute anymore to learning general knowledge, from which unconstrained tokens would have benefited though. However, in the context of this work, it is assumed that the constrained and unconstrained tokens are typologically so different that it is not possible to generalize from the former (e.g. DET-NOUN dependencies) to the latter (e.g. NOUN-VERB dependencies). This supposes that this distinction has been appropriately estimated, which is addressed in Chapter 9.

On a final note, applying the constraints also at training time has the extra benefit that the information they contain can be explicitly exploited (as features), instead of specifying mandatory edges only. Typically, using ArcEager, when considering a Left action (i.e. creating the dependency $s_0 \curvearrowleft n_0$, where s_0 and n_0 are the first elements of the stack and the buffer), n_0's head and right children are always unknown by design, because they are not attached yet; but if the information is already available in the constraints, it could be drawn from there. This can change for instance the attachment of token 'the', for the input 'the temple $\curvearrowleft$ bombings': the feature 'n_0 is a NOUN and its head is a NOUN on the right' indicates a compound noun, in which case the determiner should not attach to the first noun (Left action) but to the second (Shift action). If a previous parser handles (reliably) all compounds, then this feature can be used very straightforwardly to attach 'the' to different nouns in sentences '*The temple bombings surprised everyone*' and '*The temple bombings destroyed has been rebuilt*'.

Unrestricted data Finally, several parsing algorithms require the training data to have some formal properties, typically projectivity.[4] But since the data resulting from transfer is particularly noisy, it will come with much less guarantees: for instance, when projecting a projective tree through parallel data, the projected tree can still be non-projective, because of word order differences or noisy alignments. It can thus concern much more sentences than in the standard monolingual setting, so that the baseline strategy (discarding all inadequate examples) is riskier, even more so in a resource-sensitive scenario. Modifying the trees to make them projective is similarly delicate, because it adds noise to already noisy annotations. Consequently, to better leverage the scarce available data, the parser should be able to *train on any example*, without any restriction on their properties.[5] The same goes for constraints: depending on how they were built, they may be non-projective for instance, but even a projective parser should be able to exploit them.

7.1.2 Additional benefits of dynamic oracles

The principle of dynamic oracles satisfies most of these needs, for the only reason that they approach the whole computation differently. By breaking away from the idea of reference derivation, this approach also departs from a narrow viewpoint on oracles, whereby the only purpose of computing them is to produce reference

[4]Projectivity is the most straightforward example of such property, but the parser of Gómez-Rodríguez and Nivre (2010), for instance, requires the trees to be 2-planar, and other transition systems may expect other properties. Another example is having full coverage over the sentence, i.e. a single complete tree. That one is addressed separately (as partial training), but it serves the same purpose of making all examples usable.

[5]There remains however the requirement that annotations must always be compatible with trees, i.e. single-headed and acyclic; extensions to directed acyclic graphs would be an interesting track for future work, though.

configurations for updates during training. As a matter of fact, they can be used in many other ways.

Dynamic oracles reduce most of the oracle computation to some mathematical property of the configuration, the action cost (since oracle actions are defined as zero-cost actions). By doing so, this approach also singles out the actual computation (cost evaluation) from the context of this computation (tree properties, constraints, how the parser got to that configuration) and its purpose (training). As a matter of fact, action cost can always be computed for any (legal) action, in any parser configuration: basically it just states, for a given set of edges, whether the considered action is compatible with those edges (i.e. does not remove any of them from the reachable ones), regardless of any other consideration. Yet, being able to answer that question is exactly what the non-standard tasks listed above require.

Partial training This kind of training is in fact enabled by default as soon as dynamic oracles are employed: because updates are error-based, when there is not enough information to flag an error (because the attachment that could be troublesome is unknown), nothing happens and training simply goes on. The main difficulty in this regard is actually a matter of implementation: how partial annotations are encoded, and how action costs are specified without assuming that the involved heads are always properly identified tokens.

Constrained parsing Constraining the parser to add specific dependencies to the output can be thought of as just ensuring that they belong to the final parse, and this can be done by preventing the parser from considering any action that is incompatible with the dependency constraints. In other words, constraints can be applied by computing action costs with respect to those constraints, and authorizing only zero-cost actions to be scored. Then the parser will naturally apply only actions which respect the constraints, and include them in the final output. Training under constraints is just as easy, because the search space can be reduced in the same way at prediction and training times.

Unrestricted data The ability to train on any kind of data also results from the use of dynamic oracles: the cost is well-defined for any action, and by definition there is always at least one zero-cost action. So, all data are usable by design, regardless of non-projectivity or any other unwanted property. The issue rather resides in deriving a sound definition of the cost in certain cases. Regarding non-projective examples specifically, one way to look at them is as a set of configurations containing arc incompatibilities: when two crossing edges are reachable, only one can actually belong to the final output. Yet, this is a known setting: in non-arc-decomposable systems, from certain erroneous configurations it is not possible to reach all reachable dependencies within the same path. With appropriate cost derivation, it is thus possible to easily handle such cases.

Partial parsers are the only extension whose implementation does not rely on dynamic oracles. They are actually handled differently because their output differs from standard full parses and as such is not allowed by the state-of-the-art transition systems. The design of such parsers is addressed separately, in Section 7.5.

7.1.3 Dynamic oracles for global training

Dynamic oracles consequently appear to be beneficial in many regards, and yet they are not available for beam parsers. Chapter 3 has already explained why beam parsing and dynamic oracles are complementary: one is about choosing the negative update configuration, the other is about choosing the positive one (which at the same time increases the number of legal candidates for negative configurations). There is consequently hope to achieve extra improvements by combining both approaches.

But in the present case, there are more pressing reasons to consider dynamic oracles for beam parsers. Three of the non-standard tasks presented above entirely rely on such an oracle, and yet it is only available for greedy parsers, which even with dynamic oracles remain less accurate on standard tasks than beam parsers. But if my improved transfer framework proves successful, restricting it to low-performance parsers will likely prevent it from actually outperforming state-of-the-art cross-lingual parsing, just because the other methods can be entertained with better parsers.

7.2 Extensions of dynamic oracles

This section presents mainly two contributions to improve dynamic oracles. I first propose (§7.2.1) a generalized definition of the action cost, which has many benefits, from the derivation of dynamic oracles in non-arc-decomposable cases, to the possibility to exploit any ill-typed data (like non-projective trees) without even deriving a new oracle; in the course of this study I also amend a previous literature result, showing that the ArcEager system is not always arc-decomposable. One of my publications is dedicated to this generalized criterion and its handling of non-projectivity (Aufrant et al., 2018a). Second, I present a sound extension of dynamic oracles to global training (§7.2.2), and conclude by using that extension to formalize a unified framework which covers the diversity of parser training procedures (§7.2.3), thereby solving two issues left open in the literature. That second contribution has lead to two publications (Aufrant et al., 2016c, 2017).

7.2.1 Redefining action cost to cover more ground: non-arc-decomposable systems and non-projective trees

I propose here alternate definitions of the action cost and the gold action criterion, which extend the applicability of dynamic oracles. Their benefits are exemplified here in two ways: by a case study on a non-arc-decomposable transition system, for which a dynamic oracle is easier to derive with the new definitions, and by showing in the case of non-projectivity how those definitions make it possible to learn on unrestricted data.

Following the definitions given in Chapter 3, the cost of action t in configuration c is the drop in maximum UAS incurred by this action:

$$\textsc{Cost}(c,t) = \left[\max_{t_1,\cdots,t_{final}} \text{UAS}(c \circ t_1 \circ \cdots \circ t_{final})\right] - \left[\max_{t_2,\cdots,t_{final}} \text{UAS}(c \circ t \circ t_2 \circ \cdots \circ t_{final})\right]$$

and a system is said *arc-decomposable* if all arcs reachable from a configuration can be reached jointly, by the same derivation. In other words, their predictions are compatible. Denoting $\textsc{ForbiddenArcs}(c,t)$ the number of gold arcs that are reachable from configuration c but not from $c \circ t$ (i.e. after applying action t), it ensues that:

$$\textsc{Cost}(c,t) = \textsc{ForbiddenArcs}(c,t)$$

Non-arc-decomposable systems When this property does not hold, on the other hand, there are extra sources of cost to account for, because of incompatible arcs. In case of such incompatibilities, at some point, adding a gold arc will indeed imply renouncing to another gold arc,[6] thereby inserting a new error. But this cost cannot be attributed to the given action, it is in fact due to a much earlier action, which introduced the incompatibility. Besides, sometimes in such cases, FORBIDDENARCS is non-zero for all legal actions, in which case it is obviously not identical to the COST function, which by definition is zero for at least one action.

There are two main strategies to compute the action cost in non-arc-decomposable systems. The first is to explicitly compute the loss before and after the action, typically using dynamic programming (Goldberg et al., 2014), and then retain the difference. The second is to directly model the cost, by formalizing the configurations holding arc incompatibilities, and detecting when such incompatibilities are inserted (Gómez-Rodríguez and Fernández-González, 2015). When possible, this is computationally cheaper than a full loss computation.

[6]For instance, with the ArcStandard system and the example 'The$\curvearrowleft$bag$\curvearrowleft$fell', in the configuration where the buffer is empty and the stack is [*The bag fell*], the only way to output the *bag*$\curvearrowleft$*fell* dependency is to attach (and pop) '*bag*' immediately, thereby renouncing to its child '*The*', even tough it is currently reachable.

Relaxed action cost To formalize the cost in a non-arc-decomposable system, I define $\text{EXPECTEDCOST}(c)$ as the number of gold arcs that are still reachable from c but (considering some final configuration, reachable from c and with maximal UAS) do not belong to the final output. This counts the number of current incompatibilities. The action cost then decomposes as:

$$\text{COST}(c,t) = \text{FORBIDDENARCS}(c,t) + (\text{EXPECTEDCOST}(c \circ t) - \text{EXPECTEDCOST}(c))$$

I now introduce the RELAXEDCOST function, defined as:

$$\text{RELAXEDCOST}(c,t) = \text{FORBIDDENARCS}(c,t) + \text{EXPECTEDCOST}(c \circ t)$$

from which ensues:

$$\text{EXPECTEDCOST}(c) = \text{RELAXEDCOST}(c,t) - \text{COST}(c,t) \leq \text{RELAXEDCOST}(c,t)$$

and because at least one action has zero cost:

$$\begin{aligned} \text{EXPECTEDCOST}(c) &= \min_t \text{RELAXEDCOST}(c,t) \\ \text{RELAXEDCOST}(c,t) &= \text{FORBIDDENARCS}(c,t) + \min_{t'} \text{RELAXEDCOST}(c \circ t, t') \\ \text{COST}(c,t) &= \text{RELAXEDCOST}(c,t) - \min_{t'} \text{RELAXEDCOST}(c,t') \end{aligned}$$

In other words, the RELAXEDCOST function computes an overestimate of COST, that repeatedly counts the cost of incompatibilities, as long as they are not resolved, and not only when they are introduced. Thus, it may happen that no action has a zero RELAXEDCOST, but the actual cost can be retrieved by shifting all costs by the minimum RELAXEDCOST, which corresponds to the current EXPECTEDCOST. Hence, in this framework, the optimal actions are not those with zero cost but with minimal cost.

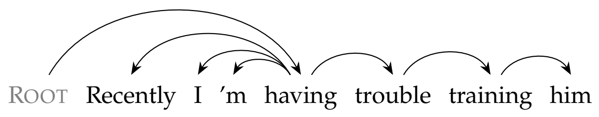
These definitions have two useful properties, which make the use of the alternate definition transparent. First, for arc-decomposable systems, EXPECTEDCOST is null, so $\text{RELAXEDCOST} = \text{COST}$. Second, since $\min_t \text{COST}(c,t) = 0$, in both cases (RELAXEDCOST and COST), shifting by the minimum cost always yields COST values.

Consequently, from now on, I use RELAXEDCOST instead of COST and define optimal actions as *minimum-cost actions*. In practice, defining the action cost explicitly then consists in listing as usual the arcs that the action makes unreachable, as well as the causes of arc incompatibilities in the future configuration. The difficulty of deriving non-arc-decomposable costs now resides in the exhaustive enumeration of arc incompatibilities.

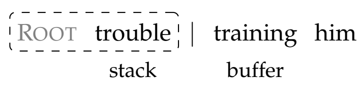
ArcEager with ROOT in first position I first illustrate the benefits of these definitions on the variant of the ARCEAGER system where the dummy ROOT is at the beginning, and for which no dynamic oracle has been derived yet.[7]

Indeed, moving the ROOT token to the beginning of the sentence has important consequences for that system. First, it changes the preconditions of some actions: SHIFT becomes illegal when the buffer contains a single element, because afterwards it cannot be attached, and the final configuration with ROOT token requires an empty stack. Similarly, RIGHT is illegal when the buffer contains a single element and at least one word in the stack is unattached.

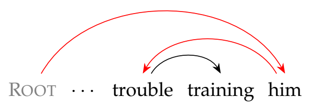
Second, and as a direct result of those new preconditions, the system becomes non-arc-decomposable. Indeed, in the configuration pictured by Figure 7.1, both heads of 'training' and 'him' are reachable, but the only way to get both of them (with a RIGHT-RIGHT sequence) is forbidden by the extra precondition, because then the parser would end up stuck in a non-final configuration.



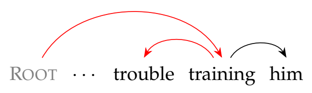
(a) Reference parse tree.



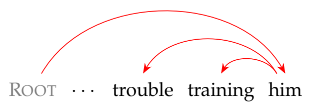
(b) Stack and buffer of the (already suboptimal) configuration to evaluate; 'trouble' is unattached. Possible actions are RIGHT, LEFT and SHIFT.



(c) Case RIGHT: best parse with REDUCE-LEFT-RIGHT-REDUCE afterwards.



(d) Case LEFT: best parse with RIGHT-RIGHT-REDUCE-REDUCE afterwards.



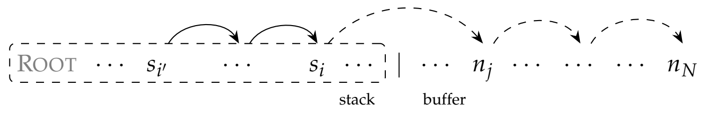
(e) Case SHIFT: afterwards LEFT-LEFT-RIGHT-REDUCE is enforced.

FIGURE 7.1: Counter-example to arc-decomposability of the ARCEAGER transition system with ROOT in first position.

Hopefully, this property does not hinder the derivation of a dynamic oracle for ARCEAGER in the general case: as advocated above, the usual definition (Table 3.12) is simply extended by listing the incompatibilities. In fact, there is exactly one type of configuration which implies arc incompatibilities, illustrated in Figure 7.2. It is the case when the reference ascendance of the last word in sentence (n_N) consists in any number of buffer tokens followed by a stack element, and deeper in the stack (including n_N's ancestor) at least one word is still unattached. Because of the extra preconditions, in that case at least one ancestor of n_N will not get its correct head, but will instead be promoted as head of the unattached stack element.

Consequently, non-arc-decomposability adds a (relaxed) cost of 1 if the parser is in this configuration, or will be put in this configuration by the given action – the cost

[7]To the best of my knowledge, this topic is not addressed in the literature: Goldberg and Nivre (2013) only study the case where ROOT is in the end, as is common practice. There are reports however of actual uses of that variant, but with a static oracle (Ballesteros and Nivre, 2013).



ROOT ··· $s_{i'}$ ··· s_i ··· | ··· n_j ··· ··· ··· n_N
stack
buffer

FIGURE 7.2: Prototype of arc incompatibilities for ARCEAGER with ROOT in first position. Dashed dependencies belong to the reference, plain ones are those already predicted.

computation is otherwise unchanged. More precisely, a single arc incompatibility is inserted whenever the parser is *not* already in that configuration, and an ancestor of the last token is shifted, or attached to a stack containing at least one unattached word; this is a case where computing the relaxed cost is much more direct and efficient than computing the actual cost.

Non-projective examples As explained in Section 7.1, non-projective examples can simply be seen as configurations with arc incompatibilities. Hence, from the oracle's point of view, the initial empty configuration already comes with embedded 'past errors' (the incompatibilities due to edge crossings). As in non-arc-decomposable systems, the cost incurred by these incompatibilities is not due to actions to come, but should be attributed to previous actions, taken in a fictive history before the initial configuration. As such, the natural behavior of dynamic oracles is to ignore this cost.

However, in the frame of this work, I have not derived the action costs for non-projective examples (that is, enumerated all non-projective arc incompatibilities in an arbitrary parser configuration). By not specifying arc incompatibilities, the cost is simply under-estimated. This has the consequence that some actions introducing incompatibilities may be deemed correct, later resulting in configurations where all actions forbid some reference arc, even though no error has been detected in the past; but such cases are handled transparently when defining gold actions as *minimum-cost* instead of *zero-cost* actions. Thus, the minimum-cost criterion makes training on such examples at least possible, even though a few updates remain unsound.[8]

The benefits of including non-projective examples in the training sets of projective parsers are evaluated on UD 2.0, by exploiting them with the proposed oracle-based method, and also by making them projective with the pseudo-projectivization technique (Nivre and Nilsson, 2005). As shown in Table 7.1 (upper part), it is empirically better to handle non-projective sentences with minimum-cost dynamic oracles than to discard them all; but this strategy also outperforms pseudo-projectivization. As expected, the gains of both methods increase when the proportion of non-projectivity increases, i.e. when more examples would have been

[8]Under-estimated costs biase the oracle toward late resolution of inconsistencies: in case of two incompatible arcs, the parser will prefer actions that keep both options as long as possible (with no cost in terms of FORBIDDENARCS) over actions that forbid one of them right away (and thus yield a cost).

discarded. Apart from higher gains on average, the advantage of the minimum-cost strategy is that it is consistently beneficial, whereas pseudo-projectivization is detrimental for the smallest treebanks.[9] Those results are supported by additional experiments with standard projectivization in (Aufrant et al., 2018a).

		% non-projective sentences				# training sentences	
	μ	$> 50\%$	25-50%	10-25%	$< 10\%$	> 500	< 500
PANPARSER – greedy ARCEAGER	78.28	56.23	76.22	75.48	82.47	81.34	67.36
+ dynamic oracle (only projective snt.)	78.94	57.74	76.98	76.25	82.96	81.92	68.34
+ dynamic oracle + pseudo-proj. snt.	+0.26	+2.01	+1.49	+0.20	-0.07	+0.46	-0.46
+ dynamic oracle + non-projective snt.	+0.48	+2.45	+1.83	+0.45	+0.08	+0.51	+0.36
PANPARSER – greedy ARCHYBRID	75.70	53.08	73.66	73.19	79.63	78.29	66.50
+ dynamic oracle (only projective snt.)	76.50	54.22	74.61	73.95	80.40	79.22	66.81
+ dynamic oracle + non-projective snt.	+0.55	+3.08	+2.16	+0.34	+0.22	+0.53	+0.61
MALTPARSER (only projective snt.)	72.88	57.87	71.74	69.99	76.68	76.81	58.87
+ pseudo-projectivized sentences	+0.37	+5.84	+1.40	+0.19	+0.07	+0.48	-0.02
+ pseudo-proj. + deprojectivized output	+0.45	+6.84	+1.69	+0.25	+0.09	+0.59	-0.05

TABLE 7.1: Comparison on UD 2.0 of various strategies to handle non-projective training examples, depending on the non-projectivity rate and on treebank size. Reported scores are the average UAS over the corresponding sets of languages. All UAS gains are computed with respect to the 'only projective' lines. MALTPARSER is version 1.9.1 with default parameters, and pseudo-projectivization uses the `head` encoding scheme of the MALTPARSER implementation.

Additional measures with the ARCHYBRID system show that the gains achieved by the minimum-cost criterion are not specific to the ARCEAGER system: despite different baseline scores, the proposed strategy yields similar improvements.

For illustrative purposes, similar experiments are conducted with MALTPARSER (Nivre et al., 2006), which is another implementation of the ARCEAGER system but differs from PANPARSER in several ways (different feature templates, static oracle); to help comparison, additional results are reported for PANPARSER without dynamic oracles. Compared to MALTPARSER, the ARCEAGER PANPARSER baseline appears much stronger (+6 UAS) on the downsized datasets; but the gains achieved when exploiting the non-projective trees (with pseudo-projectivization) are similar in both implementations. There is one exception, Ancient Greek (the only treebank with more than 50% non-projective sentences), for which the MALTPARSER gains are way larger than those of PANPARSER; but this treebank seems particular in several regards[10] and consequently does not question the superiority of the minimum-cost oracle over the pseudo-projectivization strategy, measured even in Ancient Greek for PANPARSER.

Finally, Table 7.1 also reports the gains achieved by MALTPARSER when pseudo-projectivization is followed by deprojectivization of the output. Plain comparison

[9]A plausible explanation is that rewriting the trees introduces some inconsistencies in the training material, which are only alleviated when data is large enough.

[10]The baseline MALTPARSER already behaves differently on that language: it slightly outperforms the baseline PANPARSER instead of underperforming it by a large margin. The explanation may simply lie in other differences between implementations, for instance MALTPARSER's feature templates may be particularly suited to Ancient Greek.

of this line with the minimum-cost strategy is delicate, because it does not result from better training only, but also from a gain in expressivity: it is able to retrieve even non-projective dependencies. But it is interesting to see that deprojectivization only marginally improves over pseudo-projectivization alone: most of the gain actually resides in the treebank augmentation rather than in retrieving non-projective dependencies. Besides, the minimum-cost strategy outperforms even the deprojectivized results. This finding highlights the importance of making all data usable, which this new oracle definition enables.

7.2.2 Global dynamic oracles

Apart from Björkelund and Nivre (2015)'s experiment with non-deterministic oracles (see Section 3.6), the previous work that is closest to combining dynamic oracles with beam parsers is YaraParser (Rasooli and Tetreault, 2015). In this parser, the early-update strategy uses a reference derivation as usual, except that it is not precomputed, but built during parsing and tailored to beam states. In practice, the parser maintains a reference partial derivation and ensures that it stays in the beam: at each step a zero-cost action is appended to this reference, chosen among those kept in the beam (and if there is none, an error is flagged). As this approach relies on action costs, it clearly relates to dynamic oracles; however, it still requires building a reference explicitly, which in fact prevents the oracle from truly acknowledging all possible gold derivations.[11] The YaraParser strategy can consequently not be considered as a sound extension of dynamic oracles to beam parsing. If the conditions leading to missing a reference are rare enough, this approximate extension may be sufficient, though, which explains the competitive results achieved by YaraParser on standard parsing tasks. But high spurious ambiguity, which notably occurs with partial training (because all supertrees of the provided annotations are considered correct), increases a lot the risk of incorrect updates, making the YaraParser approach unsuitable for my non-standard needs.

I thus turn to a point of view that does not involve any reference: instead of using action costs to build the set of gold actions (those with minimum cost) and then check if the predicted action is among them, I prefer to check directly whether the predicted action is correct in context, as a property of the action itself. For a greedy dynamic oracle, this alternate viewpoint has no practical implication. But the difference becomes significant for beam parsers: the idea is to check a property of a derivation instead of building a set of reference derivations, thus avoiding memory issues when considering all possible references.

[11]Indeed, in case the reference c^+ (still in the beam) has two zero-cost successors, $c^+ \circ t_1$ and $c^+ \circ t_2$, even if both are kept in the beam only one will be retained as reference (say, $c^+ \circ t_1$). Then if at the next step all successors of $c^+ \circ t_1$ are discarded but $c^+ \circ t_2$ is properly extended with a zero-cost action, an error will be flagged, even though the beam still contains an hypothesis leading to the reference tree.

Similarly to local dynamic oracles which deem incorrect the *actions* which introduce a new error into the final parse, global dynamic oracles should deem incorrect the *transition sequences* which introduce a new error into the final parse. Hence, given an initial configuration (not necessarily empty or gold), the correct configurations are those from which the maximum UAS is the same as the initial maximum.

The Boolean function that tests this condition, denoted $\textsc{Correct}(c'|c)$, can actually be efficiently computed using the $\textsc{Cost}$ function: a configuration c' is considered as $\textsc{Correct}$ in the context of a configuration c, if there exists a sequence of transitions $t_1, \ldots, t_n$ such that $c' = c \circ t_1 \circ \cdots \circ t_n$ and $\textsc{Cost}(c, t_1) = \textsc{Cost}(c \circ t_1, t_2) = \ldots = \textsc{Cost}(c \circ \cdots \circ t_{n-1}, t_n) = 0$. When using $\textsc{RelaxedCost}$, the criterion becomes that all those transitions have minimum cost in the corresponding configurations.

In other words, to check this property, the parser does not need to compute any reference derivation explicitly, it just has to check the cost of each action it performs, and to track the hypotheses that are still correct and those which are not.

Use in training strategies Global dynamic oracles, as defined by the $\textsc{Correct}$ function, can then be incorporated into both global training strategies presented in Chapter 3, early update and max-violation. The following describes the more salient aspects of the procedure; for the complete formalism, the reader is referred to Appendix A.

For early update the application is straightforward: the parser keeps track of the correctness of each hypothesis in the beam and whenever all of them have become incorrect (meaning that the last reference was just lost), it flags an error and updates toward one of the correct hypotheses that just got discarded.

The error criterion is the same for max-violation. But then the actual choice of update configurations is done by maximizing over the subsequent steps the score difference between the positive (reference) and negative (predicted) configurations. Unfortunately, to compute these scores, an explicit reference derivation has to be built, thereby hurting the premise of this oracle strategy. The positive score cannot be reasonably computed as the maximum over *all* references from there, because they can be arbitrarily numerous. I thus propose (but other strategies can be envisioned) to build a reduced set of references by performing beam search (restricted to gold actions) from there, starting with a beam populated by the correct hypotheses that just got discarded.[12] The score history of the predicted parse is then compared to the one-best reference in beam, a configuration pair is chosen and the update is performed.

[12] Alternate strategies performing beam search from the best-scored configuration among those hypotheses, or even from the initial configuration, may yield slightly different results. For the sake of future work, I provide both in the PanParser implementation, described in the next section.

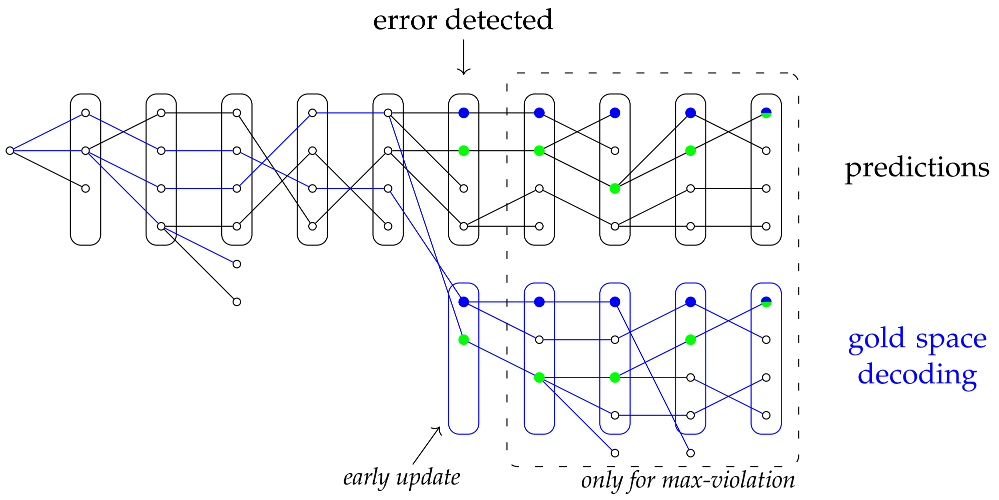


FIGURE 7.3: Evolution of beam hypotheses through time, and choice of update configurations. Edges denote actions, pictured in blue in case of gold actions. Rectangles represent successive contents of the beam (of size 4), ranked from top (one-best) to bottom (discarded hypotheses). For legibility, the hypotheses discarded from the beam are only pictured when they are correct.

It is worth noting that in the above descriptions, the positive update configurations are ill-defined. The alternatives are illustrated on Figure 7.3. For early update, one may indeed prefer the best-scored hypothesis among the correct discarded ones (blue configuration), in order to minimize the score difference to alleviate, or for instance the one which is historically closest to the predicted one (green configuration), in order to minimize the affected features. For max-violation, the score history of the predicted parse may be compared either to that of the one-best final reference (green configurations), or to the successive scores of the one-best reference in beam at each step (blue configurations), in which case the score difference is minimized over the set of references. One may also consider such an alternative regarding the predicted hypotheses. At first glance, I did not note significant performance differences between both, and I settled for arbitrary choices (one-best at each step, i.e. blue configurations in all cases), but the consequences of those choices may be worth studying in future work.

Oracle properties The global dynamic oracle as defined here is *non-deterministic*: when the parser produces any derivation leading to the reference, it is accepted and no update occurs.[13]

The oracle is also *complete* in the sense that it can be computed for any initial configuration, not necessarily gold or empty: by using the starting point of the beam as

[13]This property is not called into question in the max-violation case, even though it involves an explicit reference built arbitrarily: just as greedy updates have to choose an arbitrary gold action when there are two of them at the error point, for max-violation the beam has already strayed (after the error point) from the gold space, so all references beyond there are equally candidates and the choice is necessarily arbitrary.

context in the CORRECT function, the oracle acknowledges the past errors embedded in that configuration.

Consequently, this oracle, which is both non-deterministic and complete is verified to be *dynamic*. For now, however, only its non-determinism is really exploited, since the context configuration remains in the gold space; completeness is more involved in the following.

7.2.3 Framework unification

Chapter 3 has emphasized on the very different places that greedy and global training have in the literature, each one maintaining its own research track, and on the lack of a unified formalism to work jointly on both. I propose now to fill that gap. My approach to reunite them under the same training framework originates from a simple observation: greedy dynamic training performs a series of predictions and updates, while with global training, the early-update strategy performs a single search pass and then a single update.

However, having derived a complete oracle for global training, nothing prevents a beam parser from actually performing an interleaved series of search passes and updates: after a global update, the parser configuration is located in an arbitrary non-empty configuration, but it can simply restart training from there, using that configuration as context for the oracle. A formal description of the resulting training procedure is provided in Appendix A.

Figure 7.4 illustrates the searches and updates that this ‘restart’ strategy would result in for early update, and compares it to those of a greedy parser: from this standpoint, both paradigms now appear much more similar. And indeed, a careful comparison reveals that a greedy dynamic parser (with exploration) corresponds exactly to a beam parser with a beam of size 1, trained with early update and restarts on the negative configuration. A greedy static parser also corresponds to such a beam parser, but with restarts on the positive configuration. These observations achieve the unification of both paradigms.

Regarding the differences between early update and max-violation, my standpoint is that they both search for an update configuration pair beyond the error point: the negative candidates are collected by continuing parsing, and the positive candidates are built by performing gold-space search from the error point. The only difference then is that the max-violation criterion to choose the pair is based on score gap computations, while early update always selects the first pair of candidates – and as a result of this determinism, for obvious efficiency reasons, the subsequent candidates are not built.

To summarize, all four main strategies for training transition-based dependency parsers (greedy static, greedy dynamic, early update and max-violation) can be

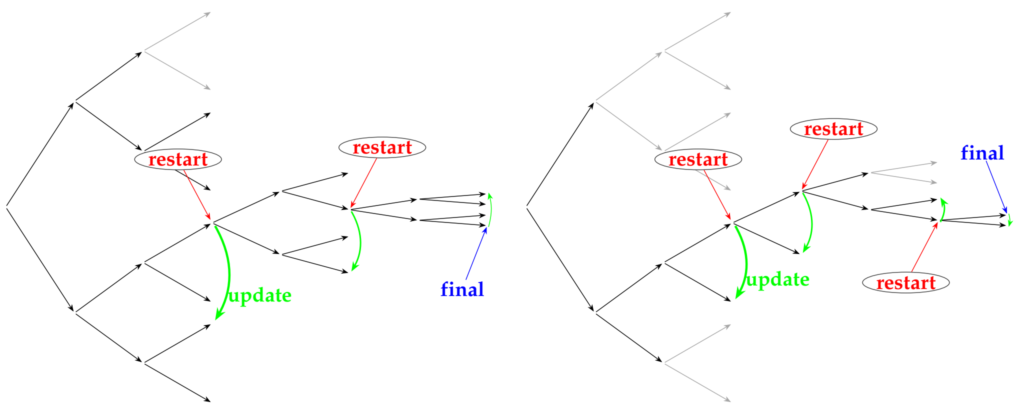


FIGURE 7.4: Outline schematic of the updates performed by beam and greedy parsers, and of the corresponding search spaces. The left-hand side pictures a beam of size 3, trained with early update and restart. The right-hand side corresponds to a greedy parser with dynamic oracle. Black edges are transitions scored by the parser, gray edges are those which belong to the search space but are not scored.

viewed as instances of the same training procedure. First, starting with an empty configuration, the beam is repeatedly expanded until an error is detected, but parsing may go on a bit further if necessary. Then some additional gold-space search is performed, and a pair of update configurations is chosen among those gold configurations and the configurations predicted beyond the error. Finally, the beam (and thus the oracle criterion) is reinitialized to either the positive or the negative configuration, and the process either stops there and moves on to the next example, or iterates until the parse is complete.[14]

This unified framework has two benefits: on one hand it simplifies reasoning on those training strategies, and opens new perspectives for code reuse when implementing several of them; on the other hand, by allowing combinations of characteristics from both paradigms, it unveils a number of new training strategies that have never been considered in the literature. For instance, how would a 1-sized beam behave when trained with early update but without restart (that is, greedy training limited to the first error)? A 1-sized beam with max-violation strategy? Or a k-beam with restart? As a side application, I have experimented with that last strategy, and shown that it has benefits in terms of accuracy and training time, as described in Appendix A.

[14]Most ideas presented here echo strongly the LaSO framework (Daumé III and Marcu, 2005). Indeed, they flag an error whenever no derivation hypothesis in the queue can lead to the reference output anymore, which is exactly the same criterion as my dynamic early update. However, they provide no practical realization of their ideas for transition-based parsing, relying on a monotonicity assumption on the task, whereby it is always possible to tell if the reference is reachable. This is precisely where dynamic oracles come in. Besides, their unified framework also contains the idea of restarting in global training, but as a reinitialization strategy that corrects past errors and comes back to the gold space. They consequently only learn in the gold space, and cannot accept past errors and move on to train in the suboptimal space, in contrary to my own framework.

In this section, I have pushed the applicability and benefits of dynamic oracles to a further level. First, I have extended their use cases, by carefully redefining their relation to action costs, in order to make all data always usable (even with projectivity issues), and also to simplify the oracle derivation for some non-arc-decomposable systems. I have then formalized dynamic oracles in the case of global training, which puts a final end to the lack of combination of both research tracks, as repeatedly noted in the literature. Finally, I have rationalized the training procedures of the four main training strategies to gather them under the same generic framework, thereby opening new research perspectives.

7.3 PanParser: a flexible parser with a modular architecture

In the course of applying my work to parsing, I have come to develop and implement a new transition-based dependency parser. This software, called PanParser, is described in further details in a technical report (Aufrant and Wisniewski, 2016), but here is a presentation of its main features and design choices.

The motivation of this implementation work is threefold: first of all to provide an implementation of the non-standard tasks presented in Section 7.1; to make fair benchmarking easier by reimplementing all algorithmic variants in the same way;[15] and to be modular enough to allow combining strategies in many ways without additional implementation work.

Thus, two principles govern the design of this software. The first is that the whole software is based on dynamic oracles and beam search, which provide the best genericity: greedy parsers are implemented as beams of size 1, as explained in the previous section, and static oracles are simulated using dynamic oracles, by precomputing a reference derivation with a gold-space search on arbitrary scores. The second is that all components of the parser are abstracted away to their most generic interface, so that they can be easily replaced and combined.

The main component is STRUCTUREDPREDICTION, which is in fact not specific to parsing, and is also used for instance for PoS tagging. It governs the search and learn procedures, that is, it contains all search logic (beam expansions) and instantiates the training framework I have described with the desired strategy, including the oracle (static or dynamic), the update strategy (early update and max-violation, with variants), optional restarts and optional exploration. The classifier is a separate component, with which the STRUCTUREDPREDICTION component interacts in

[15]Using different implementations raises the risk that they use slightly different feature templates, among others. Measures reported by Table 7.1 are an illustration of the difficulty to ascertain which effects are due to the actual proposal, or to differences in implementation: even though MaltParser and PanParser both use the ArcEager algorithm, it would not have been fair to compare directly the oracle-based strategy of PanParser with the scores that MaltParser achieves with pseudo-projectivization.

a double-blind way. During the search and learn procedures, STRUCTUREDPREDICTION manipulates objects representing parser configurations, in which is in fact embedded the whole logic of the transition system (including its semantics, preconditions and action costs) so that it does not affect the rest of the parser, and which also abstracts away feature extraction. Then again, feature templates are handled separately. Finally, a last part of the software (representing the actual parser) takes care of auxiliary tasks, like creating the empty configurations, extracting global feature vectors to actual perform the updates, formatting the outputs, etc.

PanParser contains a large number of transition systems: ArcEager, ArcHybrid, ArcStandard, but also the non-monotonic variant of ArcEager (Honnibal et al., 2013), two transition systems modifying ArcEager and ArcHybrid to allow partial predictions (described in Section 7.5), as well as variants of those, specialized in short-range dependencies. The implementation also enables variants of all those systems, with the ROOT token moved to first (or last) position. Regarding feature templates, I mostly provide variants of Zhang and Nivre (2011)'s templates, using either coarse or fine PoS tags, optional morphological features, and with the possibility to delexicalize the feature representation. As for the classifier component, most of my work uses the averaged perceptron, but PanParser also includes proofs of concept showing how neural networks can fit in that design (in the case of tagging) and how a joint classifier can be used (here, for relation labeling). The software is also shipped with a large number of experimental utilities, which I have used in many places throughout this work.

Coming back to the non-standard parsing tasks of Section 7.1, all of them are incorporated at the core of that implementation: partial parsers are implemented by dedicated transition systems, partial training and unrestricted data are enabled by design thanks to the improved dynamic oracles, and the STRUCTUREDPREDICTION component handles constrained parsing by applying additional filtering of actions during search.

As a natural continuation of that implementation effort, PanParser is the parser I use by default in the rest of this work. Table 7.2 reports accuracies measured with PanParser for the main few settings: it appears competitive with other state-of-the-art parsers, all of them being outperformed by an 8-sized beam PanParser.

7.4 Partial training: an application to cross-lingual transfer

This section illustrates the strength of partial training, with a straightforward application in the context of cross-lingual transfer.

My contribution is part of a joint work on a larger project, whose focus was on annotation projection, which as explained in Section 4.1 produces a number of partial trees that cannot be used directly to train a parser. The proposal of that work was

System	ROOT position	Greedy	Greedy dynamic	Early update	Max-violation
ArcEager	First	77.89	78.97	80.29	80.36
	Last	78.63	79.43	80.35	80.40
ArcHybrid	First	75.72	76.54	79.39	79.78
	Last	76.02	77.05	79.70	79.86
MaltParser	72.88				
MSTParser	79.52				
UDPipe	79.47				

TABLE 7.2: Average UAS achieved by PanParser on UD 2.0 with various strategies, compared to several state-of-the-art parsers. 'Greedy' results are computed with a static oracle, but for fair comparison the non-projective examples are also exploited (using a precomputed reference approximated by a dynamic oracle). 'Greedy dynamic' chooses exploration after each update. The 'Early update' and 'Max-violation' strategies use global dynamic oracles without restart. For fair comparison, UDPipe is trained without pre-trained embeddings, which would have significantly increased the available information.

to take advantage of dynamic oracles to handle them with a simple strategy based on partial training. The first results were promising, but unfortunately they fell behind the state of the art, because of the restriction to greedy parsers. I thus contributed to the success of that project by developing global dynamic oracles, which enabled the use of this transfer method with beam parsers. The resulting transfer system achieved much more competitive results, leading to joint publications (Lacroix et al., 2016b; Lacroix et al., 2016a). The method and experiments are briefly presented in the following, and the reader is referred to these publications for a more detailed description.

Transfer method After the source side has been annotated, the transfer procedure consists in three steps: partial projection, sentence filtering and partial training.

In this context, the only reason why projection is partial is the numerous unaligned tokens: some of them result directly from the predicted alignments, others are due to the symmetrization heuristic,[16] but their number is also increased by PoS-consistency filtering.[17] The claim underlying those choices is that a parser trained on high-quality (albeit partial) parses will yield a better accuracy than one trained on full but noisy annotations.

Sentence filtering is mandatory in our experiments, considering that for large parallel corpora, the size of projected data is huge for parsing (over 1 million sentences for the language pairs we experimented with); but for much smaller parallel

[16] In order to avoid multiple alignments and the resulting conflicts, the symmetrization heuristic is set here to 'intersection', in other words retaining only one-to-one alignment links predicted in both directions. This choice mechanically results in less alignment links, and consequently more unaligned tokens, both in source and target.

[17] As advocated by Rasooli and Collins (2015), we discard all alignment links between words with different PoS tags.

corpora, it is also a way to improve the overall quality of the training material. The criterion to retain sentences of highest quality is inspired by Rasooli and Collins (2015)'s work: we only keep sentences where (a) more than 80% of tokens are annotated (in order to limit the sparsity, and thus maximize the knowledge embedded in those sentences), and (b) the partial tree does not contain any non-projective pattern.[18]

The last step is partial training with the resulting (partial) data, which here means just training a dynamic oracle parser as usual on the projected trees.

Alternatives for partial training As mentioned in Chapter 4, other strategies have been entertained to handle the partial trees resulting from parse projection. Notably, Spreyer and Kuhn (2009) achieve partial training by adding fake root dependencies and then discarding configurations corresponding to fake root attachments. However, in stack-based parsing, resorting to such fake dependencies has some unavoidable side effects: some correct derivations (that is, leading to supertrees of the reference) will not be accepted.[19] In this regard, our approach is preferable, as it benefits from the soundness guarantees of dynamic oracles. As for Li et al. (2014), their approach to exploit partial trees is also quite similar to ours, but their method is designed for graph-based parsing and does not apply directly to transition-based systems.

From a practical standpoint, the oracle-based approach has a clear advantage in the fact that it does not require any specific refinement of the training procedure (at least, as long as the parser uses a dynamic oracle, which is recommended anyway), while both methods of Spreyer and Kuhn (2009) and Li et al. (2014) involve *ad hoc* modifications of the loss functions.

Experiments The method is evaluated on transfer from English to five languages (German, Spanish, French, Italian, Swedish), using version 2.0 of UDT (McDonald et al., 2013) and parallel data is drawn from Europarl (Koehn, 2005). As in related works, gold PoS tags are used at evaluation time.

Table 7.3 reports the UAS achieved by our transfer method, and compares them with the results published on the same data, for three state-of-the-art methods presented in Chapter 4 and which also exploit parallel corpora: M11 (McDonald et al.,

[18]My work on non-projectivity is posterior to that project, so this choice was natural, but it would be interesting in future work to try keeping those non-projective sentences: on the one hand they are more likely to result from misplaced attachments or translational divergences (but the PoS-consistency requirement may be sufficient to avoid those), on the other hand some syntactic structures really involve non-projective dependencies and would be impossible to learn with that filtering.

[19]Indeed, the position of fake roots in the tree affects not only the attachment configurations, but also the moment they are shifted.

2011),[20] MX14 (Ma and Xia, 2014) and RC15 (Rasooli and Collins, 2015). Among those methods, the closest to ours is RC15, which is the only one to train directly on partial trees, in their case by completing them using a parser bootstrapped on complete trees. '100%' columns indicate the scores achieved when training only on the subset of sentences which are fully annotated (for RC15, this is the boostrap), while 'partial' denotes the full method. The last column (sup.) indicates the supervised upperbound, computed with PanParser.

	M11	MX14	RC15		ours		sup.
Target			partial	100%	partial	100%	
de	69.77	74.30	74.32	70.56	73.40	69.36	84.43
es	73.22	75.53	78.17	75.69	77.05	73.98	85.51
fr	74.75	76.53	79.91	77.03	77.44	75.89	85.81
it	76.08	77.74	79.46	77.35	77.74	75.50	86.97
sv	75.87	79.27	82.11	78.68	82.13	77.26	87.89

TABLE 7.3: Comparison of the partial projection strategy with several state-of-the-art transfer methods.

Overall, our approach outperforms significantly both M11 and MX14 methods. Comparison with the RC15 method is particularly interesting, all the more because we fail to outperform it. However, it appears that this difference may not be due to our strategy to handle partial trees, but to the inherent quality of the parser: as illustrated by the '100%' results (with parsers trained roughly on the same sentences), even with beam parsing our base parser still underperforms theirs. There are indeed obvious differences between both settings: they use a beam of size 64 (8 for ours), trained with max-violation (early update for ours), and with Brown clusters (Brown et al., 1992) as additional features, which we do not use. The other major difference between both experiments (apart from the choice between keeping trees partial or completing them) is that Rasooli and Collins (2015)'s method is a 4-step procedure, where the accuracy is gradually increased by training a parser on more sparse data (completed by the previous parser) at each step; so, it is possible that our method would also benefit from a multi-step procedure, with self-training and data increase.

From a qualitative perspective, the main advantage of our method is that it achieves results which are very competitive with the state of the art, but at a much cheaper computational cost: compared to $O(n^4)$ computations in the case of Ma and Xia (2014) and the 4 successive parsers of Rasooli and Collins (2015), we only need to train a single (linear) parser.

[20]We noticed too late that the M11 results have been incorrectly reported from Ma and Xia (2014) to Rasooli and Collins (2015). Unfortunately, we reproduced the error in (Lacroix et al., 2016b), but it is corrected here: the numbers displayed in Table 7.3 are those computed and published by Ma and Xia (2014).

While additional efforts may be deployed to push that transfer method to even higher accuracies (with a more competitive parser, more careful filtering and multi-step training), our partial training approach has already shown its benefits: it is competitive in terms of accuracy, but also easy to implement, and computationally cheaper than the alternatives. This validates the oracle-based strategy.

7.5 Partial transition systems: learning to ignore

This section addresses another non-standard parsing task: partial prediction. The basic idea is to have the parser learn for which kinds of dependencies or tokens it should *not* predict any attachment, so as to reproduce the sparsity of the training data.

As explained in Section 7.1, the alternative is to perform a full prediction (using partial training if needed) and then filtering the output, but this is not a satisfying strategy. The advantage of partial prediction over the filtering approach is three-fold. First, Spreyer et al. (2010) has reported that training on partial trees can lead to outputting more dependencies of the kinds that are present in training data, compared to their frequency in that data: some probability mass is indeed lost by the unannotated tokens, which biases the distributions. Considering instead the non-attachment as a candidate label preserves the dependency distribution in training data and thus prevents against over- or undergeneration effects due to annotation biases. Second, it learns not only positive (actual attachments) but also negative facts (dependencies which were never observed in data), while a standard parser can guess any kind of attachment for the unannotated tokens and has no incentive to forbid any of them. Third, in the cases where the criterion for non-attachment is purely empirical, one may also use a third-party classifier to identify the dependencies to filter out, but it would lack the syntactic insight that the parser has when taking that decision itself.[21]

To achieve proper partial prediction, I thus propose to embed the non-attachment criterion at the core of the parser, by increasing its expressivity with slightly modified transition systems. For ArcEager, the change is just a matter of preconditions: those of the REDUCE action are relaxed to allow reducing even unattached tokens, so that a void attachment is naturally encoded by

[21]The following example illustrates those benefits. Suppose that in training data all NOUN$\curvearrowright$ADJ dependencies have been filtered out, as well as most NOUN attachments, except for NOUNs whose head is the right neighbour (e.g. 'Engineers$\curvearrowleft$solved that'). Then a standard parser will learn nothing about ADJs (no update ever occurred on an ADJ, so the model may not even know the corresponding feature), and learn to deterministically attach NOUNs to the following word. And when processing the input 'Engineers familiar with such issues solved that', it will produce the dependency 'Engineers$\curvearrowleft$familiar', which is obviously wrong but would be retained after filtering, as it attaches a NOUN to its right neighbour. Instead, a partial parser would learn that NOUN attachment is more complex than mere determinism, and truly learn the neighbour-attaching cases; it would not produce this dependency.

SHIFT+REDUCE. Adapting ArcHybrid (and similarly, ArcStandard) is less straightforward, because a 3-action system is not expressive enough to encode 3 types of attachment (left, right, void); so, it requires the addition of a fourth action, REDUCE, which again pops unattached tokens (which they all are, in those systems). With these modifications, the parser becomes able to produce any kind of full or partial (but still projective) tree.

Further modifications are required however to *learn* those partial predictions: the action cost must account for the additional possibility to leave a token unattached. For ArcEager[22] (and similarly for ArcHybrid), compared to the standard action cost (Table 3.12), new penalties are added in two cases: when a word whose reference attachment is void receives a head[23] (to enforce reproduction of non-attachments), and when an unattached word is reduced while its reference head is still somewhere accessible in the stack or buffer (and similarly for its reference children).

Experimental setup To evaluate partial parsers, I simulate partially annotated data in a monolingual setting (using UD 2.0 treebanks), and compare the accuracy of partial parsers trained on that data and non-partial parsers trained in various ways.

I experiment with three criteria for partial annotations: retaining only the classes which are easy for the average language,[24] and retaining only short-range dependencies, of length 1 (neighbours only) and at most 2.[25]

The first baseline is a standard parser, trained on full data, and whose output is then filtered according to the same criterion used for the partial parser. The purpose of this comparison is to ascertain whether the partial parser actually learns better when it has less knowledge to learn.

[22]Note that with a partial parser, ArcEager becomes arc-decomposable whatever the ROOT position is: with that extended *Reduce*, the stack can always be emptied, so that an empty buffer is not an issue and does not imply additional preconditions.

[23]This penalty goes against the property of non-partial parsers trained on partial data, which *ignore* the gaps and thus manage to extrapolate, from annotations on only some tokens, enough knowledge to cover the whole sentence. However, both approaches are fully compatible, provided that the implementation declares two types of unannotated tokens: those which should be handled by knowledge extracted from other tokens (e.g. to learn to attach all ADJs, but only based on ADJ dependencies projected with high confidence), and those which are typologically different from the annotated ones and should be predicted as unattached (if all predicative ADJs are unannotated in data, the parser should not over-generalize on its knowledge of epithets only and attach all ADJs to NOUNs). As such combination requires additional implementation work, I have not yet added that possibility in PanParser, though.

[24]I use the PoS/direction classes measured as easy in Chapter 6, using PanParser and the learning curve averaged over all languages considered then. The resulting classes are ADJ, ADP, AUX, DET, NUM, PART and PRON, all being preposed to their head.

[25]Learning only short-range dependencies is motivated by their high prevalence (40-50% and 60-70% according to measures in Chapter 6), but it also strongly echoes Eisner and Smith (2005)'s vine grammar, whose purpose is to train on tree fragments resulting from removing long dependencies. In their work, although it is based on a different parsing algorithm, they improved the F1-score on both English and Chinese (but not German) by applying a maximum length constraint during both training and evaluation.

However, when testing on the same language, it is likely that the dependencies filtered out actually share the same mechanisms as the retained ones, and that they end up helping prediction of the others: for instance with the length-1 criterion, the standard parser will see more DET-NOUN dependencies during training, because it encounters 'DET NOUN' phrases (as the partial parser does), but also 'DET ADJ NOUN' phrases with annotations on DET (which the partial parser gets unannotated). The standard parser will consequently be better informed, because the extra dependencies are, in fact, relevant for the test data. So, to simulate a case where the extra dependencies are irrelevant, I experiment with a second baseline (also based on full training and output filtering): the training data is first filtered according to the given criterion, but then completed using a Japanese parser, which is typologically very different,[26] so that the extra dependencies are mostly irrelevant for the test data. The standard parser thus provides an upperbound of what can be achieved in cross-lingual settings when keeping the irrelevant dependencies, and the parser trained on ill-typed data provides a lowerbound.

Finally, to investigate specifically the benefits of the learning method, the partial parser is also compared with a parser trained on the exact same (partial) data, but with partial training and full prediction (followed by output filtering).

In all cases, I use greedy parsers with ArcEager. The measures are averaged over all UD 2.0 treebanks. When training data is modified, the validation sets go through the same process.

Results Table 7.4 reports measures of coverage and precision, for all three criteria and all four methods. The '%tokens' lines indicate the proportion of gold dependencies corresponding to that criterion (in test data), the proportion of dependencies retained when filtering the output of the three baselines, and the proportion of tokens annotated by the partial parser. The 'precision' lines indicate the accuracy of each system on the corresponding subset of tokens, while 'std precision' corresponds to the accuracy of the standard parser on that exact same subset. Finally, the 'common' lines present the same measures performed only on the subset of dependencies annotated by all four parsers, thereby alleviating score differences due only to coverage effects.

First of all, it clearly appears that partial parsing is the best method to achieve correct proportions, even better than standard training. Without surprise, the parser using partial training significantly overgenerates dependencies of the type it knows: this confirms the need to entertain a specific training process when data comes already partial, but with a typological bias. Overgeneration explains mostly the huge drop in accuracy, but for length-based criteria, partial training remains below partial

[26]For the languages that are empirically too close to Japanese to suffer real discrepancies, I rather use an Arabic parser for completion. This concerns Japanese, German, Basque, Hindi, Hungarian, Kazakh, Tamil, Turkish and Uyghur.

Criterion	Measure	Std training	Ill-typed	Partial training	Partial parser
Easy on average	%tokens (ref: 27.4%)	28.9%	33.9%	35.6%	27.1%
	precision	86.88	69.99	68.89	85.98
	std precision	86.88	86.43	86.22	88.31
	common (26.7%)	88.61	85.14	87.28	86.81
Length 1	%tokens (ref: 42.9%)	44.4%	61.8%	80.1%	43.5%
	precision	87.42	62.78	50.61	87.06
	std precision	87.42	83.37	80.77	87.68
	common (41.7%)	88.76	87.44	88.03	88.34
Length $\leq$2	%tokens (ref: 63.4%)	65.0%	78.7%	80.9%	64.0%
	precision	85.31	69.89	69.93	85.30
	std precision	85.31	82.01	80.90	85.49
	common (61.6%)	86.54	85.04	85.93	86.46

TABLE 7.4: Evaluation of partial parsers on UD 2.0, compared to three baselines using standard and partial training. The 'Criterion' column indicates the subset of annotated tokens.

parsing even on the common subset; it thus appears that the wild guesses performed by a full parser on unannotated tokens pollute in fact the contextual information used to predict local dependencies, making them harder to learn.

The standard parser outperforms in general both partial methods, but it appears that this gain is only due, as expected, to the fact that its extra dependencies are actually relevant for the test set. Indeed, when those become irrelevant (and even though the corresponding output is filtered out) in the ill-typed setting, parser accuracy drops a lot, even on the common subset (which suppresses the overproduction effect): the ill-typed scores are clearly below all other methods, indicating that irrelevant dependencies also pollute the knowledge acquired on relevant attachments. So, when training a cross-lingual parser on source data with full annotations (typically a delexicalized parser), a filtering step is indeed beneficial – except maybe for language pairs which are close enough for having very few irrelevant dependencies – because in case of discrepancy the drop can be huge.

Regarding the partial parser specifically, its accuracy drop from the standard parser (that is, compared to 'std precision' values) seems to shrink when token coverage increases: the difference is large for a 27% coverage, while results are nearly identical with 64% of tokens. This suggests that partial parsers still have a hard time exploiting very sparse data,[27] even though they correctly learn the proportions.

[27]Note that for partial parsers there remains only one reference tree, and apart from spurious ambiguity, only one gold derivation: the treatment of unattached words becomes clear, they must be reduced before moving on to other tokens. In comparison, for non-partial parsers the oracle accepts any supertree of the reference one, i.e. a much larger reference space, which makes training much less challenging, and predictions much more often accurate, when annotations are very sparse. At the same time, both are impacted by the reduced usability of non-local features like '*previous action*' or '*other dependents of the token under focus*', which become much less informative for the parser as a consequence of their emptiness (for partial parsers) or their randomness in the explored space (for non-partial parsers, because there are much more gold derivations than with standard spurious ambiguity). Decisions thus become much more local, and consequently harder, an effect which increases with higher sparsity. It is consequently no surprise that partial parsers perform poorly in such cases.

Overall, these experiments validate the strategy of using partial parsers to produce partial annotations (subsequently forwarded as constraints to ulterior stages in the cascade): they achieve accuracies which are competitive with a standard parser trained in ideal typological conditions, and they yield better proportions. In the particular case where the data is already partial and a partial output is expected, they also outperform the partial training strategy by a large margin. However, because of remaining issues with high sparsity, it seems detrimental to start a cascade with a very partial parser: the first (unconstrained) parser should receive enough material to achieve an already good coverage of the data (at least around 40%), and let the subsequent stages refine the output with very sparse syntactic knowledge.

7.6 Parsing under constraints: exploiting prior annotations

To complete the technical improvements needed by the cascading framework, let us now move on to the last of the non-standard parsing tasks: training and using a parser under constraints.

Implementation As explained in Section 7.1, constraints are enforced with a cost-based filtering of valid actions. In PanParser, feature extraction is also adapted to directly exploit the constraints: the head, the left children (and valency) and the right children (and valency) of the first element in stack (s_0) are extracted from the parser state if already known, and from the constraints otherwise. The same is done for the first element in buffer (n_0), except that in that case its head and right dependencies are never available in the parser state – I have added feature templates using its right dependencies in a similar way as the left ones, but its head remains unexploited.

Evaluation Constrained parsing is evaluated using the same experimental setup as in the previous section. A parser trained under constraints is compared to a standard parser, when parsing test data under various kinds of constraints: gold constraints, constraints predicted by a standard parser with output filtering, and also constraints predicted by a partial parser, which is closer to the intended use case. In order to evaluate the effect of train-test similarity, during training I try applying either gold constraints ('Constrained' columns) or the constraints predicted by a partial parser ('Const.-pred' columns). Since they use the same criteria, the partial parsers are reused from experiments in the previous section.

For each constraint criterion and each kind of constraints, Table 7.5 reports the accuracies of all 3 parsers, computed only on the unconstrained tokens, a subset which differs depending on the type of constraints (see Table 7.4 for the size of those subsets).

Constraints	Gold			Standard parser			Partial parser		
Training	Constrained	Const.-pred	Std	Constrained	Const.-pred	Std	Constrained	Const.-pred	Std
Easy on average	76.73	75.82	76.00	74.50	75.04	75.40	72.46	73.52	73.99
Length 1	77.39	74.28	70.46	71.14	70.76	69.99	69.60	69.79	70.09
Length $\leq$2	74.80	71.25	64.87	64.30	64.60	62.94	62.99	63.83	62.76

TABLE 7.5: Evaluation of constrained parsers on UD 2.0, when applying different types of constraints at training and evaluation times. The 'Criterion' column indicates the subset of tokens expected as annotation constraints.

Under gold constraints, comparison of the 'constrained' and 'standard' parsers reveals a modest gain with the hardness criterion (+0.7), but huge improvements for both other criteria (up to +10). This shows that the upperbound gains achievable by that method are high. However, this benefit is lost to a large extent when applying constraints predicted by the standard parser: most of those gains are consequently due only to the better context provided by gold constraints. But the improvement is not null in the predicted case either (+1.1 and +1.4 with length-based criteria), which indicates that the parser achieves better training on the unconstrained tokens when it can focus on them, as expected. It is worth noting that in contrary to the previous section, here the best improvement is achieved when the parser has very few tokens to annotate, which is in fact when much information is provided by the constraints. Unfortunately, the remaining gains do not hold under constraints predicted by a partial parser: overall, constrained training underperforms standard training, even though it still yields a small improvement with the length-2 criterion.

A parser trained under constraints is consequently able to leverage them with great benefits, but its success depends a lot on the quality of those constraints; and when trained under gold constraints, it then fails to exploit the noisy constraints which would be provided in realistic settings.

An important cause of failure in such cases is the train-test discrepancy regarding the available information. In order to alleviate it, I consequently apply at training time the same constraints that the parser will encounter on test data, i.e. those predicted by the partial parser.[28] According to 'Const.-pred' results, the resulting parser is not as good to exploit gold constraints, but it is more robust to noisy constraints (even those of a kind it did not meet at training time, i.e. the output of a standard parser). Notably, and as expected, it performs much better under constraints predicted by a partial parser, so that it becomes on par with the standard parser in that case, and even achieves a significant gain for the length-2 criterion.

To summarize, when trained under constraints, the parser can exploit them with great success, but only if those constraints are of better quality than what it could

[28]Because the partial parser has been trained on the same data for which it then predicts the constraints, these constraints are relatively close to gold ones, and consequently not so realistic. Still, the difference appears sufficient to alleviate train-test discrepancy.

have learned itself. Otherwise, there may be some improvement, but it is not guaranteed and it depends (among others) on the amount of information conveyed by the constraints.

This limitation may not be an issue, though: while the above experiments are performed in a monolingual setting in order to better control the train-test consistency and the applicability of constraints, my true motivation for constrained parsing is cross-lingual. And even though in the monolingual case I have not yet found how to learn the constraints significantly better than the standard parser, I have also shown that in a context of transfer, the source partial parser will perform in general better on target data than the source standard parser, because it ignores the irrelevant dependencies. So, in a cross-lingual setting, the constraints have a real added value (compared to what that parser would have learned alone), so that the parser will manage to exploit them successfully.

Still, one question remains open: whether when learning directly from source data, the parser (typically a delexicalized one) should be trained under gold or predicted constraints. From the above experiments it appears that both choices have their advantages: the partial parser available at that time is indeed meant for target processing, and will thus perform poorly on source data, thereby providing noisy, useless constraints; but conversely, the gold constraints will significantly differ from those actually applied at test time, and measures have shown how badly this hurts constrained parsing. For the other treebanks involved in transfer (i.e. projected parses and target samples), it is clear however that the predicted dependencies should be preferred, since they are applied on target data only.

7.7 Conclusion: a dynamic oracle is all you need

This chapter has presented the design, functionalities and implementation of a newly released software, PanParser.

As it turns out, the transfer framework I aspire to is impossible to set up with state-of-the-art parsers. I have consequently proposed a series of improvements to the transition-based parsing framework, ranging from better training criteria to non-standard execution modes. My long-term purpose, reflected by PanParser's architecture, is to rationalize the various training strategies in parsing, in order to unify their frameworks and uncover new research perspectives; the ability to entertain parsing in non-standard use cases naturally arises from that effort.

Another key motivation of my work was to depart from common restrictions put on training data, which I achieved by allowing the use of partial trees and of non-projective examples with very simple means. This initiative serves the purpose of making all resources usable, in real-world low-resource scenarios.

The main lesson I retain from this study is the strength of the dynamic oracle framework, whose benefits surpass the mere improvements of some training properties. Indeed, they are in fact the technical solution underlying most improvements I have mentioned here. By extending them to global training, I have consequently opened a new research avenue on advanced training strategies.

The contributions of this chapter are not only formal: it concludes with a series of empirical studies providing insights on the usability and efficiency of parsing in the non-standard use cases I envision. I have shown that in a context of annotation projection, training on partial trees with the method I advocate yields accuracies comparable to state-of-the-art transfer, but in a painless and much cheaper way. I have also validated the design of parsers producing partial outputs and operating under dependency constraints: while I fail in general to outperform standard parsers in a monolingual setting, I have shown that for cross-lingual applications, they serve their purpose well, by ignoring noxious pieces of knowledge, by making use of the whole information at their disposal, and by focusing on their own task.

As a final note, it seems important to remind that despite what this chapter may suggest, my transfer approach is not tailored to one specific NLP task. Notably, even if achieving the non-standard tasks described in Section 7.1 has required work that is specific to parsing, the tasks themselves are not. This means that my proposal to apply a cascading framework for transfer remains task-agnostic. For instance, with a PoS tagger able to train on partial tag sequences, to predict partial tag sequences, to tag under tag constraints, and to exploit any PoS-annotated data without restrictions, the framework could be entertained exactly in the same way.

Chapter 8

Incompatible representations of knowledge

***

Contents

This chapter addresses the second prerequisite of a transfer framework with selective combination, which is the conversion of PSEUDO-SHARED knowledge into SHARED knowledge, as defined in Chapter 5. Indeed, my framework is based on the idea that even distant sources (typically Slavic languages for Romanian) can provide valuable syntactic knowledge, and it aims at identifying and extracting relevant information from those. Yet, in practice, it is hard to query anything from such poor sources, because their syntax as a whole is so different that it hides the few pieces of useful knowledge. It would then be pointless to develop relevance measures and selective combination if only the best sources were usable.

The work presented here mostly borrows its content from (Aufrant et al., 2016d). It does not cover all aspects of converting PSEUDO-SHARED into SHARED knowledge, but focuses on delexicalized parsing,[1] for which the main source of cross-lingual divergences relates to word order. I claim that in spite of common data representation (i.e. UPOS and UD), the *knowledge* extracted from this data comes in different flavours: precisely because of word order, the feature-level representation of the same knowledge piece may vary a lot between source and target, hence preventing querying and effective transfer.

In Section 8.1 I exhibit theoretical and empirical evidence regarding this issue, which I propose to solve in Section 8.2, by contrasting two preprocessing techniques, one data-driven and one knowledge-driven. Extensive experiments, reported in Section 8.3, reveal that these methods indeed outperform the baseline by a wide margin; but they also show that my method achieves exceptionally huge gains on some classes, turning cases where virtually nothing is transferred into usual transfer accuracies, thereby confirming that knowledge representations by themselves can hinder transfer.

8.1 Typological differences incur transfer impossibilities

In the following, for clarity, I illustrate the issue, measure the effects and evaluate my proposal using specifically the greedy ARCEAGER version of PANPARSER. However, the motivations hold regardless of the chosen parsing system, and in the course of exploratory experiments, I have measured similar effects and improvements with other systems, like beam parsers or even graph-based parsers.

8.1.1 Order-dependent knowledge representation

While it is clear that transition-based parsers like ARCEAGER handle left and right dependencies differently, with two separate LEFT and RIGHT actions,[2] it is much less obvious that the actual predictions correspond in fact to two completely different input configurations, nor that this concerns graph-based parsing as well.

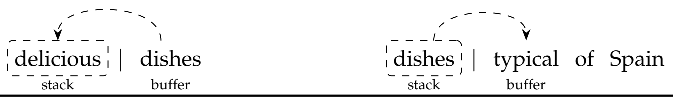
Indeed, most features representing the prediction configurations are themselves sensitive to word order, as they are typically extracted from a limited window centered on the two tokens under focus (i.e. to move or attach). For transition-based parsers like ARCEAGER, these are the top of the stack (s_0) and deeper stack elements (s_1, s_2) on its left, the head of the buffer (n_0) and additional tokens (n_1, n_2) on its right. The resulting feature representations consequently encode the position of the word in stack or buffer, which in turn depends on its position in the sentence. Graph-based

[1]While it is true that the transfer issues presented here are at least partially solved by annotation projection, this method has itself other issues, in first place the availability of parallel data, and as such is not the answer to the reported transfer failures.

[2]Note that labeled parsers which predict the label together with the arc have exactly the same differential handling, between e.g. the RIGHT_{obj} and RIGHT_{iobj} actions.

parsing is slightly less order-dependent, since the tokens under focus are identified as ‘candidate child’ and ‘candidate head’ instead of their position; but the statement still holds because many other standard features, like ‘tag between head and child’ or ‘tag of child’s left neighbour’, remain sensitive to word order.

Hence, since data-driven parsers never learn the available knowledge at the conceptual level, but as *embedded* in a given sentence, it ensues that even generic knowledge like ‘adjectives depend on nouns’ ends up represented as instantiated, non-generic knowledge at the classifier level. Figure 8.1 shows in which measure this representation can vary.



Conceptual level	Adjectives depend on nouns	
Data level	ADJ NOUN	NOUN ADJ
Classifier level	Feature (s_0 = ADJ $\wedge$ n_0 = NOUN) has a high weight for LEFT	Feature (s_0 = NOUN $\wedge$ n_0 = ADJ) has a high weight for RIGHT

FIGURE 8.1: Knowledge representations of two ADJ-NOUN dependencies in English, at the conceptual, data and classifier levels.

8.1.2 Transfer failures

The idea of delexicalized transfer is to transfer language-independent knowledge, resulting from linguistic universals or relatedness. But I claim that in spite of this shared knowledge, in many cases the corresponding pieces of knowledge cannot be transferred, precisely because the source and target *knowledge* representations are incompatible: as such, that knowledge does not belong to the SHARED category but to the PSEUDO-SHARED one, which gathers the syntactic content that is similar in both languages but in theory only.

I illustrate this on delexicalized transfer from English to French. Let us assume that we have a delexicalized English parser that is able to perfectly predict the dependency structure of the noun phrase *the following question* and we use it to annotate the corresponding French phrase *la question suivante* (literally, *the question following*[3]). Thanks to the UD guidelines, these phrases can be represented in a unified manner by mapping word forms into the corresponding PoS, yielding respectively DET ADJ NOUN and DET NOUN ADJ. As the English parser has learned that ‘DETs depend on NOUNs’ and that ‘ADJs depend on NOUNs’, the appropriate parse for the French phrase should be obvious, as these rules apply cross-linguistically. PoS sequences thus seem to provide an appropriate level of abstraction for cross-lingual transfer.

[3]Keeping the original order (*la suivante question*) would be wrong in French.

However, contrary to what this intuition suggests, the transfer of the ADJ-NOUN dependency often fails in practice. Indeed, when parsing the French phrase, the parser configuration will be mainly described by the feature $(s_0 = \text{NOUN} \wedge n_0 = \text{ADJ})$ (as *question* appears before *suivante*, it will be put on the stack first) while for the English phrase the relevant parser configuration would be $(s_0 = \text{ADJ} \wedge n_0 = \text{NOUN})$. For lack of connecting these two situations, the parser has no way to predict the correct attachment in French using only English training instances.

Experimentally, on UD 1.3, and denoting UAS $\left[\begin{smallmatrix} C_2 \\ C_1 \end{smallmatrix}\right]$ the percentage of C_1 tokens depending on a C_2 token that are correctly attached by the parser, while the English delexicalized model has a UAS $\left[\begin{smallmatrix} \text{NOUN} \\ \text{ADJ} \end{smallmatrix}\right]$ of 91.1% on English, it drops down to 50.8% for French. This decrease directly results from the word order difference between French and English, as English adjectives are almost always preposed[4] while their position in French is less deterministic: in the French UD, 28% of the adjectives occur before their head noun and 72% after it. As a result, the UAS $\left[\begin{smallmatrix} \text{NOUN} \\ \text{ADJ} \end{smallmatrix}\right]$ score on French actually decomposes as 96.8% for UAS $\left[\begin{smallmatrix} \text{NOUN} \\ \text{preposed ADJ} \end{smallmatrix}\right]$ and 34.5% for UAS $\left[\begin{smallmatrix} \text{NOUN} \\ \text{postposed ADJ} \end{smallmatrix}\right]$.[5] These observations highlight the impact of word order on delexicalized transfer: attachment patterns are not robust to variations in word ordering. Note that transfer from French to English is much more successful, with a UAS $\left[\begin{smallmatrix} \text{NOUN} \\ \text{ADJ} \end{smallmatrix}\right]$ of 80.5%. This is because the source language (here French) contains a sufficiently large number of preposed adjectives, which makes it possible to learn the patterns that are useful for English.

The discrepancies in word order can have an even more dramatic effect when transferring parsers between languages in which adjectives have a fixed position. This, for instance, happens when the source is Bulgarian (almost only preposed adjectives) and the target is Hebrew (only postposed): the resulting UAS $\left[\begin{smallmatrix} \text{NOUN} \\ \text{ADJ} \end{smallmatrix}\right]$ is as low as 28.7% (compared to an overall UAS of 60.1%). In the reverse direction, it drops down to 2.8% (UAS $\left[\begin{smallmatrix} \text{NOUN} \\ \text{preposed ADJ} \end{smallmatrix}\right]$ of 0.7%, UAS $\left[\begin{smallmatrix} \text{NOUN} \\ \text{postposed ADJ} \end{smallmatrix}\right]$ of 54.5%, with an overall UAS of 50.6%).[6]

The impact of differences in word order on cross-lingual transfer is not limited to the attachment of adjectives. Consider, for instance, the English phrase *the neighbor's car* (DET NOUN PART NOUN) and its French translation *la voiture du voisin* (DET NOUN ADP NOUN). After attaching function words, all that remains for the parser to process is the bigram NOUN NOUN: while the English parser has been trained to predict a left dependency (*car* being the head of *neighbor*), for French it must predict

[4]In the English UD corpus, 93% of the adjectives come before the noun they depend on.

[5]This observation is consistent with the English monolingual scores (93.2% for the UAS $\left[\begin{smallmatrix} \text{NOUN} \\ \text{preposed ADJ} \end{smallmatrix}\right]$ majority case, and 55.0% for the much rarer UAS $\left[\begin{smallmatrix} \text{NOUN} \\ \text{postposed ADJ} \end{smallmatrix}\right]$ case).

[6]Source data quality cannot be the only cause of such poor results: when delexicalized models apply monolingually, UAS $\left[\begin{smallmatrix} \text{NOUN} \\ \text{ADJ} \end{smallmatrix}\right]$ is 97.4% in Bulgarian and 88.4% in Hebrew.

a right dependency (*voiture* being the head of *voisin*). Here the discrepancy of genitives' position across languages does not involve unseen features, but still leads the model to predict a wrong dependency with high confidence.

My work aims at addressing such scenarios in which knowledge transfer is impeded by the word order of the source language. While current state-of-the-art models learn that 'an ADJ followed by a NOUN depends on that NOUN' and 'the first NOUN depends on the second NOUN', I would like them to transfer more abstract patterns such as 'ADJs depend on NOUNs' and 'genitives depend on NOUNs', leaving it up to the target side to decide which of both NOUNs plays the role of genitive.

8.2 Reshaping training instances

My proposal to solve the aforementioned issues applies Bender (2009)'s ideas, whereby introducing linguistics can contribute to language independence, here by singling out typological divergences.

Despite the intuition, it is indeed out of question to fully abstract the learned knowledge from word order (e.g. with feature engineering), because it would make sentences like those of Figure 8.2 indistinguishable. I rather aim at keeping the systems order-dependent, but accounting for opposite typologies during the transfer step.

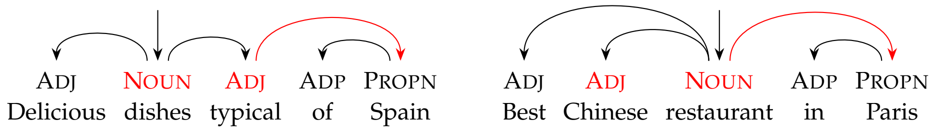


FIGURE 8.2: Pair of sentences for which word order information matters: otherwise the parser cannot attach the proper noun correctly.

I see two strategies to this end: either a direct mapping of feature weights (i.e. converting (s_0 = ADJ $\wedge$ n_0 = NOUN) values into (s_0 = NOUN $\wedge$ n_0 = ADJ)), or transforming the data itself, so that the knowledge 'adjectives depend on nouns' is embedded in sentences with appropriate typological properties and is naturally learned on (s_0 = NOUN $\wedge$ n_0 = ADJ) features. Because it is a generic extension applicable to any parsing system, and because the transformations are easier to interpret, I have chosen to transform the data.

Hence, I propose to preprocess source data to fit the word order typology of the target *before* training a delexicalized parser. Actually, I do not restrain to word order, but also consider typological phenomena like the absence of a given PoS class in one language (typically, determiners: knowledge is embedded differently in 'ADP

NOUN' and 'ADP DET NOUN'). With this preprocessing step, the new source treebank is more similar to target word sequences, which makes the cross-lingual knowledge more accessible. It is important to note that the available information is the same before and after preprocessing (no dependency is ever added), the only modifications are on how it is presented at training time.

In the following, I introduce two ways of generating such transformations, by removing or permuting tokens. The first approach is data-driven: it estimates a language model (LM) on target PoS sequences and uses this PoSLM to find a tree in the neighbourhood (i.e. local reorderings) of the source sentence which fits target word order. The second strategy extracts typological features from the WALS database (Haspelmath et al., 2005; Dryer and Haspelmath, 2013), to generate a series of heuristic transformation operations.

The problem of finding good reorderings of a source sentence is closely related to the problem of word (p)reordering in Statistical Machine Translation (SMT) (Bisazza and Federico, 2016). However, where preordering aims at finding an optimal permutation of source words *for each source sentence*, my objective here is different, as I only intend to 'fix' a sufficiently large number of divergent patterns between the source and target languages, so as to increase the effectiveness of transfer *at the model level*. I will show that optimal reorderings at the sentence level can even be detrimental in this frame.

8.2.1 Target-optimal reordering with a language model

My first resource-light approach consists of two steps. I first generate a small subset of possible token permutations, compactly encoded in a finite-state graph. In my experiments, I consider all the permutations licensed by the MJ-2 reordering scheme (Kumar and Byrne, 2005), which generates all possible local permutations within a window of three words. Machine Translation experiments have shown that the MJ-2 constraints capture lots of plausible reorderings (Dreyer et al., 2007). In the context of cross-lingual transfer, its local nature enables to correct several important divergences in word order (e.g. the adjective-noun divergence described in previous section), while keeping the size of the reordering lattice polynomial with respect to the sentence length (Lopez, 2009).

The permutation lattice is then searched for a reordering that (a) corresponds to a high probability target PoS sequence and (b) preserves the projectivity constraint. By enforcing projectivity, I prevent the reordered words from being moved far from their own dependents, which would create other inconsistencies. In practice, I first generate the lattice of MJ-2 reorderings, score it with a language model estimated on the target PoS sequences, and extract the 1,000-best sequences. After filtering the reorderings inducing non-projective trees, I retain the one-best sequence (if one projective tree exists), or the original sequence otherwise. I expect the word order of this transformed source to be very close to the word order of a typical target sentence, and hence to fit the typology of the target. I then transform the gold dependency tree according to this permutation and use it to train a target-adapted model.

This approach can be viewed as an extension of the data selection technique of Søgaard (2011) in which the delexicalized model is trained only on the source examples that are the most *relevant* for the target at hand. The similarity between the source and target languages is based on the similarity between their PoS sequences: experimentally, the author retains the 90% sentences with lowest perplexity according to a target PoSLM. I add here an extra degree of freedom by allowing changes in the word order of the source PoS sequence, rather than simply discarding sentences: instead of selecting relevant sentences, I maximize the relevance of *each* sentence.

8.2.2 Heuristic rewrite rules based on typological knowledge

My second proposal takes advantage of the linguistic knowledge that is now available for many languages. I use here the WALS database, which contains a series of linguistic features documenting 2,679 languages. Some of these features are of prime interest for my study, and express general properties related to word order; they are available for more than 1,000 languages. In this work I focus on the following seven features that describe whether some PoS classes exist in a language and their relative position (preposed or postposed to the noun, or no dominant order): 37A (definite articles), 38A (indefinite articles), 85A (order of adposition and noun), 86A (order of genitive and noun), 87A (order of adjective and noun), 88A (order of demonstrative and noun) and 89A (order of numeral and noun).[7]

I first extract the relevant features for each considered language, quantify their value and automatically associate them to properties of the raw UPOS sequences. I indeed extrapolate the order of DET and NOUN from feature 88A and identify the genitives mentioned by feature 86A as NOUNs or PROPNs depending on a NOUN. With an otherwise straightforward mapping, this results in the following set of properties: no definite DET, no indefinite DET (including the affix cases), and a precedence rate (denoted PR) of 0% (postposed), 50% (no dominant order) or 100% (preposed) for ADPs (resp. genitives, ADJs, DETs, NUMs) depending on a NOUN.[8] It is worth mentioning that, contrary to other uses of WALS in the cross-lingual literature, here the actual semantic of each feature value is used.

The 'No dominant order' feature value of WALS provides very useful quantitative information: contrary to the PoSLM-based approach, which puts hard constraints on each phenomenon by choosing a reordering even when several choices

[7]I do not consider here features (81A, 82A, etc.) describing the relative position of a head VERB and its dependents. Their use would require to condition the preprocessing patterns on labeled dependency relationship in the source, a task I leave for future work.

[8]For ADPs, and for resilience to annotation inconsistencies, I also include ADPs that are heads of NOUNs.

would be almost equally likely, WALS features indicate when and how to balance the transformed treebanks.

By comparing two languages based on their feature values, it is then possible to define actionable transformation rules that remove or permute tokens and their associated subtrees. Table 8.1 lists the transformation rules derived from each pair of features. I preferred smooth transformations (with mean PR objectives and error margins) to prevent a full transformation of the corpus and a risk of information losses if the child position is less deterministic than expected. For instance, in the case of transfer from English (ADP-NOUN) to Japanese (NOUN-ADP) and according to the fourth transformation rule, I target a precedence rate of ADPs to NOUNs between 45% and 55%. This means that during source treebank traversal, while the precedence rate in previous sentences is above 55% (resp. below 45%), any encountered ADP-NOUN (resp. NOUN-ADP) bigram holding a dependency is switched, along with their dependents to preserve projectivity. According to the first rule, for transfer to Czech (no definite article) from any source, all definite articles are systematically removed from source data.

Source feature	Target feature	Transformation rule
any	no DEF-DET	remove all definite DETs
any	no IND-DET	remove all indefinite DETs
PR = 0%	PR ≥ 50%	switch subtrees to reach PR = 50% (with 5% error margin)
PR = 100%	PR ≤ 50%	switch subtrees to reach PR = 50% (with 5% error margin)
PR = 50%	PR = 100%	switch subtrees to reach PR = 75% (with 5% error margin)
PR = 50%	PR = 0%	switch subtrees to reach PR = 25% (with 5% error margin)

TABLE 8.1: Transformation rules extracted from the comparison of the feature values of a language pair. All other feature pairs result in a no-op.

For each sentence, I apply each rule on the whole sequence (and then iterate 3 times to capture recursive phenomena). Such heuristic rule application strategy is undoubtedly sensitive to the rule ordering, but I have not yet investigated this aspect and simply apply rules according to the lexicographic order of the child tag, breaking ties using word position.

In comparison to the PoSLM-based approach, the WALS-based approach suffers from a lack of exhaustivity regarding word order; by design, less phenomena will be captured. However, since the objective is not the best possible reordering but only more compatible PoS sequences, exhaustivity is probably not a big issue. Besides, working with a discrete and reduced set of transformation operations gives a better control on the rewriting of dependencies. It also allows to use extra operations such as word deletion, a transformation that may be difficult to control in the PoSLM approach.

Altogether, this linguistically rich method presents a notable upside: since all the required information is available in WALS, it is readily usable for more than thousand languages. Provided that PoS tags can be generated for the target data to parse, no extra resource is required, while estimating a PoSLM still requires a sufficiently large corpus of reliably PoS-tagged target data.[9]

8.3 Experiments

To evaluate the effect of this preprocessing on transfer, I conduct extensive experiments on 40 languages exhibiting very different characteristics and covering several language families. On top of overall gains, this consequently enables per-family evaluation, and fine-grained analysis of the results.

8.3.1 Experimental setup

Both proposals are evaluated on UD 1.3[10] and compared with three baselines: (a) standard delexicalized transfer, (b) the data point selection method of Søgaard (2011) and (c) the multi-source combination of Rosa and Zabokrtsky (2015) weighted by the KL_{cpos^3} metric (both described in Section 4.1). I also include the UAS of KL_{cpos^3} multi-source combination built on top of my knowledge-based model.

When target PoS sequences are needed (to train PoSLMs and compute KL_{cpos^3}), I use the training sets of UD, stripped of all dependencies. The PoSLMs are based on trigrams. From WALS,[11] I extract and use the 37A, 38A and 85A to 89A features, but for some languages, this information was incomplete. I completed missing features with a majority vote of the languages of same genus if available in the database; otherwise (i.e. for ancient languages, all absent from WALS) I assumed that there were separate article tokens and that there was no dominant order for word order features.

For each component of the algorithms, I use the UPOS tagset and gold PoS tags. While this scenario is probably unrealistic, I needed to remove discrepancies in PoS quality between languages, in order to get a clearer picture of the effect of my methods on knowledge transfer. For parsing, I use the ArcEager perceptron version

[9]While the standard approaches for PoS tagging are either to train a tagger on such target data, or to leverage parallel data which is not always available, I believe that these are not the only options and that the tagging limitation should not restrict the cross-lingual parsing scenarios. Already strong tagging baselines can indeed by obtained e.g. by Wiktionary crowd-sourcing or by training on sample annotated data, too small to estimate a PoSLM but sufficient for baseline tagging. Garrette and Baldridge (2013) also argue in favour of tag dictionaries, which are a better use of sparse annotators' time than standard data annotation. A few straightforward rules (easy and fast to write) can also successfully complement these baselines to handle inflected forms: when data falls short, data-driven systems are definitely not the only solution. Moreover, the impact of using a poor tagger seems much worse if the annotated data is used to train a PoSLM, because it would incur many illegitimate reorderings, than if it concerns only the test data, in which case errors have consequences limited to the current parse.

[10]In order to present only fair sources, for languages in which several treebanks are available, I retain only the *main* treebank. This is the case for the following languages: cs, en, es, fi, grc, la, nl, pt, ru, sl and sv.

[11]`http://wals.info` (accessed in June 2016)

of PanParser, with a greedy dynamic oracle and Zhang and Nivre (2011)'s feature templates (assuming fully delexicalized representations and unlabeled arcs).

8.3.2 Results

	[Mono-source															]	[Multi-source]	
Target	Delexicalized				PoSLM selection				PoSLM reordering				WALS rewrite rules				Delex.	WALS
	min	med	max	μ	min	med	max	$\Delta\mu$	min	med	max	$\Delta\mu$	min	med	max	$\Delta\mu$		
ar	5.1	43.2	56.9	36.1	4.8	43.0	57.2	-0.2	18.9	45.1	57.2	+5.6	12.2	47.6	56.9	**+5.7**	57.3	57.8
bg	26.4	67.5	78.9	59.6	26.3	67.5	78.9	-0.1	35.5	65.1	74.4	+1.5	27.2	67.6	78.9	**+1.8**	79.6	79.0
ca	28.4	62.3	78.5	57.8	27.9	62.0	78.7	-0.1	33.5	60.8	75.5	-0.2	30.4	66.2	78.6	**+1.9**	79.2	79.1
cs	29.7	58.5	74.0	54.3	29.2	58.6	74.0	-0.0	37.1	58.4	68.8	**+1.4**	30.8	59.3	73.8	+1.4	74.0	73.8
cu	22.6	58.5	74.7	53.9	22.6	58.7	75.4	+0.0	38.2	60.2	70.0	**+3.8**	24.3	60.3	74.7	+1.6	74.7	74.7
da	28.0	64.5	75.3	58.2	27.5	63.8	75.3	-0.2	40.3	61.0	70.4	-0.0	28.6	64.6	74.5	**+1.4**	75.7	75.2
de	36.2	61.2	70.5	57.7	35.9	61.0	70.0	-0.2	45.4	61.1	69.3	+1.3	43.0	61.8	70.8	**+1.5**	68.7	68.7
el	29.0	51.0	67.8	49.5	28.9	50.5	68.2	-0.0	33.5	51.6	64.9	+0.4	29.8	51.6	67.6	**+1.7**	67.0	66.2
en	33.1	56.5	65.8	52.8	32.5	56.2	65.5	-0.2	38.4	57.0	63.8	**+1.2**	32.2	58.4	65.9	+1.2	65.7	66.0
es	30.0	63.9	78.5	58.9	29.3	64.4	78.5	-0.0	37.6	62.8	76.1	+0.3	31.5	66.6	78.4	**+1.8**	79.2	79.3
et	28.4	53.1	69.4	51.5	26.9	52.9	69.6	-0.2	36.3	57.0	67.9	+3.7	37.1	57.9	69.3	**+4.4**	69.4	69.3
eu	20.7	45.2	57.8	44.2	20.9	46.4	57.7	-0.1	24.3	54.6	64.1	**+8.1**	24.6	52.0	63.3	+5.7	55.7	60.9
fa	17.9	45.3	56.1	40.3	17.6	45.0	56.0	-0.1	26.7	46.3	56.8	+3.2	25.5	48.4	58.8	**+5.2**	61.4	63.3
fi	27.4	48.1	62.1	46.6	27.4	48.2	61.8	-0.0	32.4	50.2	58.8	+2.3	32.4	53.0	62.2	**+3.7**	62.1	62.2
fr	30.9	64.0	79.1	59.0	29.9	63.9	78.7	-0.2	35.0	61.4	76.8	+0.5	34.2	66.0	78.9	**+1.8**	79.8	79.5
ga	16.4	56.0	65.8	50.1	16.3	56.2	66.4	-0.1	26.6	56.2	64.6	+1.8	20.8	59.0	65.3	**+2.3**	67.4	67.2
gl	33.0	40.3	47.5	40.6	32.8	40.5	47.5	-0.1	32.1	43.0	48.2	+1.5	35.6	43.7	51.0	**+2.6**	46.7	46.6
got	26.4	58.0	72.7	54.3	26.4	57.5	73.4	-0.0	38.0	60.0	66.3	**+3.1**	28.2	58.9	73.9	+0.9	72.7	73.9
grc	32.5	53.9	57.3	50.7	29.8	53.9	57.8	-0.1	39.0	53.9	58.3	**+1.1**	32.4	53.9	57.8	+0.0	61.0	60.3
he	20.1	53.8	68.0	49.9	19.8	54.2	67.7	+0.1	30.2	54.1	63.6	+1.2	21.9	55.4	65.8	**+1.6**	71.2	68.7
hi	11.0	27.1	66.5	32.3	11.1	26.9	65.8	-0.1	22.0	37.5	61.6	**+6.7**	19.8	33.8	66.8	+5.4	37.1	44.2
hr	26.8	55.4	71.2	52.0	26.0	56.3	70.9	-0.2	35.7	56.3	66.9	**+1.7**	28.9	56.3	70.3	+1.5	73.9	73.0
hu	27.8	52.7	67.8	50.8	27.1	53.1	68.2	-0.0	40.4	56.3	65.4	**+4.4**	40.1	55.4	68.3	+3.4	63.0	64.4
id	17.4	49.2	70.1	48.8	18.2	49.0	70.3	+0.1	27.6	50.2	66.1	+1.5	23.1	53.8	69.6	**+3.7**	70.8	71.9
it	31.0	67.1	82.6	61.7	30.4	66.9	82.2	-0.1	38.1	67.0	80.7	+1.2	34.0	70.8	82.3	**+2.2**	83.2	82.9
ja	7.0	18.6	72.6	26.7	7.2	18.3	72.2	+0.1	15.7	32.9	70.6	+10.0	18.1	35.2	72.3	**+11.4**	63.3	63.5
kk	10.7	33.0	56.3	32.4	10.9	34.0	54.3	+0.2	17.9	35.2	52.6	+3.4	20.6	38.5	55.6	**+4.9**	53.9	54.6
la	14.4	49.9	64.1	47.1	14.3	50.0	63.5	+0.0	19.5	51.4	61.8	+1.9	21.1	53.3	63.3	**+2.1**	58.3	60.0
lv	22.6	40.3	55.7	40.6	22.5	40.1	55.8	-0.2	28.7	42.6	51.0	+1.8	35.0	47.1	55.4	**+5.5**	50.2	57.3
nl	27.5	51.9	61.7	48.9	27.9	52.2	62.2	+0.1	32.4	49.3	56.8	-2.4	28.4	52.4	60.4	**+0.5**	62.3	60.7
no	25.8	64.0	76.6	57.1	25.6	63.9	76.7	-0.1	37.2	58.5	69.2	-0.2	26.5	63.8	76.2	**+1.3**	76.6	76.5
pl	25.7	62.1	77.9	59.2	25.6	62.7	77.9	-0.1	36.0	65.6	76.2	**+3.4**	29.5	65.5	77.4	+2.4	78.0	77.6
pt	30.8	62.8	75.5	56.7	30.2	63.3	75.5	+0.0	35.7	60.0	73.5	-0.9	32.8	63.5	75.4	**+1.4**	75.5	75.7
ro	19.8	55.5	69.2	51.8	18.6	55.7	68.7	-0.1	31.7	58.5	67.7	+3.1	24.1	60.5	70.3	**+4.0**	71.8	72.0
ru	26.8	53.9	69.0	51.3	26.1	54.0	68.9	-0.0	34.6	55.1	67.9	+2.3	30.9	59.2	68.7	**+4.2**	71.0	70.4
sl	30.6	65.2	80.4	59.4	30.4	65.1	80.5	-0.0	41.8	64.8	77.4	**+2.7**	30.5	64.9	80.4	+1.4	80.4	80.4
sv	29.4	62.7	75.5	56.9	29.4	62.3	75.5	-0.2	39.6	60.1	70.6	+0.5	30.4	63.9	74.9	**+2.3**	73.1	73.5
ta	9.1	36.3	66.3	36.8	9.1	35.6	66.4	-0.0	18.9	43.7	64.5	**+6.2**	19.0	41.1	65.8	+4.7	66.3	65.8
tr	14.1	35.3	67.0	38.8	14.7	35.4	67.0	-0.1	19.5	39.5	64.5	+2.0	21.5	40.8	67.8	**+3.5**	58.6	58.5
zh	15.6	32.5	43.1	31.7	15.8	32.5	43.0	+0.1	18.9	35.5	41.8	+2.3	20.1	36.6	44.1	**+3.5**	40.2	42.2
μ	23.7	52.0	68.2	49.2	23.3	52.0	68.1	-0.1	31.8	53.5	65.6	+2.3	27.9	55.2	68.3	**+2.9**	66.9	67.4

TABLE 8.2: UAS of the various mono-source and multi-source transfer methods, on each UD target language (using UD language codes). The first line reads as follows: for delexicalized transfer to Arabic, the worst, median and best sources yield UAS scores of 5.1, 43.2 and 56.9, and the average score over all 39 sources is 36.1, which the WALS-based method improves by 5.7 points.

Table 8.2 presents UAS results for the various transfer methods considered. As these experiments amount to 6,320 evaluated parsers, I provide the results in a compacted form as follows: for each target language, for mono-source transfer, I report the scores of the worst, median, best sources and the average score (or average gain) over all sources.

Overall, both preprocessing techniques outperform the direct transfer method of Zeman and Resnik (2008) as well as the selection strategy of Søgaard (2011). The WALS-based rewriting approach yields higher improvements (+2.9 UAS on average) than the PoSLM-based reordering strategy (+2.3 UAS on average). Thanks to the diversity and the large number of sources, the multi-source methods have here much higher accuracies, often better than the best source; even in this setting, using WALS provides a slight improvement over the baseline multi-source parser.

My experiments also show that the selection baseline method does not perform as well on UD as it did on the CoNLL 2006 Shared Task data. Those differences can be explained in three ways. First, I experiment with cleaner treebanks and benefit from the availability of unified tagsets and annotation schemes. This is in contrast with previous experiments, which were using a tagset mapping as a preprocessing step, making the net effect of data selection more difficult to single out and evaluate precisely. Second, the data selection method was primarily intended for distantly related languages, whereas the UD corpus now offers a wide language diversity and often a few good sources for which data size reduction is only detrimental. Third, the original work experimented on multi-source transfer, in which case selection can also help by balancing each language's contribution, which it cannot in this mono-source setting.

In general, my methods do not improve the best source but have a large effect on bad and average sources, often turning them into competitive sources. This is particularly true with PoSLM reordering, which improves the worst sources by 8.1 points and degrades the best ones by 2.6 points. By contrast, the WALS-based method is more conservative and offers lower but more reliable improvements, which in average proves successful.

		Target language					
		Romance	Germanic	Slavic	Finno-Ugric	Semitic	Ancient
Source language	Romance	67.1‖65.6‖67.2	60.4‖60.4‖61.7	63.1‖63.5‖63.0	46.4‖50.8‖52.5	54.1‖52.1‖52.9	56.7‖56.5‖54.9
	Germanic	61.2‖63.5‖65.8	65.9‖63.1‖65.8	61.3‖62.2‖63.2	57.2‖58.6‖58.5	41.2‖48.2‖49.8	54.5‖57.1‖56.7
	Slavic	63.5‖61.7‖66.0	63.8‖60.5‖64.3	72.6‖68.4‖71.8	53.2‖57.0‖58.4	54.7‖53.6‖56.8	59.0‖59.2‖60.1
	Finno-Ugric	46.3‖51.9‖52.3	57.1‖56.2‖57.6	53.8‖58.6‖56.9	64.1‖63.0‖64.2	30.0‖43.6‖41.5	50.8‖55.7‖56.1
	Semitic	54.1‖54.2‖54.1	40.6‖48.2‖51.1	42.5‖54.6‖56.1	30.8‖41.2‖44.1	55.4‖55.6‖54.8	53.7‖55.9‖54.4
	Ancient	56.1‖49.2‖55.9	56.7‖51.5‖56.1	60.9‖57.5‖60.6	52.2‖54.9‖56.0	51.1‖47.0‖50.6	62.7‖60.0‖62.6

TABLE 8.3: Delexicalized, PoSLM-based reordered and WALS-based UAS aggregated over language family pairs. The first column reads as follows: the average UAS over all pairs of two Romance languages is 67.1 for mono-source delexicalized transfer; it is slightly improved (67.2) by the WALS-based method. Over all pairs of a Germanic source and a Romance target, the average mono-source UAS raises from 61.2 (delexicalized baseline) to 63.5 (PoSLM-based reordering) and 65.8 (WALS-based rules).

Table 8.3 reports the average over some language families[12] of the UAS of the baseline, reordered and WALS-based mono-source models. It shows that accuracies of related sources are only marginally modified when source sentences are transformed according to WALS, which could be expected as related languages share most of their typological features. On the contrary, large gains are obtained for distantly related languages. Such languages are typically poor sources in direct delexicalized transfer due to systematic labeling errors that mostly concern few frequent

[12]While the considered ancient languages belong to some of those families, I chose to gather them into a separate category, since they rely on the same WALS completion heuristic, instead of their actual typological features.

word classes (in correlation with their typological features). I have found that such errors can often be corrected by transforming the source sentences. Once those errors are handled, the now competitive sources can in turn contribute with valuable knowledge in multi-source settings; the primary purpose of this study has been achieved.

8.3.3 A fine-grained analysis

I have also investigated the improvements made over the baseline by my best method, the WALS-based rewriting rules, by analyzing the gain in accuracy separately for various PoS. It appears that, unsurprisingly, improvements mostly concern PoS classes covered by the WALS features. For instance, the issue mentioned in § 8.1 for the English-French pair is almost solved with source reordering: 90.4% of the postposed ADJs are correctly predicted by the WALS-based method (34.5% in the baseline), without any detrimental impact on the preposed ones. The same holds for the Hebrew-Bulgarian textbook case, where the UAS $\begin{bmatrix}\textsc{Noun}\\ \textsc{Adj}\end{bmatrix}$ raises from 0.7% to 95.1%.

I observe similar behaviors across the board for all the classes targeted by transformations: transfer from Czech to Danish had UAS $\begin{bmatrix}\textsc{Noun}\\ \text{preposed } \textsc{Noun}\end{bmatrix}$ and UAS $\begin{bmatrix}\textsc{Noun}\\ \text{postposed } \textsc{Noun}\end{bmatrix}$ scores of 2.8% and 78.4%, with WALS-based preprocessing they are respectively 61.1% and 80.4%. In Finnish-Arabic, scores of 6.3% and 30.9% on ADJs and ADPs become 65.8% and 61.4%, etc. In whole, 21% of the considered language pairs present very large gains (over +50 points) for at least one frequent tag pair (over 30 dependency occurrences in test data).

Careful comparison of results for both PoSLM-based methods shows that reordering improves ADJs' attachment for instance, when data selection does not. This can be explained in two ways. First, if the source corpus contains a very limited number of preposed ADJs, even with perfect selection the ADJs in final data cannot be mostly preposed. Second, data selection mostly targets sequences very far from the target syntax: sentences that only violate a local preference of child position are less fluent, and consequently have a lower rank, than their hypothetical counterpart with switched positions; but they are not ungrammatical enough to be pushed into the 10% worst territory. On the other hand, the data transformation approach is not restricted to preexisting n-grams, and it establishes a direct competition between the given sequence and its counterpart, to keep only the most fluent, thus acknowledging local preferences. These key differences are confirmed experimentally on English-French data: PoSLM-based reordering lowers the precedence rate of ADJs to NOUNs from 93% to 60%, while the rate varies by less than 1% in Søgaard (2011)'s approach, leaving the majority case adjectives still under-trained.

Finally, detailed analyses reveal that the PoSLM reordering approach has lower improvements than the WALS-based one on *easy reorderings* such as the nearly deterministic Hebrew-Bulgarian adjectives (UAS $\begin{bmatrix}\textsc{Noun}\\\textsc{Adj}\end{bmatrix}$ of 93.2% with WALS, versus 86.1% with the PoSLM). This suggests that the data-driven technique still wastes part of the available knowledge: indeed, the use of a probabilistic model to rank reorderings does not guarantee that any interesting reordering is in fact selected. Another advantage of the knowledge-rich approach is that the distribution of *local* word orderings is easier to control, since it explicitly regulates the balance between co-existing word orders. Indeed, when two structures are possible and fluent, the PoSLM-based method will always prefer the majority class. While the projectivity requirement generally softens this hard constraint by discarding many reordering candidates, the effect holds for instance on typologically close languages: during transfer from French (PR = 28% for ADJ-NOUN) to Italian (PR = 32%), PoSLM-based reorderings harden the preference down to PR = 16%, and end up under-training the ADJ-NOUN minority class.

In spite of this, the PoSLM reordering method is still competitive, since it covers more diverse phenomena. For instance, when transferring from English to Tamil, the UAS $\begin{bmatrix}\textsc{Verb}\\\textsc{Aux}\end{bmatrix}$ only raises from 31.4% to 35.0% with the WALS-based method, but achieves a nice 91.2% with the PoSLM. Such improvements are however less predictable and unexplained losses also occur, as for the UAS $\begin{bmatrix}\textsc{Verb}\\\textsc{Aux}\end{bmatrix}$ in Hungarian-Tamil (98.5% for the baseline and WALS, 66.4% with PoSLM reordering).

These results suggest that both approaches have their upsides and downsides, which remain to be combined, either at the language level, by using PoSLMs for the poorest sources and WALS otherwise, or at the token level, by confronting both reordering proposals for each dependency.

8.4 Qualitative comparison of relevance models

Both Søgaard (2011) and Rosa and Zabokrtsky (2015) agree on the idea that even distant sources can contribute to target processing, if appropriately handled, and that they should not be ruled out right away. I interpret their strategies through the prism of relevance; as such my proposal can be viewed as an extension that pushes further, and to a finer grain, the intuition of extracting useful knowledge even from poor sources. While Rosa and Zabokrtsky (2015) reward good source *languages* and Søgaard (2011) rewards target-relevant *sentences*, I reward relevant *local patterns*, by performing a local reordering of target-irrelevant parse subtrees rather than ignoring the whole sentence. Thus, the knowledge embedded in these subtrees is used more effectively, but the rest of the sentence as well.

To see this, consider the case of transferring an English parser to a language in which no verb is labeled as auxiliary.[13] In this case, the method of Søgaard (2011) is likely to discard all English sentences containing auxiliaries and the parser will hardly see, at training time, sentences involving passive constructions or past participles; by contrast, methods based on data transformation would not remove the full sentence, but just the auxiliary – all the other dependencies, e.g. the by-agent, can still contribute to learning. Similarly, because the absence of auxiliaries affects the distribution of trigrams, Rosa and Zabokrtsky (2015)'s method will reduce the contribution of the English source as a whole, even when the current input is a noun phrase and does not involve auxiliaries at all.

Thus, in comparison to previous works favoring close word orders at the cost of discarding some training examples or reducing source contribution, my proposal differs by improving cross-lingual transfer *without knowledge loss*.

8.5 Conclusion: beyond the proof of concept

This chapter has exhibited the adverse effects of typological differences on the effectiveness of direct transfer, both theoretically and empirically: language-independent knowledge loses its genericity when embedded in concrete data, which prevents transfer at the classifier level when source and target typologies differ. To address such phenomena, I have proposed a preprocessing technique, which transforms the source data to fit target typology before training, and can thus be entertained with any parsing system. In practice, I have experimented with two approaches for source reordering or rewriting: one is data-driven and exploits a PoS-based target language model, and the other is heuristic and relies on the WALS typological database. Both yield significant improvements (+2.3 and +2.9 on 40 languages), but the knowledge-based method proves more effective, with the additional benefit of being entirely resource-free and thus readily usable for over thousand languages. Notably, even if the current proposal for WALS-based rewriting targets only a few frequent PoS classes, its improvements on such classes are huge, often 30 and up to 90 points.

Incidentally, these experiments shed a new light on earlier proposals to improve direct transfer. The recent efforts on annotation consistency indeed allow fairer evaluation, and this contrastive experiment unveils important differences in how each method handles specific patterns; for instance the selection-based technique is unable to address minor fluency issues like adjective-noun inversions, but reordering can. Besides, my approach has the additional benefit of preserving the amount of available knowledge in the data.

Despite already substantial gains and readily usable methods, this study is above all a proof of concept of how knowledge can solve representation issues,

[13] In UD this is, for instance, the case for Greek, Latvian or Galician.

which are the main reason why PSEUDO-SHARED knowledge is not SHARED. The set of WALS-based rewrite rules used here is indeed a baseline proposal, and can be completed in many ways, typically with article insertions, PoS substitutions and patterns involving verbs, which were not considered so far. Besides, word order may not be the only reason of incompatible knowledge representations, and other kinds of knowledge databases could be exploited. Notably, since knowledge always appears as instantiated in data-driven methods, the effect exhibited here on dependency parsing presumably concerns many other NLP tasks, and not only the order-dependent ones.

The question of how to maximize the effectiveness of parallel data and projection methods also remains open: while they tackle such word order issues by design, they have other shortcomings related to typological differences. Typically, the syntax-semantics association depends on the language, which makes semantically-driven attachments harder to leverage for syntactic tasks, as exemplified in Chapter 5. It would be interesting for instance to leverage the knowledge that one language prefers the passive voice over the active one, in order to amend some alignments or alter the projection step.

Chapter 9

Cascading models for multi-(re)source selective transfer

***

Contents

Chapters 5 and 6 have motivated the design of a cascading framework in a cross-lingual context, Chapters 7 and 8 have brought technical improvements that made this approach achievable. Time has now come to put cascading into practice.

There are indeed two aspects, in the main contribution of this thesis: the development of a cascading framework for parsing, and its effective use, which mostly comes down to devising the metrics that will parameterize it. But I believe the core of my contribution to be the framework itself, which can be used in many other ways than those described here and opens new avenues for future research. This chapter proposes one way to exploit it, i.e. one set of metrics and architecture choices, and most of the reported experiments concern that setting, but throughout the text I suggest many variations on my use of cascading; experimenting (or further experimenting, in some cases) with those appears as the most promising track for future work.

The main motivation underlying this development was actually twofold: enabling multi-source combination to be more selective and fine-grained, and designing a flexible system able to combine seamlessly multiple and diverse resources for low-resource NLP. But as my study unfolds, having developed a quite generic system, it turns out that its possible applications are in fact numerous: multi-system combination, domain adaptation, and presumably many others. Hence, the scope of that contribution exceeds that of cross-lingual parsing.

The cascading approach is described more precisely in Section 9.1, which also highlights its differences with other ensembling methods and presents some technicalities of its application to cross-lingual parsing. Section 9.2 then fills the sole gap that remains at this point to perform cross-lingual transfer with cascading, namely the ability to measure cross-lingual relevance; depending on the available resources, four approaches are proposed and empirically illustrated. Section 9.3 hints at the diversity of other applications that my cascading framework can have, with three monolingual examples. I conclude this chapter by reporting, in Section 9.4, my participation in the *CoNLL 2017 UD Shared Task*: this evaluation campaign offered a nice opportunity to assess in realistic conditions the efficiency of my cascading approach, as well as several other contributions.

9.1 Transfer cascades

9.1.1 Core method

Figure 9.1 summarizes the cascading approach that I have proposed to address cross-lingual parsing: a first parser gets a raw input and produces a partial annotation, which is forwarded to the next one as an enriched input; then the second parser uses that additional information to predict a new partial tree, which must include all previously predicted dependencies; several similar stages may follow, until the last one, which also receives a partially annotated example but this time outputs a complete tree. This way, each token is annotated by exactly one model. At each stage, the decision between annotating a token or leaving it unannotated is taken by the parser itself, based on a criterion that was specified at training time: for instance if, between the target and a given source, only the prepositional phrases behave similarly, I will specify the delexicalized parser trained on that source to focus on attaching adpositions (and ignore all other tokens).

Since each model is supposed to learn a specific part of the syntax, these subsets of dependencies must be known beforehand. Following the vocabulary used by Kuncheva (2004, chap. 6) when reviewing methods for classifier selection, I refer to them as *competence regions*, and I will describe in Section 9.2 the method advocated by Kuncheva (2004) to *preestimate* them,[1] as well as a few alternative proposals. The strategies to choose and order the sources (that is, specifying their correspondence with stages) are also discussed in Section 9.2.

In order to ensure train-test consistency of the partial annotations used as constraints, the same architecture is used at training time: each model is trained under the constraints predicted by all previous stages, so that training is also performed stage by stage.[2]

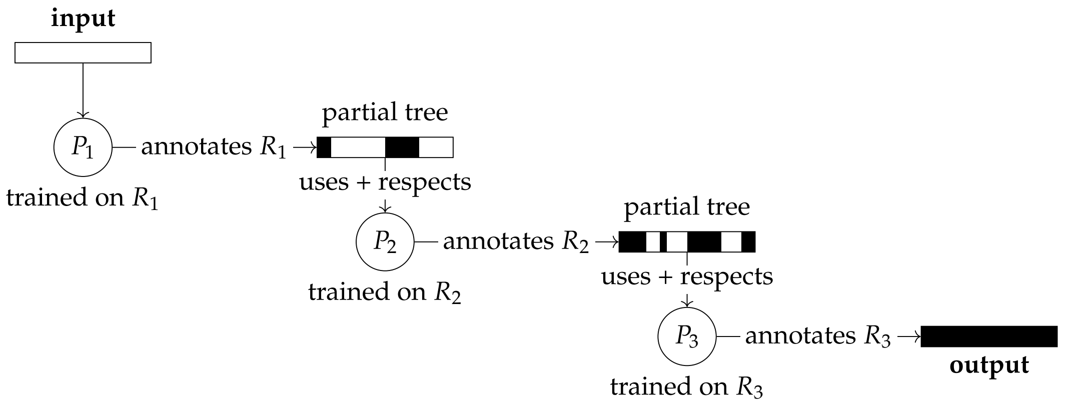


FIGURE 9.1: Processing of an input sentence by a 3-stage cascade. The cascade contains parsers P_1, P_2 and P_3, which have been respectively assigned competence on subsets R_1, R_2 and R_3; each token belongs to exactly one of them. On the input sentence, the white areas represent tokens whose head is unknown, while the black areas represent tokens whose head has already been predicted.

The main mechanisms underlying this method, i.e. the ability to parse (and train) given a set of dependency constraints (in the form of a partial annotation) and that of annotating only a specific part of the input, have been made possible by

[1]Kuncheva (2004) actually mentions other options to assign competence, where the choice of the competent classifier depends on the prediction (using e.g. classifier confidence), or on the input only (e.g. by evaluating each model on the K nearest neighbours of the input). But the need to know competences before training makes such dynamic estimation out of question in the case of my transfer framework.

[2]In theory, this makes the training time quadratic in the number of stages: treebank number k is preannotated by $k-1$ models. However, the processing time of a treebank (even with multiple stages) is small compared to the training time on the same treebank, so that the training time of a cascade is in practice close to linear (one complete training per stage). Still, the number of stages in the cascade (and thus, of sources) remains computationally constrained, and massive source sets (e.g. all UD languages) are not desirable: in my experiments, I have not exceeded a dozen of stages.

the extensions of the parsing framework proposed in Chapter 7. The applicability of the method has then been extended in Chapter 8: addressing typological divergences turns even bad sources into relatively competitive ones and thereby makes their knowledge more accessible to the target; this way, in languages where really only one type of dependencies is relevant (say, attachment of adpositions, as in the previous example), the system becomes able to single out that knowledge.

9.1.2 Relations with the meta-learning literature

Formally, my proposal is grounded on the cascading meta-algorithm (Alpaydin and Kaynak, 1998; Kaynak and Alpaydin, 2000), an ensembling method which improves a classification task by using a *sequential* pool of diverse classifiers: the input is fed to each one of them in turn, until one classifier is confident enough to classify it. So, each stage of the cascade can either classify the input, or refuse to do so and let the next stage annotate it. To apply this idea in a context of structured prediction,[3] I let each stage decide, for *each token* of the input (and not for each input), whether it should annotate it or not.[4] The notion of input enrichment along the cascade (based on the tokens which *have* been annotated) is thus an addition of the structured application, and so are the mechanisms of constraints and partial outputs.

Although the original motivation of the cascading approach was specifically to reduce computational costs (by running the most complex systems only for the most difficult decisions), this chapter widens its perspectives by aiming at accuracy improvements (by using each model where it performs best).[5]

The proposed cross-lingual application also departs from the standard framework in the mechanisms at work when accepting or refusing to annotate a token. In contrary to the traditional cascades where the decision to annotate an input can be based on the classifier's *own* confidence, in cross-lingual contexts the competence criterion will necessarily be *extrinsic*: English data alone provides no insight on e.g.

[3]As a matter of fact, Weiss and Taskar (2010) already propose to extend the cascading approach to structured prediction, and formalize what they call *structured prediction cascades*: cascades which prune the (structured) search space at each stage. However, their proposal is in practice much closer to the approach of coarse-to-fine NLP (Petrov, 2009) than to mine: while I reduce at each stage the *number of tokens* to annotate, they reduce the number of candidates *for each token* at each stage. For instance, the first stage of their cascade could decide for each adjective if it attaches on the left or on the right, and the second stage would predict the actual attachment. In my proposal, a first model would annotate all left-attaching adjectives (leaving the right-attaching ones unannotated), then a second model would annotate all right-attaching adjectives.

[4]Taking this kind of decision is especially complex for a transition-based parser: my remark on action-edge correspondence applies again, so that the decision of non-attachment does not result from one specific decision of the classifier (as is the case in the original proposal) but from a series of decisions throughout the derivation.

[5]When cascading parsers, speeding up annotation is out of question anyway: since each stage builds a full derivation, processing time is in fact *increased* by a factor around the number of stages.

English-Romanian similarity, it has to be provided by a third-party computation.[6] Conversely, in my proposal the decision on which tokens to annotate is taken *internally* by the parser (when focusing on a subset of the training data, it has also internalized the competence criterion), whereas in the original proposal the actual decision is not taken by the classifier, but using an *extrinsic* threshold.

Alternative architectures The meta-learning literature proposes several other ensembling methods, among which some might have been used instead, in order to combine the predictions from multiple sources with the target. But as explained in the following, none of them would have fully matched the objectives I have sketched in Chapter 6.

Stacking methods, like Cascade Generalization[7] (Gama and Brazdil, 2000), are the closest to cascading, in the sense that successive predictions of various systems are used to gradually enrich the input. However, in stacking those predictions are rather *proposals*, which each stage can amend, and it is up to the last stage to perform the final prediction, which may or may not match the earlier proposals. In my setting, I prefer to avoid calling into question the previous predictions: I do not want a parser specialized in language-specific dependencies to try learning the language-independent ones, and I consequently dismiss the stacking approach.

Another possibility, in a perspective of specialization, is the multi-expert approach, where each system annotates a handful of tokens and their outputs are then collected and merged. However, using this method would neglect the fact that some predictions can hardly be made without access to information from other predictors.[8]

A third option would be to perform standard predictions separately and combine them *a posteriori* (which typically corresponds to MST reparsing); but this would have failed to address the fact that in a cross-lingual context, the prediction of relevant dependencies is polluted by the irrelevant part of the data.

Cascading appears therefore as the only ensembling method that allows at the same time to specialize the models, eliminate irrelevant, potentially detrimental information, and let the models exchange some information with each other.

[6]Intuitively, an English speaker, who never heard of Romanian, its language family, or any of its properties, has no idea of whether they will be able to understand (even partially) a Romanian sentence, at least not until they actually encounter such a sentence. The same goes for a parser to which only English data is provided.

[7]Despite its name, which refers only to its layered architecture, this method is not a cascading algorithm in Alpaydin and Kaynak (1998)'s sense: previous predictions are only used to enrich the input and are not enforced in the final output.

[8]Such information is needed for instance to attach DETs before considering NOUN-VERB dependencies, but also to ensure compatibility of all edges to form a tree.

9.1.3 Algorithmic refinements

The above description has presented the core of the cascading approach, but several additional mechanisms can be built on top of the main architecture. They are merely proposals though, and not requirements of the core method.

Kuncheva (2004) mentions for instance the issues raised by *ties* when measuring competence for a given input. In my framework, there are easy ways to address that: if several parsers are deemed equally relevant for a given class of dependencies, they can be placed at the same stage in cascade, train and predict in parallel, and then vote on the disagreeing dependencies, using MST reparsing.

On another matter, it may happen that a dependency, supposed to be predicted early in the cascade, is missed; in such cases, all subsequent models will be incompetent to handle it, as they focus on very different parts of the syntax, and the final attachment of that token will be virtually random. In order to make the cascade robust to such missed dependencies, I propose to add small *overlaps between regions*, by means of two tricks: assigning each class to at least two models (typically, the two most relevant sources) and applying dropout on constraints during training (that is, removing for instance 10% of all constrained tokens at each stage).[9]

Conversely, sometimes a parser will output a few noisy dependencies, on top of those belonging to its region. To help distinguishing those, *confidence mechanisms* can be wrapped around the parser's own knowledge of its competence region: a dependency that is unstable (i.e. that has been output by one particular parser, but is absent from other hypotheses at the same stage) is more likely to result from noise than to belong to the competence region. One possibility to assess the confidence (or stability) of predicted dependencies[10] is to use MST-based voting on a pool of parsers, trained in parallel at the same stage: for instance for a pool of 5 parsers, one can retain only the dependencies predicted by 3 of them, and use MST to extract a stable tree from those. This ensembling procedure is computationally expensive, though. At the level of a single model, such MST reparsing can also be performed over the set of beam hypotheses, in order to discard the most unstable predictions of that particular model.

On a final note, sometimes the dependency constraints cannot be fully respected by a parser, for instance if they are non-projective,[11] so that some information gets

[9]This proposal contradicts my earlier claim that a model should not be concerned at all by classes attributed to previous models. But if the dropout is small enough, I believe that the extra dependencies to handle will not weight enough in the training material to pollute the knowledge acquired on the parser's main focus.

[10]Several methods exist to compute confidence scores at the level of the classifier, even in the specific cases of structured prediction and transition-based parsing (Crammer et al., 2009; Mejer and Crammer, 2010; Haulrich, 2010). However, once again, the lack of action-head correspondence hinders their effective use in this context.

[11]With the oracle refinements described in Section 7.2, non-projective constraints can be exploited by a projective parser, but it will remain unable to output them. The final output will consequently

lost along the cascade. But an MST-based trick (where dependency constraints are more weighted than actual outputs) can solve that easily, by *enforcing the constraints* after each step.

9.2 Concrete measures of relevance

This section proposes several concrete ways to preestimate the competence regions, which remain the last missing piece needed to use my transfer framework in practice.

This cascade of models has been designed, above all, for multi-source transfer and for the combination of cross-lingual and target resources. In that context, the core of computing competence regions is to estimate, for each dependency class in each source, its relevance for the target; then only a few aspects remain to be addressed, like the order in which the languages fit into the cascade, and threshold criteria for the actual attribution of classes to each source.

Regarding the definition of dependency classes, I advocate using class criteria which refer directly to input features,[12] like the child and head PoS tags, or their distance and relative positions, which make it easy for the parser to distinguish classes.

As for the ordering of languages, I propose to start and finish with the target language, so that the input is preprocessed with all easy dependencies[13] (using the target treebank to approximate hardness measures, as in Figure 6.8), and any missing information at the end of parsing is fetched in the right language. This leaves in between the cross-lingual models, whose ordering requires language-level scores. To ensure that the first annotated tokens (which impact the whole processing afterwards) are handled by good sources, I advocate here to use relevance measures at the language level.[14]

Depending on the available resources (target data with or without annotations, or no data), I propose several approaches to estimate both class-level and language-level relevances.

include only a projective subset of those dependencies, the others have been ignored when constraining the search space.

[12]Relation labels would have been a natural choice, since their link with the underlying syntax is intuitive: one can say that 'subjects behave similarly in French and English'. But the issue with that criterion is that labels are not part of the inputs, so that in order just to learn its own region, the parser would have to learn to predict labels on raw data, which is almost as difficult as parsing itself.

[13]Section 7.5 has pointed out the difficulty of training accurate parsers on very sparse data, therefore the first model of the cascade should already have a significant coverage. If there are not enough easy dependencies, this stage could also cover a few harder ones.

[14]Another possibility would be to rank them from the biggest to the smallest competence regions, which increases coverage in the early stages of the cascade and thus minimizes the sparsity that each partial parser faces; this strategy does not guarantee, however, that the early stages are reliable sources. In practice though, relevant sources will often be those with a large competence region.

9.2.1 Using a development set

Kuncheva (2004) has an empirical approach to competence regions: the purpose of measuring relevant classes is to ascertain which ones will yield high accuracies on test data, so why not directly measure that, on a sample of annotated target data? The method described by Kuncheva (2004) consists in training all models on the whole data, grouping the possible inputs into classes,[15] and then attributing each class to the model which performs best on that class. I propose to follow this procedure, even though it is an expensive way to compute competence regions: since I intend to retrain each model after computing the regions (which is not the case of Kuncheva (2004)), this method *doubles* the number of parsers to train, thereby increasing a lot the training time.

This relevance criterion has also a natural counterpart at the language level: the overall accuracy on the development set, which states indeed whether the corresponding treebank is a good source.

One advantage of using this empirical approach is that it applies similarly to the target and other languages: if target data is available on top of the development sample, a target parser can be trained and evaluated as well – a class will simply be attributed to the monolingual model whenever it outperforms all cross-lingual parsers. On the contrary, the other methods listed below are based on explicit similarity measures, and thus cannot be used for the target, which would always get a perfect score.

The bottleneck of this method is the size of the development data, expected to be small in scenarios requiring cross-lingual transfer. If that data is too sparse, it can result in incorrect estimation of relevance, either because the dataset is biased, or because scores computed on very few occurrences (typically, a score of 100% on a class with a single occurrence) are not representative of actual prediction quality. In order to avoid unwarranted assignments, I advocate to leave unassigned any class which has too few occurrences in development data (less than 5) or always low accuracy (under 20 UAS). By design, the last parser of the cascade (in my experiments, the target one) always performs full predictions and consequently takes care of such unassigned classes.

9.2.2 Treebank similarity with a target sample

Still in the case where a target piece of data is available, other measures of relevance can be entertained without involving trained parsers, and thus without requiring

[15]Kuncheva (2004) builds them in an unsupervised way, using the K-means algorithm on raw data. Such procedure is not necessary in the context of parsing, since dependency classes have already been identified, based on properties like PoS tags or direction.

additional training:[16] the idea is to directly measure the similarity of the source treebank with the target sample, but at the level of a dependency class. As explained in Chapter 5, such similarity metrics (at least in a context of cross-lingual parsing) should involve both the words and the annotations.[17] The goal is thus to quantify, for a given input, how similar its source-side attachments are with those in target.

I consequently propose to measure relevance with the similarity of the attachment distributions, where attachments are characterized by their child PoS, head PoS, direction and dependency length (which I cap to 4, to avoid long-tail effects). I compute it with the Jensen-Shannon divergence,[18] so that it yields in fact a measure of *irrelevance*: the irrelevance of a child-head pair of PoS is the divergence between distributions of (length, direction) values in both languages. The irrelevance based on the child PoS only is computed similarly, but with (head PoS, length, direction) values.

The motivations to characterize attachment preferences in this manner can be illustrated on English-German auxiliaries. Most English AUX attach to a VERB (57%) but they can also relate to predicative ADJs (23%) or NOUNs (13%). And when they modify a VERB, that VERB is almost always close (distance 1 or 2) and on the right of the AUX (53% of all AUX). In German, on the contrary, AUX depend almost always on VERBs (97%), but only 10% of them depend on a VERB that is close on the right. Rather, 46% of AUX depend on a distant VERB on the right (distance at least 4), and 29% on their left neighbour. Syntactic differences between both languages appear clearly: AUX have different distributions of head PoS, and even for those with the same head PoS (VERB), the head is rarely in the same location. Hence, the AUX class can be assumed to rely on different linguistic mechanisms, and it should be deemed as irrelevant (in both transfer directions).

With the proposed metrics, the irrelevance of AUX in English-German is 0.49 (0.04 for English-French AUX, 0.67 for English-Japanese AUX, which is consistent with intuition), 0.45 for the AUX-VERB pair (0.02 in English-French, 0.67 in English-Japanese), but it drops to 0.16 for AUX-ADJ (0.03 in English-French, 0.67 in English-Japanese): this states that even though their verbal forms are built differently, both languages use predicative adjectives in the same way, which seems linguistically consistent and validates the approach.

[16]The main drawback of this method compared to the previous one is that it accounts for the knowledge that is embedded in the data, but not for that which is actually extracted from the treebank. Neither the size of each source treebank, nor the chosen parsing and training strategy are considered, although they both impact the actual efficiency of the transferred model. But such regions are undeniably faster to compute.

[17]Otherwise, when measuring similar occurrences of the 'NOUN VERB' sequence in both languages, one could deem the source subjects (NOUN$\curvearrowleft$VERB) as relevant to identify the root of relative clauses (NOUN$\curvearrowright$VERB).

[18]The Kullback-Leibler divergence is indeed known to be ill-defined in case of null probabilities, which I expect to be numerous when working on a mere sample of target data. This issue can be solved by smoothing, but the Jensen-Shannon divergence does it in a heuristic-free way:
$\mathrm{JS}(P\|Q) = \frac{1}{2}\left(\mathrm{KL}(P\|M) + \mathrm{KL}(Q\|M)\right)$, where $M = \frac{P+Q}{2}$.

Since this measure cannot be reasonably used to estimate the target region (target-target similarity would always exceed source-target similarity), I set a threshold at 0.3,[19] above which the class is considered irrelevant, and should be learned in target.

The KL_{cpos^3} metric already provides a data-driven similarity measure at language level, and it generally yields quite intuitive results; but since it takes only the words into account, it remains prone to the aforementioned confusions. It could, however, be improved by disambiguating the PoS trigrams by keeping their internal dependencies (thus computing the distributions of sequences like ‘DET$\curvearrowleft$NOUN SCONJ’). Another option – which has the double benefit of extending the class-level metrics and using less sparse distributions – would be to compute the similarity between the distributions of child-head PoS pairs for instance (or child/head/direction triplets).

9.2.3 Approximation with raw target data

In real-world scenarios, it is not guaranteed to have access to a target sample of dependency trees however, although I believe this case to be rare. Access to raw data (that is, at least PoS-tagged) is much more likely; but as already explained, similarity measures between raw corpora could be misleading, and the metrics must take the dependencies into account. In this scenario, dependency annotations are only provided on the source side, but this information may be sufficient to compute relevance scores: I propose a way to approximate treebank similarity, when only a source treebank and raw target data are available.

My approach is to evaluate whether the target sequences are at least *compatible* with the assumption that the source and target languages share the same dependency distribution. For instance, if most source-side DETs depend on a neighbour NOUN while the average distance from target DETs to their nearest NOUN is 4, it is unlikely that those DETs are relevant. Similarly, if all source AUX depend on a distant VERB on the right (as in German), while in target the AUX are almost never followed by a VERB, the source AUX can be safely assumed as irrelevant.

I thus assume that both languages have the same distribution of head PoS and dependency direction, and check that their average dependency lengths match.[20] However, since dependency length cannot be measured on the target side, I do not

[19]This value is motivated by measures on Japanese AUX: they have an irrelevance of 0.29 for Turkish – which is also an Altaic language, albeit distant – and 0.38 for German, whose verbs are indeed often placed at the end of the sentence, like in Japanese, but are governed by completely different mechanisms. Hence, a threshold between both values seems reasonable.

[20]Even though one of the corpora may have longer sentences in average, I assume that at least locally, both languages are similarly verbose. Dependency lengths are thus supposed to match without accounting for the sentence length.

use the true dependency lengths in the source, and prefer to resort to the *same* (approximated) measure on both sides.

In practice, say that p% of the source tokens with PoS P have a head with PoS P_h in direction d. In raw data (either the source or the target) I identify, for each P token, the closest P_h token in direction d;[21] among those dependency candidates I only retain the p% shortest ones (according to the short-dependency preference, they are the most likely attachments in the $P/P_h/d$ class)[22] and then compute their average length. By weighting this length with p and summing over all P_h/d values, I obtain an estimate of the average length of P attachments. Regarding English-German transfer of AUX, the average length estimated is 1.6 for the English source (which underestimates the true average, 1.8), 3.8 for the German target (1.5 for a French target and 5.0 for Japanese, which is consistent with intuition).

The relevance of the PoS class is then simply expressed as the length difference, which quantifies the mismatch between both distributions. I consider as irrelevant the classes for which that value exceeds 1.5, so that there is a tolerance of roughly 1 token: a 1-token difference corresponds indeed to minor local variations, like DET-NOUN dependencies in English-French (the DET attachment in the English 'DET ADJ NOUN' phrase is 1 token longer than in the French 'DET NOUN ADJ'),[23] but larger differences seem unlikely, as they would turn local dependencies into long-distance ones.[24] For English-German AUX, the score is thus 2.2, and the class is deemed irrelevant, as expected.

At the language level, a direct extension of that approximated similarity would be to sum the scores computed for each PoS, weighted by its source-side frequency.

9.2.4 Another use of typological knowledge

In a last scenario, there may not be any data at all for the target language, in which case only typological knowledge would be available. Such information can also indicate relevance: for a given PoS category, if all associated typological features are identical for both languages, it is likely that tokens with this PoS behave similarly on both sides, and that their annotations in source data are relevant for the target.

[21]When there is no such token, I suppose the worst case, where the token depends on the furthest token in sentence. This ensures that the cases where the source dependencies are simply not possible in target data are judged as irrelevant.

[22]This selection underestimates in fact the true lengths, first because sometimes a longer attachment is preferred (in English this is often the case with compound nouns: in 'DET NOUN NOUN', the DET depends on the second NOUN), second because among these short attachments, some are actually hindered by another attachment to a different head PoS, but I do not check for overlaps.

[23]The difference is even smaller in practice: the approximated average length for DETs is 1.4 for English, 1.1 for French. Their true averages are 1.7 and 1.1.

[24]Recall that around 35% of dependencies only are of length 3 or more: such dependencies are unlikely by themselves, getting them on a systematic basis and as the exact translation of short-range dependencies is even less likely.

The issue with this approach is that it overlaps with the rewriting method described in Chapter 8: many typological divergences, at least among those directly related with syntactic patterns, can be addressed by that method, so that the source data gets, in fact, the same typology as the target. As a result, the corresponding features cannot be used to distinguish relevant and irrelevant contents.

It may not be the case for all syntax-related features, though. Feature 124A of WALS (*'Want' Complement Subjects*), for instance, documents for many languages whether the subject of 'want' is implicit ('She wants to go') or overt (literally, 'She wants that she goes'); a broad interpretation of this feature would extrapolate general preferences for subordinate or infinitive clauses. As a matter of fact, I have mentioned in Chapter 5 that Romanian (in contrary to other Romance languages, and as an inheritance from Slavic languages) prefers in general subjunctive subordinate clause over infinitives; and indeed, feature 124A reports the subject as implicit in English, French and Spanish, but overt for Romanian and Bulgarian. On one hand, this feature seemingly indicates for which languages the subjects and verbs of subordinates are relevant; on the other hand, extracting rewriting rules from feature 124A (i.e. turning automatically any infinitive clause into a subordinate, and vice versa) seems non-trivial, so that such divergences will unlikely be addressed in the short term.

I thus believe that with some further work, it would also be possible to exploit typological knowledge, either feature 124A or another, either the WALS database or another resource, in order to evaluate class-level relevance.

As for the language level, there already exists a way to compute a similarity score based on typological features: the inverse Hamming distance of both languages' vectors, as proposed by Søgaard and Wulff (2012).

This section has introduced a series of concrete metrics to estimate the relevance of a given dependency class, or language, for the task at hand (parsing the target). Each one of those metrics is adapted to a different resource setting, which makes it possible to compute competence regions (and thus, to set up transfer cascades) in a wide range of low-resource scenarios, even the most devoid ones.

9.3 Monolingual cascades

The primary motivation for this cascading framework was to improve cross-lingual transfer with selective combination. I believe, however, that the scope of its applications goes way beyond that scenario.

Hence, I propose here three other ways to entertain cascading in a monolingual setting: a strategy for multi-system combination, a proposal for domain adaptation and a divide-and-conquer approach.

9.3.1 Multi-system and multi-domain cascades

By design, this cascading framework can combine several sources, several resources, but also several systems. It is indeed a well-known fact (reassessed in Section 5.2) that different parsing systems exploit training data in different ways, and hence excel on different parts of the syntax (McDonald and Nivre, 2007); so, why not use those parts as competence regions in a cascade?

For instance, a multi-system cascade could start by placing a few dependencies heuristically (typically for some very deterministic attachments, or to exploit a third-party lexicon of multi-word expressions), then annotate most of the input with a state-of-the-art neural parser like UDPipe,[25] and fill the remaining gaps with PanParser, which seems better at learning rare attachments in a one-shot manner and has the additional ability to exploit prior annotations.

Other applications can be found in the domain adaptation field. Similarly to the transfer cascade which exploits first generic knowledge (shared) then specific knowledge (target-specific), I propose to train a 2-stage cascade, with an out-of-domain model followed by an in-domain one – except that in this case the estimation of competence regions is made easier by the fact that both treebanks are in the same language.

In both cases, the process remains exactly the same as for transfer cascades: competence regions are computed using class-level measures (like the UAS on a development set), with the goal of estimating the strengths and weaknesses of each parser for the task at hand.

Experiments have been conducted with these two types of cascades, in the frame of the competition presented in Section 9.4.

9.3.2 Divide-and-conquer cascades

With further investigation, it appears that the cascading approach can even prove useful when the same treebank and system are used at each stage (i.e. in a purely monolingual context, with a single domain and using a single system), by implementing a divide-and-conquer strategy (henceforth, D&C).

[25]Using UDPipe in a cascade is not straightforward, because its implementation does not allow the use of constraints or partial outputs. As such, it can only be trained and used in a fully standard way. Still, its inclusion in a cascade can be simulated by filtering the outputs and discarding the dependencies which are inconsistent with the constraints. This way, it can still provide useful constraints to subsequent models based on PanParser.

Dependency length I have mentioned in Sections 6.2 and 7.5 the strong bias toward short dependencies, and my doubts regarding the need to train complex models for predictions as simple as the numerous attachments to direct neighbours. In their attempt to speed up parsing by limiting dependency length, Eisner and Smith (2005) suggest that '*a learner might start with simple structures that focus on local relationships, and gradually relax this restriction to allow more complex models*'. Such a strategy is totally achievable within the cascading framework: the first stage of the cascade would be a partial parser focusing on short dependencies and its output would be forwarded as constraints to the second stage, completing the predictions with long-distance dependencies.

As a matter of fact, the experiments regarding that idea have already been performed in Chapter 7, when evaluating partial parsers and constrained training. According to Tables 7.4 and 7.5, starting with very short (length 1) dependencies leads to minor drops on both short and long dependencies; but with a broader criterion (length $\leq$2), even though the short-only model still underperforms slightly the standard parser, the long-only model outperforms it by a large margin. It thus appears possible to improve parsing with a D&C strategy on dependency length. However, the benefits of that method seem to depend a lot on the language, and on a careful choice of the threshold between short and long dependencies, so that this promising strategy would require further investigation. While it departs from the core concerns of this thesis, this is an interesting track for future work.

General case I believe that the applicability of this divide-and-conquer strategy is not limited to length differences: the cascading framework applies naturally whenever the dependencies can be split into two categories, corresponding to two different types of predictions, and where the dependencies of one group may depend on the other, but not the other way around.

Although the system architecture is the same, the motivation in such cases differs significantly from that of the transfer cascade, though. The idea underlying a D&C monolingual cascade is to *split* the task, into subtasks, performed by submodels which are simpler, or more specialized: each submodel in the cascade is an expert on its own competence region. The approach adopted for a cross-lingual cascade is rather to *reduce* the task assigned to each source, to the region where the corresponding model outperforms all others; the discarded dependencies are effectively ignored, and no parser will be trained on them afterwards. So, the main goal in the cross-lingual case is to avoid polluting useful knowledge with inconsistent and irrelevant dependencies; it is only a fortunate side effect if the model becomes simpler by focusing on its region, whereas this kind of specialization is crucial in the D&C approach.

More precisely, the D&C cascading method can be beneficial in four ways: (a) with a reduced and thus simpler task, the first submodel may become more accurate;

(b) with many dependencies already predicted by the first model, the second one has access to much more information at parsing time; (c) the second submodel may also benefit from the ability to focus on a smaller task; and (d) if the first submodel actually becomes more accurate, it will propagate its good predictions to the second one, thereby improving its contextual features, and hence its predictions.[26] So far, Chapter 7 has assessed the ability of that framework to leverage effects (b), (c) and (d), although not systematically, while effect (a) remains elusive. Experiments with gold constraints have notably revealed effect (d) as particularly promising, with gains up to +10 UAS on the second region (i.e. +3.6 overall). Unfortunately, such gains remain purely theoretical as long as effect (a) is not observed.[27]

Regarding the choice of split criterion, the monolingual setting is again completely different from the cross-lingual one. While in a cross-lingual context that criterion is necessarily extrinsic (as explained in Section 9.1), in the monolingual case it can well be intrinsic (the parser can already know its strengths and weaknesses, by looking at its errors during training: no third-party information is needed). This enables the use of much more diverse critera: length, child-head PoS and direction, but also in-tree depth, relation labels, hardness, of even a fully empirical criterion, like correct versus incorrect predictions.[28] Unsupervised clustering may also be considered, even though the necessity to reason on tokens instead of prediction configurations remains.[29] Besides, while the most straightforward (and computationally cheapest) application of D&C cascades involves two submodels, it may also be entertained with any number of submodels, for instance to learn separately the short, medium and long dependencies.

Overall, the monolingual setting makes many new options available, and the computation of competence regions becomes much more straightforward than numerous measures of fine-grained relevance. The stakes are also lower: when some information is discarded in the cross-lingual case, it disappears forever from the

[26] Depending on the desired effects, there are consequently two strategies to decide how to split the competences (for instance, to fix the threshold between short and long dependencies): either choosing the subset of short dependencies that maximizes the gains achieved by effect (a) (and gains on the long ones will naturally ensue, thanks to effect (d)), or singling out just a few long dependencies, so that these difficult predictions become as simple as possible, thanks to effect (c) (but that approach is not particularly beneficial to short dependencies).

[27] There is, nevertheless, another way to benefit from effect (d): replacing the first submodel with another parsing system, designed specifically to excel on the first region (e.g. a local classifier for neighbourhood attachments, which dispenses with the tree constraints). Provided that this alternative system manages to outperform baseline parsing, big improvements on the whole sentence will surely follow.

[28] I have explored an approach inspired by boosting, where one or several base parsers (for instance trained on a sample) are used to preannotate training data, thereby identifying the most difficult dependencies (on which most of the base parsers fail); they are then attributed to the second submodel, while the first one takes care of the rest. Preliminary experiments with this method have yielded promising results, but they remain to be confirmed by future work.

[29] The main reason is the lack of a one-to-one correspondence between actions and edges, but the impossibility to *observe a priori* the distribution of configurations (which depends entirely on the behaviour of the parser at prediction time, and thus on the learned parameters) precludes even more any proposal to cluster the configuration inputs.

available knowledge, whereas in a D&C cascade it is merely reassigned to the next model. But the choice to split the data is in fact strong, in the sense that it is not prescribed by factual properties of the language (like relevance is). It really is a choice, and not a trivial one. Walking down that path promises substantial matter for future work.

This section has proposed several ways to extend my cascading approach to other use cases: improving monolingual parsing, system combination and domain adaptation. It sketches the broad range of applications that can be investigated in future work. Besides, an extra option resides in the possibility to *combine* these different types of cascades: one can for instance envision a transfer cascade where each source is split into several submodels, or a transfer cascade incorporating several systems. On top of multi-system monolingual cascades, the next section also reports experiments performed with multi-system transfer cascades.

9.4 Experiments in realistic conditions

My participation to the *CoNLL 2017 UD Shared Task* (Zeman et al., 2017) was the occasion to evaluate in realistic conditions the success of my cascading framework, but also several other contributions: my PanParser implementation and its global dynamic oracle (Chapter 7), partial projection with partial training (Chapter 7) and WALS-based rewrite rules (Chapter 8).[30]

This large-scale evaluation (81 test sets, 45 languages) on UD data was more realistic than usual experiments, in the sense that parsing was done on raw input text: phrase segmentation, tokenization, PoS and morphological tags had to be predicted. The task also included 4 surprise languages, for which the available data (samples around 20 sentences, released one week before test phase) corresponds precisely to the amount of resources that I target to exploit. Realism was additionally ensured by the use of a blind-test platform for the evaluation. Parses were evaluated with F1 scores on labeled dependencies, which allowed me to extend my previous work with the new dimension of labeling.

This challenging task included many other aspects, like the computation of accurate word embeddings, the diversity of treebank sizes, some provided with and other without development data, as well as domain adaptation considerations. However, I chose to focus on the low-resource aspects of the task, i.e. on the treebanks under 1,000 sentences. For a complete description of implementation details and

[30]Still, not all my thesis contributions could be included in that evaluation, as some of them were posterior to the end of the shared task. This is notably the case of the hardness measures, so that the cascades presented here do not include stages dedicated to easy dependencies.

shared task results, I refer the reader to the system description paper (Aufrant and Wisniewski, 2017).

This paper presents notably two other contributions, which I describe only briefly here. First, by carefully postprocessing the baseline tokenization provided by the task organizers, I have managed with very simple means to outperform all other teams on that aspect, thereby ranking 1st (out of 33 teams) on tokenization and 2nd on complete preprocessing (including morphological tags). Second, in the course of analyzing my results after the evaluation, I have investigated in details the cross-treebank consistency of UD, notably among treebanks of the same language, and revealed several preprocessing inconsistencies and the large extent to which they affect the results. This analysis has given me a valuable experience of the difficulty to build and use consistent annotation guidelines, and of how long the remaining path is.

Overall, according to the official results, my system achieves 67.72 LAS and is ranked 17th out of 33 participants, while the baseline proposed by the task organizers (UDPipe 1.1, by Straka (2017)) achieves 68.35 LAS and is ranked 13th. However, this result is penalized by particularly bad accuracies on the domain adaptation test sets (largely explained by cross-treebank inconsistencies). As these test sets were not relevant to the focus I had when tackling this task, I have also computed the score I would have achieved without these drops: 68.90 LAS, which would have given a rank of 9th. This much better rank proves the ability of my proposal to improve over a state-of-the-art system, and thereby validates my cascading approach, as well as PanParser's design.

In the following, I successively present the main components of my system, how I build cascades upon them, and per-treebank results and analyses.

9.4.1 System components

The system designed for this task implements several strategies, summarized in Figure 9.2: depending on the treebank size, I build and compare several parsers with various designs, and finally submit the parser or combination of parsers that performs best on development data. In that regard, the approach is exactly the same for all treebanks: for each one of them (including the surprise languages), I make sure of having training, tuning and development data – they only differ in size.

As shown in Figure 9.2, by default I compare several monolingual parsers, and for small treebanks only, I also train a few cross-lingual parsers based on delexicalization, projection and multi-source combination. Monolingual and cross-lingual cascades are then built upon those parsers. I first present the base parsers, while technicalities of the cascades are addressed in the next section.

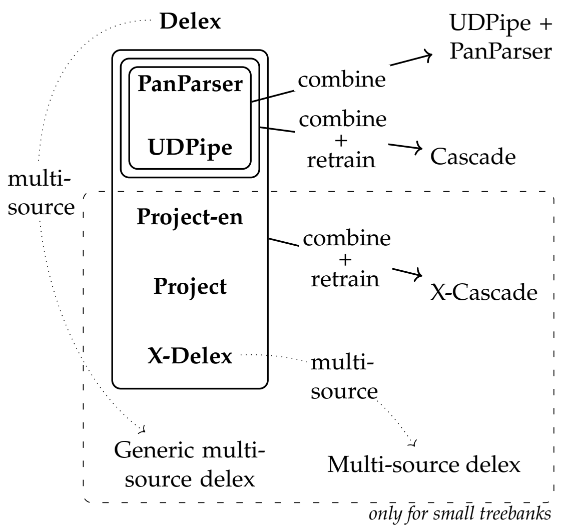


FIGURE 9.2: The various strategies contained in my shared task system. The strategies written in bold consist of a single model, the others are combinations of models.

Monolingual parsers As it served as the baseline of the task, **UDPipe** models were provided by the task organizers for all languages except the surprise ones; I have consequently resorted to those models for several languages which were out of my focus. I also consider **PanParser** models, either greedy or beam, and with optional use of morphological features and of the word embeddings provided by the organizers. Contrarily to my previous work, here parsing is labeled; relation labels are predicted in a second step, which enables to use features of the whole parse tree for this prediction. **Delex** is the delexicalized variant of PanParser, which may prove useful in monolingual context, for the languages with smallest treebanks. Finally, since relation labels are sometimes better predicted by PanParser than UDPipe, I also consider combining their outputs at prediction time (**UDPipe+PanParser**): the input is first annotated by UDPipe, then the predicted labels are discarded and replaced by those predicted by PanParser on UDPipe trees.

Cross-lingual parsers For small treebanks (under 1,000 training sentences), I additionally consider cross-lingual transfer, and start by identifying the presumably best source for each treebank, using a heuristic based on treebank size, WALS features and the KL_{cpos^3} metric.[31]

When parallel data with that source is available,[32] I build the **Project** parser with the partial projection technique described in Chapter 7. However, compared to the previous experiments, here parallel data is much smaller in general; in order to ensure that the projected parser is trained on enough trees, I thus adapt the sentence

[31]When the result was another treebank of the same language, I rather experimented with domain adaptation cascades, but this vanilla attempt was not conclusive.

[32]The shared task authorized the use of all parallel corpora available on the OPUS platform (Tiedemann, 2012).

filtering heuristic to be more permissive regarding coverage.[33] Since parallel data is often larger with English, I also project parsers from English (**Project-en**).

X-Delex is the delexicalized PanParser trained on the single-best source, but after WALS-based rewriting of the treebank as advocated in Chapter 8.[34] I train in fact such parsers on each treebank (except the target) and combine them into **Multi-source delex**, with an MST-based vote weighted by KL_{cpos^3}.[35] In order to select a subset of good sources, I experiment with three heuristics.[36] **Generic multi-source delex** is the language-independent version of that combination, without WALS-based rewriting (that is, it is built on Delex models), with unweighted votes, and including the target Delex model.

Training and tuning I always use gold tokenization and segmentation during training, but in order to improve robustness to noisy tags, all models (except the surprise ones) are trained on treebanks with predicted tags. Besides, the various parsing methods that I have presented involve a number of hyperparameters; they are tuned in all cases on the target tuning data, except of course for 'Generic multi-source delex' (the hyperparameters of the underlying Delex models are tuned on the corresponding source).

9.4.2 Monolingual and cross-lingual cascades

The description of my use of cascading in this task requires a few precisions on technical details. First, the competence regions are based on child-head PoS pairs, and assigned to each base parser according to its score on tuning data. Sometimes the output contains a few noisy dependencies on top of the dependencies belonging to the competence region. To help distinguishing those, I also use confidence filtering based on ensembling (that is, each stage is a pool of 5 similar parsers).

As in base PanParser models, relation labels are predicted in a second step, using features from the whole predicted parse tree. Since each label is predicted independently, the competence regions are computed only *after* training, without retraining: at test time I simply use the label predicted by the competent classifier. For each component separately, label prediction is trained on full data, preannotated with parse trees resulting from the whole cascade, and competence regions are computed for these label classifiers.

[33] I retain all projective trees with coverage over 20% if there are less than 5,000, trees with coverage over 80% if there are more than 5,000 (up to 10,000), or the 5,000 trees with highest coverage otherwise.

[34] In order to increase train-test similarity, especially for Uyghur and Kazakh whose tagging accuracy is particularly low (under 70%), I also experimented with artificial noise added to the source tags (either random or designed to match the error types of the target tagger) but it was not conclusive.

[35] Labels also result from a weighted vote, for each dependency.

[36] I try retaining only treebanks over 2,000 sentences, treebanks that minimize the number of rewrite rules, and the top 5 treebanks according to the KL_{cpos^3} metric; the best heuristic depends on the language.

Choice of components I apply the cascade combination method both in monolingual and cross-lingual contexts. In each case, I try out several subsets of components, training cascades for each subset, and tune this choice separately for the heads and the labels.

In monolingual settings (denoted **Cascade**), when the scores of UDPipe and PanParser on tuning data are close enough, I hypothesize that their views may complement one another, and train cascades with the following component candidates: either UDPipe followed by PanParser, or UDPipe alone, or PanParser alone. Hence, I compare 3 cascades on the head prediction task, and 3 computations of competence regions on the label prediction task.

In cross-lingual configurations (denoted **X-Cascade**), the cascades are trained with the best projected parser (either Project or Project-en, when they exist), followed by X-Delex, the target UDPipe and the target PanParser. I try removing one or both of X-Delex and UDPipe, thus comparing 4 cascades.

I have consequently experimented with transfer cascades, but also with multi-system cascades, as well as complex cascades combining both the transfer and multi-system approaches.

9.4.3 Per-treebank results

In the rest of this section, I present the strategy that was adopted for each treebank, along with detailed analysis on the results. The analysis is conducted separately for the large (over 10,000 training sentences), medium and small (under 1,000 sentences) treebanks. I do not report here the results on the treebanks for which I used custom tokenization, nor on the domain adaptation test sets; they are described in details in (Aufrant and Wisniewski, 2017).

Apart from the detailed evaluation of the cascading method, I focus on three aspects: the efficiency of PanParser compared to a state-of-the-art neural parser (UDPipe), the success of model selection depending on the size of development data and the impact of bad preprocessing.

Throughout the section, I report both the official results of the shared task (obtained on the TIRA platform), and my own measures using the official evaluation script on the released test files (Nivre et al., 2017b). The latter are displayed in italics.

Treebanks over 10,000 sentences Table 9.1 reports the scores obtained on the largest treebanks, and the corresponding baselines. It includes evaluation of both UDPipe and PanParser based on gold segmentation, for a better comparison of parsers (as they were trained, tuned and selected using gold segmentation), and an assessment of the LAS drop due to segmentation errors. In most cases, my submission is simply the baseline UDPipe, as I did not focus on improving the system for large treebanks. Thus, no improvement is reported on these treebanks.

Model selection proves successful on all treebanks except Latin-PROIEL and Dutch. In particular, it detects that the German PanParser is significantly better than UDPipe; considering the PanParser LAS on the other large treebanks, this is unexpected and remains to be investigated.

		cs	ru_syntagrus	cs_cac	la_ittb	no_bokmaal	fi_ftb	grc_proiel	fr	es_ancora	la_proiel
	Trainset size	65,070	46,373	22,304	15,017	14,911	14,231	14,103	13,825	13,589	13,482
GOLD SEG.	*UDPipe*	*83.76*	*87.55*	*82.47*	*77.74*	*83.94*	*76.07*	*70.69*	*82.15*	*83.97*	*67.22*
	PanParser	*78.65*	*82.41*	*79.80*	*74.78*	*83.38*	*75.49*	*69.03*	*81.33*	*83.50*	*65.36*
TIRA	UDPipe (F1)	82.87	86.76	82.46	76.98	83.27	74.03	65.22	80.75	83.78	**57.54**
	LIMSI (F1)	82.87	86.76	82.46	76.98	83.27	**74.04**	65.22	80.75	83.78	57.51
		es	no_nynorsk	de	hi	ca	it	en	nl	fi	grc
	Trainset size	13,477	13,465	13,412	12,638	12,466	12,196	11,915	11,713	11,606	10,902
GOLD SEG.	*UDPipe*	*81.97*	*82.71*	*71.33*	*86.84*	*85.46*	*85.90*	*80.72*	*71.10*	*75.41*	*56.16*
	PanParser	*80.58*	*81.04*	*72.98*	*85.88*	*84.55*	*85.25*	*79.64*	*70.10*	*73.87*	*52.45*
TIRA	UDPipe (F1)	81.47	81.56	69.11	86.77	85.39	85.27	75.84	**68.90**	73.75	56.04
	LIMSI (F1)	81.47	81.56	**70.89**	**86.82**	85.39	**85.28**	75.84	68.31	73.75	56.04

TABLE 9.1: LAS results on the large treebanks. The models selected on development data are underlined. For 3 languages, I selected other models: UDPipe+PanParser for `la_proiel`, a monolingual Cascade (using UDPipe and PanParser) for `hi` and `it`.

Treebanks from 1,000 to 10,000 sentences The results for the medium treebanks are reported in Table 9.2. Compared to Table 9.1, it also includes gold segmentation evaluation of the monolingual Cascades, when they were considered.

As treebank size reduces, PanParser is more and more often the best parser; but with smaller development sets, the number of failures to select the best model increases (12 out of 32 treebanks).

The Cascade model provides significant gains on several treebanks, outperforming both UDPipe and PanParser; it is indeed able to extract knowledge from the lowest parser and use it to improve upon the best parser (see Basque, Indonesian, Turkish, Gothic, French-Sequoia and Greek). However, these gains are not consistent, and despite confidence and tuning mechanisms, the method still requires empirical validation.

Treebanks under 1,000 sentences Table 9.3 presents the last group of UD treebanks, the smallest ones, including surprise languages.

For each treebank, I report several scores, now including Delex, which is a promising candidate to parse the smallest treebanks. When cross-lingual methods are used, I indicate the source treebank and the scores of the projected base parsers (except for the surprise languages lacking parallel data), X-Delex and their combination X-Cascade.

		pt_br	bg	sk	pt	ro	hr	sl	pl	ar	nl_lassysmall	eu	he	fa	id	ko	da
	Trainset size	9,180	8,461	8,058	7,914	7,640	7,304	6,154	5,795	5,771	5,738	5,126	4,978	4,558	4,253	4,180	4,163
GOLD SEG.	*UDPipe*	*85.79*	*84.55*	*74.23*	*83.10*	*80.57*	*77.51*	*81.20*	*79.31*	*73.85*	*81.34*	*69.23*	*78.31*	*79.82*	*75.02*	*60.04*	*74.98*
	PanParser	*84.38*	*84.13*	*75.90*	*81.72*	*80.55*	*78.21*	*81.64*	*80.42*	*74.20*	*81.54*	*67.45*	*77.83*	*79.33*	*74.34*	*62.19*	*75.24*
	Cascade		*84.09*	*75.19*			*77.28*	*81.24*	*79.88*		*80.48*	*69.31*	*77.60*	*79.30*	*75.18*	*59.60*	*74.92*
TIRA	UDPipe (F1)	85.36	**83.64**	72.75	82.11	79.88	77.18	81.15	78.78	65.30	78.15	69.15	57.23	79.24	74.61	59.09	73.38
	LIMSI (F1)	85.36	83.22	**74.45**	**82.19**	**80.11**	**78.02**	**81.37**	**79.95**	**65.86**	78.15	**69.21**	57.23	79.24	**74.78**	59.09	**73.85**
		sv	cu	ur	ru	tr	got	sv_lines	en_lines	lv	gl	et	fr_sequoia	sl_sst	el	la	en_partut
	Trainset size	4,087	3,916	3,840	3,657	3,500	3,217	2,601	2,601	2,197	2,162	2,149	2,119	1,816	1,578	1,133	1,035
GOLD SEG.	*UDPipe*	*77.28*	*72.56*	*76.74*	*74.45*	*55.93*	*68.98*	*74.94*	*73.64*	*61.15*	*77.46*	*59.87*	*82.10*	*55.22*	*79.92*	*43.81*	*74.08*
	PanParser	*77.14*	*74.58*	*76.26*	*76.07*	*57.41*	*69.50*	*74.68*	*73.87*	*60.99*	*76.74*	*60.96*	*81.76*	*56.62*	*79.58*	*44.17*	*73.89*
	Cascade	*76.99*	*72.67*	*75.69*	*75.17*	*57.59*	*69.69*	*73.98*	*72.91*	*59.85*		*59.37*	*82.62*	*55.49*	*80.05*	*43.67*	*72.80*
TIRA	UDPipe (F1)	76.73	62.76	**76.69**	74.03	53.19	59.81	74.29	72.94	**59.95**	77.31	58.79	79.98	46.45	79.26	**43.77**	**73.64**
	LIMSI (F1)	76.73	**65.64**	76.65	**75.65**	**55.23**	**60.94**	74.29	72.94	59.81	77.31	**59.80**	**80.55**	**46.71**	**79.38**	43.55	73.60

TABLE 9.2: LAS results on the medium treebanks. The models selected on development data are underlined. For `pt`, `ro`, `sl`, `id`, `ur`, `sl_sst` and `en_partut`, I rather selected UDPipe+PanParser.

For this group, since the development sets are very small, model selection is not reliable and misses the best model in 5 out of 12 cases. Additionally, the even smaller tuning sets lead for the cascades to poor estimation of competence regions. But the main challenge faced by cascades is noisy tags. Indeed, while my preliminary experiments on X-Cascade with gold tags were very promising, turning to predicted tags makes the X-Delex models very unreliable (as their only features are unreliable), and they cannot provide useful insights to the cascade anymore, in which case cascades lack interest.

Regarding Uyghur, Kazakh and the surprise languages, for lack of data I have decided to reduce the training set in favor of the tuning set, in the hope that better estimated cascades would compensate for worse base parsers. This proves to be a bad strategy in half the cases, and I end up underperforming the baseline by a large margin in Kazakh and Buryat.

However, X-Cascades still achieve significant improvements over their base

		hu	uk	fr_partut	gl_treegal	ga	cs_cltt	ug	kk	hsb	kmr	sme	bxr
	Trainset size	864	733	527	510	481	441	75	12	10	10	10	10
GOLD	*UDPipe*	*64.67*	*60.92*	*78.77*	*68.50*	*62.25*	*72.77*	*36.66*	*27.10*	*32.91**	*27.93**	*20.72**	*12.88**
	PanParser	*65.60*	*61.97*	*79.78*	*68.36*	*63.38*	*74.94*	*36.21**	*23.48**	*47.27**	*39.38**	*31.99**	*28.59**
	Delex	*61.97*	*59.79*	*74.14*	*66.01*	*59.44*	*66.42*	*36.54**	*22.75**	*46.44**	*38.27**	*32.84**	*29.73**
	Cascade	*65.46*	*61.64*	*79.13*	*68.75*	*62.55*	*73.51*	*35.83**	*23.99**	*46.55**	*37.80**	*31.35**	*25.91**
	Source	fi_ftb	sk	fr_sequoia	gl	id	cs_cac	tr	tr	sl	fa	et	et
SEG.	*Project-en*	*44.01*	*52.59*			*42.48*		*16.65*	*13.87*	*37.77*			
	Project	*36.61*	*55.52*			*38.46*		*23.27*	*23.08*	*43.27*			
	X-Delex	*41.42*	*54.75*			*38.80*		*22.89*	*25.63*	*62.20*	*43.32*	*37.58*	*26.16*
	X-Cascade		*57.43*			*60.09*		*37.35**	*22.35**	*54.61**	*40.26**	*37.79**	*30.13**
	Multi-source									*68.78*		*41.52*	
TIRA	UDPipe (F1)	64.30	60.76	77.38	65.82	61.52	71.64	34.18	**24.51**	53.83	32.35	30.60	**31.50**
	LIMSI (F1)	**65.18**	**61.68**	**78.30**	**65.85**	**61.94**	**73.49**	**34.70**	20.94	**57.79**	**35.59**	**31.03**	25.86

TABLE 9.3: LAS results on the small treebanks. For the last 4 columns (surprise languages), 'gold seg.' results use gold segmentation *and* gold tagging. '*' denotes parsers whose monolingual training data is smaller than the data used by the UDPipe baseline, hence important score differences. The models selected on development data are underlined. The Multi-source line reports the scores of two models: 'Generic multi-source delex' for `sme`, and 'Multi-source delex' for `hsb`, with the heuristic retaining only treebanks over 2,000 sentences.

parsers in Uyghur, North Sámi and Buryat. This suggests that cascading can indeed prove useful in cross-lingual contexts; with the appropriate amount of data and additional effort on estimation of competence regions and on robustness to noisy tags, it may convey much larger gains.

Overall, this evaluation has confirmed that PanParser is quite competitive for low-resource parsing, as it outperforms UDPipe on most small treebanks, but also many medium ones and even one large treebank. It has also assessed the benefits of the cascading method, which proves useful in many cases, with sometimes large improvements. However, the gains do not appear sufficiently consistent to be reliable and still require further work. The shared task conditions have indeed uncovered several challenges faced by the method: it lacks confidence mechanisms, delexicalized models are too unreliable in case of predicted preprocessing, and the lack of target data hinders accurate estimation of competence regions (although I have not experimented here with the alternative methods I have proposed for competence computation). Another regrettable outcome of tiny development data is that even when a method yields substantial improvements, it may well be dismissed in favour of a much less accurate approach. Low-resource parsing would consequently benefit from further work on the analysis of each approach's strengths and weaknesses, so that the parsing method can be chosen based on properties of the language, rather than data that does not exist.

9.5 Conclusion: the road ahead

This chapter has addressed practical aspects of cascading parsers: details of their implementation, choice of sources and their ordering, computation of competence regions and concrete measures of relevance.

The relevance metrics I propose are of four kinds, depending on the available resources: a target dependency sample and a pre-trained parser, a target dependency sample, raw target data, or pure typological knowledge. This way, even if overall my method relies a lot on the availability of a target annotated sample, it remains usable even without that specific resource, which was a primary concern in Chapter 4.

I have also emphasized on the variety of applications that this framework has, either by offering new ways to combine multiple sources, resources, systems, domains, or by devising new strategies to exploit a given treebank, when learning one type of dependencies can be singled out from learning the others.

Taking part in the *CoNLL 2017 UD Shared Task* has allowed me to confront many of my contributions to real-world conditions, where parallel data is small or nonexistent, tagging accuracy is way too low, and target data is merely a sample. It is a success, in the sense that the achieved improvements validate the ideas underlying my proposals and promise successful application in concrete use cases.

But this experience has also unveiled a few bottlenecks of my cascading method, which should be the primary focus of future work. First, it lacks reliability: additional confidence mechanisms to ensure that only high-quality constraints are forwarded, as well as ways to compute competence regions in the most resource-free, and yet reliable, manner. As of now, empirical gains are indeed not consistent enough and thus compel to resort to evaluation-based model selection, but that fallback solution is not reliable either when data is scarce. Second, a cascade proves only useful when combining several models that actually have useful knowledge to offer – and this is not the case of delexicalized parsers, in the not-so-uncommon use case where the preprocessing of the low-resourced target is unreliable, while at the same time parallel data is unavailable. Nonetheless, as long as these aspects are taken care of, a new avenue opens for future work, in the search for even more flexibility and diversity in the competence criteria.

Part IV

Conclusion

Chapter 10

Summary and perspectives

***

This thesis aims at enabling NLP for low-resourced languages, and therefore it has explored the increasingly popular strategy known as cross-lingual transfer, which consists in leveraging resources available for other languages. The main focus of my work is to maximize the applicability of that strategy, in real-world scenarios when such resources are numerous and diverse, but at the same time particularly sparse. Typical examples are typological information and annotated samples.

As a matter of fact, my thesis work unfolded in a quite particular context, where the NLP field underwent significant changes, chiefly as a consequence of the fast evolving field of Machine Learning. At the time I conclude this dissertation, the Machine Learning algorithms that are the most popular for NLP applications (mostly, recurrent neural networks and word embeddings, as exemplified in Chapters 2 and 3) have indeed few in common with those in use when I started that work (typically, Conditional Random Fields, Support Vector Machines, Maximum Likelihood Estimation and Expectation-Maximization optimization). To prevent the fruit of my work to become outdated too quickly, I have thus tackled aspects of cross-lingual transfer that are the least dependent on the underlying classifiers. Therefore, I have focused on data-side manipulations, leaving to future work the exploration of transfer approaches based on classifier-level changes.

Again with a concern for reusability, I have conducted my study in a way that is, as much as possible, agnostic to the considered NLP task. But in practice, the availability of substantial experimental data has led me to instantiate my proposals mostly on dependency parsing, although in theory they remain applicable to any sequence labeling task. Still, some parts of the document provide further perspectives on the tasks of PoS tagging (see Chapter 5 and Appendix C) and word alignment (Appendix B).

Empirical analysis and framework specifications Having set the frame of my work, I have undertaken an extensive analysis of the state of the art, both in cross-lingual transfer and in the specific context of dependency parsing; this study, conducted successively on the qualitative and empirical grounds, has allowed me to exhibit a series of shortcomings that I then chose to address.

Reviewing the literature of cross-lingual transfer in general (Chapter 2) and cross-lingual parsing in particular (Chapter 4) has indeed revealed that, notwithstanding the diversity of available resources for cross-lingual transfer, most works seem to focus on a handful of them, thereby overlooking valuable information from sparser resources, like samples of annotated data. As for transition-based parsing (Chapter 3), it also appears to be a rich and active field, for which many approaches have been designed; but all those methods lack a unified framework, notably for combining two successful lines of research, beam parsing and dynamic oracles. As a matter of fact, variants and combinations of the popular approaches have only been studied on a per-need basis, hence leaving many gaps in between.

Regarding empirical efficiency, Chapter 5 has reassessed that of the main transfer methods, by showing that despite promising absolute scores, even advanced state-of-the-art methods fail to outperform a parser trained on just a sample of data in the desired language, i.e. what would have been done before cross-lingual transfer came into play. Nevertheless, the same chapter has revealed that both kinds of parsers learn, in fact, quite different types of dependencies, which I have then further formalized, and characterized with systematic empirical means, in Chapter 6: cross-lingual parsers provide more information on the high-level structure of the sentence, while monolingual parsers trained on tiny data excel at the most deterministic types of dependencies, and more generally those with lowest Kolmogorov complexity – which is in fact consistent with intuition, but remained to be precisely measured. As such, they appear rather complementary than rival.

With this complementarity in mind, I have proposed in Chapter 5 a linguistics-oriented typology of the various types of knowledge involved in a context of transfer: the SOURCE-SPECIFIC pieces of knowledge, whose extraction from the cross-lingual resources should be avoided (except when they concern basic attachments which in turn help the extraction of other knowledge, or when they help in identifying the language-independent rule by taking care of the language-specific exceptions); the TARGET-SPECIFIC ones, which cannot be acquired elsewhere than in target data; and the SHARED knowledge, which should preferably be learned cross-lingually, in order to let the target system focus on its idiosyncrasies (except when it is significantly cheaper to acquire them in a more direct fashion in target data). An amendment is made however to the last category when I exhibit examples of knowledge pieces which in theory are shared, but in practice do not appear so, when considering practical transfer methods: a word that behaves similarly in both languages, but cannot be detected either by a common PoS context or via semantics-driven word alignment, can hardly be considered governed by shared knowledge. Thus, the category of PSEUDO-SHARED knowledge gathers what is only shared at the conceptual level, but not in terms of the knowledge pieces actually manipulated by the systems.

A series of low-level analyses completes this exploration (Chapter 6), by providing a better understanding of the inner workings of low-resourced parsers, either cross-lingual or not. This way, I have been able to identify concrete solutions to address the aforementioned shortcomings, and to propose a new transfer framework based on the cascading principle: each parser in play (that is, each system-resource combination) focuses on a given area of the syntax, for which its training material is most relevant, so that they annotate part of the sentence each; but they do so in a sequential fashion, where the most basic, or language-independent, attachments are predicted first, to be forwarded to the subsequent stages. Using the same process at training time, this progressive input enrichment simplifies indeed the task of the last and most complex stages, which can then truly focus on their own area.

Implementing cascading transfer At that point, the architecture of my framework proposal was fully specified; the rest of my work has thus been dedicated to its concrete application in a parsing context. This implementation was done in several steps: first by improving the underlying parsing algorithms for more flexibility (Chapter 7), then addressing a specific representation issue due to typological divergences (notably, word order), forbidding the use of distant sources even when they are partly relevant (Chapter 8); the actual use of the framework is then left for Chapter 9.

Along the technical improvements that were compulsory to set up my transfer framework, Chapters 7 and 8 have in fact also contributed to maximize the usability of resources: one has presented ways to exploit non-standard examples in treebanks, instead of discarding them as usual, on the basis of their sparsity (partial trees) or other properties (non-projectivity); in the other I have shown a valuable use of typological information (extracted from WALS).

To put this framework into practice, I have devised, and reported in Chapter 9, a series of metrics based on various resources, to evaluate which part of the syntax is shared by which languages; and as a final touch to this work, I have taken part in a challenging evaluation campaign, in which I could test all my thesis contributions in a realistic frame.

The outcome of that thesis work is thus primarily the cascading transfer framework, together with a software release, PanParser, which provides the technical basis of this method, but has also its own benefits: it notably enables a systematic study of algorithmic variants for parsing, and thus fairer benchmarking.

Additional contributions This work has also produced a number of otherwise valuable findings, which I did not exploit to their largest extent, as they did not belong to the core of my work; I believe however that they can serve as starting point for further studies.

The most advanced of these side contributions is undoubtedly my work on transition-based parsing algorithms (Chapter 7). While in the literature, dynamic oracles are only presented as a means to improve the efficiency of updates and allow exploration strategies, my work has revealed that their benefits go way beyond their original scope. In that perspective, I have extended them to beam parsers, as well as proposing a generalized version of their definition. As this endeavour made me step back and carefully compare the existing algorithms for structured training, I have identified that they all correspond in fact to various instances of the same framework, whose unification was not conceivable in previous literature. This finding led me to question several aspects of state-of-the-art parsing, and notably the train-test consistency of search space sampling: I have indeed uncovered, and addressed, a significant bias of global training toward the beginning of derivations (Appendix A).

An obvious continuation of this work is to make full use of PanParser's modularity, by adding more and more modules for other classifiers (especially LSTMs, whose Stack-LSTM variant (Dyer et al., 2015) has proved quite successful for transition-based parsing) and transition systems: SwapStandard (Nivre, 2009), but also the recently released ArcSwift system (Qi and Manning, 2017), as well as list-based systems like EasyFirst (Goldberg and Elhadad, 2010) or the Covington algorithm (Covington, 2001). It would also be interesting to include other tasks covered by transition-based algorithms, like constituency parsing (Coavoux and Crabbé, 2016) or semantic parsing (Wang et al., 2015). More generally, my work advocates for more systematism in the study of training strategies, and transition system properties: I have uncovered that the proof of ArcEager's arc-decomposability (Goldberg and Nivre, 2013) does not always hold, depending on the ROOT position – I propose to pursue on this path, by tackling the derivation of dynamic oracles in a more systematic manner for the existing systems, and notably their usual variants, like those imposing root unicity. Besides, I have proposed an approximate way to exploit non-projective examples, using dynamic oracles, but they would surely benefit from a sound oracle derivation for that case. A last possible track is to further relax the types of objects (either input, output, constraints or training annotations) manipulated by PanParser, so that syntactic information of any sparsity and granularity can be exploited. As a matter of fact, I have already taken a few steps in that direction, by applying dependency *direction* constraints, based on an inventory of PoS with nearly deterministic attachment direction, and this technique yielded promising results when used as first stage in a cascade. Another non-standard input would be partial *sentences*, as done by Köhn and Menzel (2014).

Regarding the capacity of parsers to generalize, I have shown ways to control it to some extent, by scheduling techniques and filtering of lexicalized parameters, both in monolingual and cross-lingual contexts (Chapter 6), with additional experiments in PoS tagging (Appendix C). I believe that such generalization techniques can also be useful for domain adaptation, and to speed up the training process in general. Bartenlian et al. (2016) have attempted similar experiments for domain adaptation, but they only report negative results, which remain largely unexplained. I would thus start by tackling the main issue that impedes contributions on that matter: the lack of a dedicated *intrinsic* metric to quantify the system's ability to generalize upon linguistic content. My work on classifier-oriented measures of data complexity (Chapter 6) can well be the starting point of that study, since complex classes are typically those with poor generalization.

On a final note, the interpretation of the behaviour of NLP systems at the classifier level has seen a growing interest in the last few years, urged by the higher opacity of neural systems (Li et al., 2016). While such investigations are necessarily easier in a feature-based case, I have still experimented with a large range of analysis and visualisation techniques in Chapter 6, which could be further improved for better model interpretation.

Fulfillment of objectives There were three aspects in my initial research question: making all resources usable, enabling full exploitation of any arbitrary set of resources, and taking multi-source combination to a finer-grained level. I assess in the following the success of my work along each objective.

New usages have been proposed for resources that are typically overlooked (typological information and tiny target data). As a matter of fact, I have used throughout my work – and my submission to the *CoNLL 2017 UD Shared Task* uses – a large variety of cross-lingual resources: *crowd-sourced* data (Section 5.1, for tagger transfer), *typology-based similarity scores* (Chapter 9, for source selection and ordering), the actual *typological values* (with the rewriting rules of Chapter 8), *parallel* data (Section 5.1 as a case study, Section 7.4 for a new projection technique), *source delexicalized* data (whose usability is maximized by Chapter 8), *target annotated samples* (whose complementarity is exhibited and characterized in Sections 5.2, 6.4 and Chapter 9) and large *target raw data* (used in Chapter 8 to increase source-target similarity). I have also relaxed some classic requirements on such data, allowing non-projective trees and partial annotations. For specifically *small parallel data*, Appendix B ensures that they remain usable in transfer contexts. *Lexical similarity*, which I have found especially overlooked in the literature, has not been thoroughly investigated, but Appendix C provides interesting perspectives on its use in tagger transfer.

I see however a few remaining gaps in my use of diverse resources. Appendix B exploits *bilingual lexicons* to transfer a lexicalized word aligner, and they could also be used to ensure translational consistency of alignment links; but I propose no way to leverage them regardless of the availability of parallel data, apart from their existing uses for resource glossing (Zeman and Resnik, 2008) – which underperforms delexicalization – and parameter sharing (Duong et al., 2015b). Instead of just betting on enough similarities to yield good accuracies with direct transfer, I could also have used *relatedness* in a more explicit way, notably by exploiting the phylogenetic *links* themselves (as proposed by Berg-Kirkpatrick and Klein (2010)), and not just by computing language distances. Finally, future work should investigate ways to exploit quantitatively the qualitative priors on the target language (like knowing that it has a free order, rich morphology, large vocabulary), or sparse word-level knowledge (when a word is recognized as denoting a month for instance, but without knowing which month, so that building a lexicon is not possible).

Regarding the combination of arbitrary sets of resources in real-world contexts, I have ensured that all of them are optional: there is no predefined set of resources that must be made available before entertaining transfer. For all of my contributions that were built on a specific resource, I have indeed provided a resource-lighter alternative, or a fallback on another kind of resource. This is notably the case in Chapter 8, where typology-based rewriting rules can be substituted, for the undocumented languages, with a method based on raw data. Similarly, seeking to prevent the availability of a target treebank sample to become the bottleneck of my method, I have also devised strategies to overcome its lack in Section 9.2. Although such fallback options are presumably not as accurate as the original proposals, they preserve the applicability of the method, whatever the available resources are.[1]

As for multi-source combination, including the target itself, I have achieved it at a fine-grained level with the notion and measure of cross-lingual relevance. However, I believe that there remains much room for improvement in that regard, as Chapter 9 proposes only a handful of ways to estimate relevance, compared to the whole range of possible strategies. To move towards processing even more languages, future work should focus on investigating finer ways to ascertain cross-lingual relevance, notably when target data lacks. This approach is, however, a slippery slope: devising a strategy that works even when completely devoid of data or knowledge would imply the ability to readily process accurately any new unknown language, a belief which I do not share.[2] I notably distance myself from the growing community interest for bilingual NLP based on monolingual data only, without any cross-lingual input. I rather believe that the biggest achievements reside in a better collaboration with the linguistics field, for establishing a more systematic typology of the linguistic phenomena that are subject to variations in cross-lingual relevance

[1]Questions remain, however, on whether a target PoS tagger, whose availability is assumed in general, notably for delexicalized transfer, can be considered a resource. If so, could annotation projection be achieved without PoS tags, for instance using projected embeddings, or lexical features only? Another possibility is to transfer them, but Zeman et al. (2016) report negative result on the combination of delexicalized tagging and parsing; still, coupling partial tree projection with partial PoS projection (Wisniewski et al., 2014) could be more successful.

[2]The only grounding of such ability would be language universals, that is, expectations based on the fact that all those languages are spoken by human beings (with similar biological constraints), living in the same world (with similar pragmatic knowledge). But as pointed out in Chapter 2, a debate has been opened on that matter, and Evans and Levinson (2009) have expressed some reservations regarding the truly universal nature of such properties. Instead of universal, they rather appear to be universal *among the languages we are most familiar with*, in other words in cases where tidbits of knowledge are always available, in any form.

(the order of subjects, verbs and objects has for instance been established as a major difference between languages, and as impacting a lot their whole syntax), so that the amount of target knowledge needed to estimate relevance can be minimized.

Overall, all my initial concerns have been addressed, although my proposals can still be further improved in several ways.

This thesis has extended in many regards the perspectives of cross-lingual transfer, but also monolingual parsing. I have named a few other use cases, but the scope of applications of my cascading framework is broad, and I believe that this selective combination approach is a promising path towards full and accurate exploitation of resources of any kind, size and quality. That way, we can hope building a future where NLP is not a privilege, but a given for any language in the world.

Appendices

Abstract

The appendices accompanying this thesis describe three side projects conducted in the frame of that work, but which do not directly contribute to its main topic, cross-lingual parsing. The first appendix further deepens the formalism and empirical benefits of using dynamic oracles with beam parsers. Then a study on word alignment transfer exemplifies how the proposed typology of transfer approaches can be applied to other tasks, including the poorly addressed transfer of bilingual knowledge. Finally, lexical similarities are shown to be valuable for cross-lingual transfer, with a series of experiments on PoS tagger transfer.

Appendix A

Advanced strategies for global training in parsing

***

Contents

This appendix describes more formally the restart strategy proposed in Section 7.2, in contexts of early update and max-violation. The main benefit of this option is to allow rethinking the global training framework on the same ground as local training. But it is shown here that using a restart strategy has also several empirical benefits, on accuracy, sampling quality during training, and convergence time. Notably, a training bias toward the beginning of the derivation – directly related to the choice of updating on partial derivations – is unveiled, and addressed by that method.

Section A.1 first clarifies the restart strategy with a formal description. Three training biases affecting beam parsers are then presented in Section A.2, as well as how they are addressed by the restart strategy (together with the use of global dynamic oracles). Empirical evaluation (Section A.3) reports small but consistent accuracy gains with that method, but further measures reveal that it is especially beneficial to the end of the sentence, and relate those findings with distribution similarity measures. The material presented here is drawn from two publications (Aufrant et al., 2016c, 2017).

A.1 Restart strategy with global dynamic oracles

Algorithm 1 describes the main procedure for global training, in the traditional fashion and when restart is used. In the baseline version, for each sentence x (with gold parse y), the parser is initialized in configuration INITIAL(x), then the search space is explored to find a pair of update configurations c^+ and c^- (ORACLE procedure), after which a single UPDATE operation is performed. With restart, the only difference is that the ORACLE and UPDATE steps are repeated as many times as needed to completely parse the sentence (indicated by FINAL); each time, the parser is reinitialized in the predicted configuration c^-, which allows some exploration during training.

Algorithm 1: Global training on one sentence, with and without restart.

```
$\theta$: model parameters, initialized to $\theta_0$ before training
FINAL$(\cdot)$: true iff the whole sentence is processed
Function STRUCTUREDTRAINING$(x, y)$
    $c \leftarrow$ INITIAL$(x)$
    $c^+, c^- \leftarrow$ ORACLE$(c, y, \theta)$
    $\theta \leftarrow$ UPDATE$(\theta, c^+, c^-)$
Function STRUCTUREDTRAININGRESTART$(x, y)$
    $c \leftarrow$ INITIAL$(x)$
    while $\neg$FINAL$(c)$ do
        $c^+, c^- \leftarrow$ ORACLE$(c, y, \theta)$
        $\theta \leftarrow$ UPDATE$(\theta, c^+, c^-)$
        $c \leftarrow c^-$
```

The scope of this high-level description is general, and not limited to a particular classifier: UPDATE denotes any way that the underlying classifier has to update its parameters, based on a pair of positive and negative configurations. In PanParser, UPDATE applies the perceptron update rule on global vectors, while in a globally normalized neural parser (Andor et al., 2016) it would correspond to gradient computation.

It also applies to various training strategies, corresponding to various implementations of the ORACLE function. Algorithm 2 describes how the early update and max-violation strategies fit in that framework, here using a global dynamic oracle. In both cases, the parser first searches for a violation (FINDVIOLATION procedure), then based on the erroneous beam state that was identified, some additional operations are performed to actually build the update pair: EARLYUPDATEORACLE just retains the best correct derivation in beam, as well as the top-scored hypothesis, while MAXVIOLATIONORACLE pursues parsing to select the pair with maximal score difference.

A.2 Using global dynamic oracles to correct training biases

As explained in Section 3.6, with static oracles, beam parsers remain prone to the same two effects that motivated the use of dynamic oracles for greedy parsers. On

Algorithm 2: Global dynamic oracle: error criterion and choice of an update configuration pair.

```
$c_0$: configuration to start decoding from
$top_\theta(\cdot)$: best scoring element according to $\theta$
$\textsc{Next}(c)$: the set of all successors of $c$ (or only $c$ if it is final)
Function FindViolation($c_0$, $y$, $\theta$)
    Beam $\leftarrow \{c_0\}$
    while $\exists c \in Beam, \neg\textsc{Final}(c)$ do
        Succ $\leftarrow \cup_{c \in \text{Beam}} \textsc{Next}(c)$
        Beam $\leftarrow$ $k$-best(Succ, $\theta$)
        if $\forall c \in Beam, \neg\textsc{Correct}_y(c|c_0)$ then
            gold $\leftarrow \{c \in \text{Succ} | \textsc{Correct}_y(c|c_0)\}$
            return gold, Beam
    gold $\leftarrow \{c \in \text{Beam} | \textsc{Correct}_y(c|c_0)\}$
    return gold, Beam
Function EarlyUpdateOracle($c_0$, $y$, $\theta$)
    gold, Beam $\leftarrow \textsc{FindViolation}(c_0, y, \theta)$
    return $top_\theta$(gold), $top_\theta$(Beam)
Function MaxViolationOracle($c_0$, $y$, $\theta$)
    gold, Beam $\leftarrow \textsc{FindViolation}(c_0, y, \theta)$
    candidates $\leftarrow \{(top_\theta(\text{gold}), top_\theta(\text{Beam}))\}$
    while $\exists c \in Beam, \neg\textsc{Final}(c)$ do
        Succ $\leftarrow \cup_{c \in \text{Beam}} \textsc{Next}(c)$
        Beam $\leftarrow$ $k$-best(Succ, $\theta$)
        Succ$^+ \leftarrow \cup_{c \in \text{gold}} \{c' \in \textsc{Next}(c) | \textsc{Correct}_y(c'|c_0)\}$
        gold $\leftarrow$ $k$-best(Succ$^+$, $\theta$)
        candidates $\leftarrow$ candidates $+ (top_\theta(\text{gold}), top_\theta(\text{Beam}))$
    return $\text{argmax}_{c^+, c^- \in \text{candidates}}(score_\theta\,(c^-) - score_\theta\,(c^+))$
```

the one hand, because only one derivation is designated as reference, in case of spurious ambiguity the parser can consider correct derivations as incorrect, and thus attempt noxious updates;[1] but this issue is solved as soon as a global dynamic oracle is solved.

On the other hand, the parser is only ever trained in gold parts of the search space and does not know how to take good decisions elsewhere, thus starting to make worse and worse predictions as soon as a first error has occurred. While for greedy parsers the unknown territory starts whenever an erroneous action is applied, beam parsers are more familiar with incorrect actions, as they explore many of them. But they still have an unknown territory: states in which the beam contains only erroneous configurations. Although they have only been trained to keep gold hypotheses inside the beam (and in the end, rank them first), in such cases they

[1]According to my measures, a beam parser trained with early update and a static oracle counter-intuitively predicts correctly *fewer* heads of the current sentence just after an update than just before, for 15% of the updates (French SPMRL, during 10th epoch).

cannot achieve that, and are supposed instead to put the hypothesis with best UAS on top of the beam, which they never learned about. The issue here is that in the standard global training procedure, even with a global dynamic oracle, this lack is not addressed: as the beam is always initialized in an empty configuration (not embedding any previous error), the first error is to leave the gold space, and only that is corrected. In that sense, even if global dynamic oracles are crucial to achieve it, training in the suboptimal space is enabled specifically by the restart strategy: it implies initializing the beam in the middle of a derivation, including past errors, and as such the parser can learn to reward hypotheses with good (albeit not perfect) UAS.

Additionally, I have unveiled a third issue with traditional training strategies, which also introduces a discrepancy between training and testing conditions. This sampling bias is in fact specific to global training, as it results from the choice of updating on partial derivations (early update and max-violation strategies), instead of performing full updates. For instance on the French SPMRL, when training with an *early update* strategy, the end of the derivation is reached for only 41% of the examples at the 10th epoch[2] and, on average, only 57% of a derivation is considered; the *max-violation* strategy, which computes longer partial derivations, partly alleviates this effect: these proportions raise, respectively, to 53% and 81%. But while the choice of partial updates has been experimentally proved (Huang et al., 2012) to be critical in achieving good performance, it prevents parsers from visiting configurations corresponding to derivation endings. This explains why configurations and transitions involving final punctuation marks, verbs in SOV languages like Japanese or German subordinate clauses, the ROOT token when placed at the end (Ballesteros and Nivre, 2013), but also stack features involving long-distance siblings, are too rarely seen in training, thereby hurting predictions in such configurations. For this issue, the restart option is a straightforward solution, since it ensures that configurations that are close to derivations endings will be seen more often during training: for a given example, each part of the derivation is involved exactly once in an update, which restores the balance between beginnings and endings.[3]

The standard procedure for training beam parsers is consequently biased in three ways: rejection of correct hypotheses, reinforcement of gold derivations only, under-representation of derivation endings; but all those issues are solved at the same time, when using global dynamic oracles with the restart strategy. I thus propose to experiment with this advanced training strategy.

[2]On the French SPMRL treebank, at the 10th epoch, the parser is close to convergence (see Section A.3).

[3]Standard training with full update also ensures this, but with the risk of divergence (Huang et al., 2012). Restarting in c^- with a new beam has the same convergence guarantee as standard *early update* and *max-violation*.

A.3 Experiments

The validity of the approach is evaluated on the SPMRL treebank (Seddah et al., 2013).[4] The considered baselines are a greedy parser trained with a dynamic oracle (GREEDY DYN) and beam parsers trained with the *early update* and *max-violation* strategies and a static oracle (resp. EARLY and MAXV). The IMP-EARLY and IMP-MAXV systems correspond to the improved versions of both strategies, when using global dynamic oracles and restart. All experiments use the ArcEager version of PanParser, with ROOT placed at the end (which are the default parameters, and the best performing ones).

Main results Table A.1 reports the UAS of all training strategies, where each score has been averaged over 5 runs.[5] Results show that my learning strategy consistently outperforms the corresponding baseline, with average increases of 0.2 UAS, up to 0.7 UAS.

	ar	de	eu	fr	he	hu	ko	pl	sv	μ
GREEDY DYN	83.98	90.73	84.00	84.23	83.78	84.33	82.79	87.66	86.35	85.32
EARLY	85.03	92.74	84.42	86.02	85.39	85.63	82.73	89.60	87.00	86.51
IMP-EARLY	**85.27**	92.89	84.59	86.26	**85.84**	**85.74**	**82.98**	89.55	**87.37**	86.72
MAXV	85.06	92.77	84.59	86.10	85.53	85.57	82.68	89.42	87.16	86.54
IMP-MAXV	85.04	**92.90**	**84.68**	**86.26**	85.83	85.55	82.94	**90.12**	87.31	**86.74**

TABLE A.1: UAS of the baseline and improved strategies for global training, on the SPMRL datasets.

Further analysis In order to assess that these improvements target indeed the aforementioned training biases, I report additional measures on these aspects. They are detailed here only for the French treebank, but similar results have been obtained in other languages.

Table A.2 shows the performance imbalance between various positions in the sentence (which roughly correspond to various positions in the derivation) and confirms that my improvements partly alleviate this phenomenon: the scores on the first half of the sentence are mostly unchanged, while large gains are reported on the second half.

[4]The particularity of that dataset is that it consists in morphologically-rich languages only. However, I have not used that property here, and notably resorted to standard feature templates, with coarse PoS tags.

[5]As a matter of fact, these experiments do not use the standard SPMRL tests sets, but only their projective trees. Indeed, non-projective dependencies are typically long-distance, so that tokens in the beginning of the sentence can be attached very late in the derivation, which would have made the fine-grained analysis much less legible. This choice does not prevent fair comparison between the baseline and improved strategies, but note that the baseline results reported here are consequently not comparable with other published results.

Quarter	1st	2nd	3rd	4th
EARLY	90.0	85.4	83.1	84.7
IMP-EARLY	90.0	85.3	84.2	85.1

TABLE A.2: UAS of the standard and improved early update strategies, depending on the position in the sentence (French SPMRL dataset). The first quarter corresponds to the attachment of tokens in the first 25% of the sentence length.

	Baseline	Improved
EARLY	0.350	0.280
MAXV	0.357	0.277

TABLE A.3: Effect of the proposed improvements on the Kullback-Leibler divergence between the train and test feature distributions (French SPMRL dataset).

To assess that these UAS gains result from a better matching of training and test configurations, I compute the Kullback-Leibler divergence between the probability distribution (estimated with frequency counts and 0.1 Laplace smoothing) of the features of all configurations in beam scored during the 10th training epoch and the feature distribution seen at test time.

Table A.3 reports the Kullback-Leibler divergences induced by my refinements with respect to the corresponding baselines. It clearly shows that my 'improved' learning strategy considers training examples that are closer to test configurations. Similar experiments on greedy parsers show that their train-test divergence is reduced from 0.320 to 0.219 by the dynamic oracle and exploration strategy of Goldberg and Nivre (2012). In these two experiments, feature similarity correlates with UAS improvements and can therefore provide a new way to interpret oracle influence.

Finally, regarding efficiency, I observe (Figure A.1) that IMP-EARLY converges in a number of epochs similar to that of standard MAXV. Despite an increased number of updates, it is however slightly faster (in CPU time) because it avoids the extra reference pre-computation.

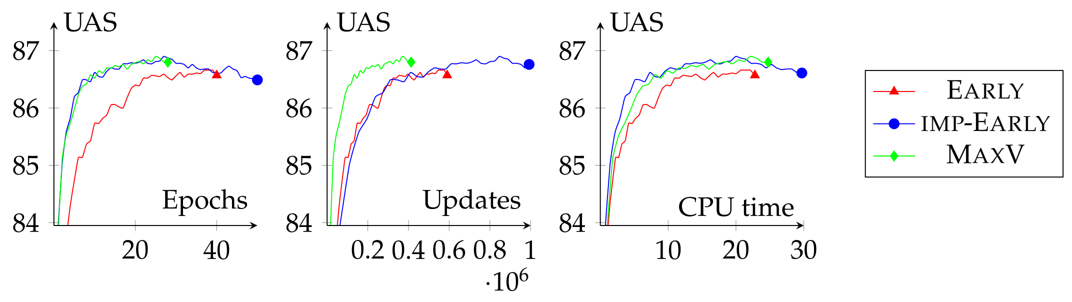


FIGURE A.1: Learning curves on the validation set (French SPMRL dataset). IMP-EARLY has the same update efficiency as EARLY, but with the epoch and computation time convergence of MAXV. All curves are drawn up to convergence, except for the IMP-EARLY strategy, for which the full learning curves are shown.

It thus appears that the proposed training strategy is purely beneficial, as it improves sampling at training time, and consequently the accuracy on the end of the sentence, but without any computation time overhead.

A.4 Conclusion

In this work, I have presented in more details the restart strategy for global training of parsers, and shown that its benefits go beyond formalism: compared to the standard early-update and max-violation strategies, it reduces the discrepancy between the feature distributions seen at training and testing times, which has been empirically assessed. As a result, parsing accuracy is improved, especially on the second half of the sentence. As an additional benefit, this training strategy also has a better convergence time than both state-of-the-art methods.

Restarting in the update configuration is one way to address the various sampling biases, but there may be other successful variations of the method. Building on the new range of possibilities offered by global dynamic oracles, an interesting track for future work is thus to devise new strategies to sample the configurations in which to initialize the beam: these oracles allow indeed the parser to train in any part of the search space, and even randomly generated configurations would be suitable.

Appendix B

Word alignment transfer

***

Contents

This appendix presents a series of experiments on cross-lingual transfer of word alignment models, which have been published in (Aufrant et al., 2016a).

The purpose of this work is twofold. First, it addresses a major bottleneck of transfer methods based on annotation projection, whereby the size of parallel data impacts transfer efficiency twice: small data provides less training material, but has also lower alignment quality. This penalty is a direct consequence of using state-of-the-art alignment methods, which are unsupervised and whose accuracy thus depends a lot on data size. The paradox is then that reliable word alignments – on which annotation projection relies a lot – can only be computed for large-scale parallel corpora, a situation that is unlikely to happen for actual low-resourced languages, although the annotation projection method has been designed for precisely such languages.

Second, this work extends the perspectives of the cross-lingual approach, by showing that the main paradigms for transferring monolingual models can also apply to bilingual tasks, in this case word alignment. Hitherto, word alignment transfer has hardly been addressed in the literature. Wang et al. (2006) and Levinboim and Chiang (2015) improve alignment of small parallel data thanks to large parallel corpora with another language (the pivot), by means of triangulation, i.e. marginalization over the pivot language. While close in spirit to the transfer approach, in practice those works rather focus on the *combination* of two well-resourced models to build a model in a low-resourced pair. In comparison, my aim is to either *build* parallel data in the low-resourced pair, or *modify* a single existing model, as is usually done in the cross-lingual transfer literature. To my knowledge, the only work that tackles low-resourced alignment with that approach is that of Nakov and Ng (2012), who improve alignment of e.g. English-Indonesian data by concatenating it with a much larger English-Malay corpus (Malay being closely related to Indonesian) before unsupervised alignment; they also explore how balancing both parts of the resulting corpus can further improve the alignment.

After a quick review of the standard algorithms for word alignment (Section B.1), I formalize in Section B.2 a series of scenarios in which word alignment transfer can be entertained. By instantiating the main transfer paradigms described in Chapter 2, Section B.3 proposes some basic methodologies for transferring bilingual knowledge *across pairs of languages*. Experiments reported in Section B.4 show that, at least for some of these scenarios, simple-minded methods can be surprisingly effective and open a discussion on further prospects and perspectives for future work.

In the transfer literature, the terms 'source' and 'target' denote the well-resourced and low-resourced languages; but the word alignment literature also uses them to identify both sides of the parallel data, as the alignment algorithms are often asymmetric. The following uses the version of the word alignment literature: source and target are not the well-resourced and low-resourced pairs, but the languages of the low-resourced pair. Besides, this vocabulary is consistent with the perspective of using the small parallel corpus for annotation projection: word alignments are produced between source and target languages, which are the source and target of monolingual transfer.

B.1 Aligning words

The most popular models for statistical word alignment are the IBM models 1 to 6 (Brown et al., 1993; Och and Ney, 2003) and the HMM model of Vogel et al. (1996). These probabilistic generative models decompose the probability of an aligned sentence pair as the conjunction of a word translation model, a distortion model (models 2 and up) and a fertility model (models 3 and up). Distortion is absolute for models 2-3 and relative for models 4-6; in the HMM model, it is captured by Markovian dependencies between consecutive alignments links. Among these parameters, the translation model is lexicalized in both languages, fertility is lexicalized on the source side and distortion is unlexicalized but relies on word clusters for models 4-6. In all cases, parameters are learned in an unsupervised way using the EM algorithm.

Many refinements to these algorithms have been proposed, often to improve computational performance (Dyer et al., 2013). Another line of work tries to improve the low generalization power of IBM and HMM models by using feature-based models (Moore, 2005; Berg-Kirkpatrick et al., 2010). However, the IBM models remain today the most widely used approach both because of their efficiency and because they do not require any annotated data. They will thus serve as the main baseline.

B.2 Word alignments: cross-lingual scenarios

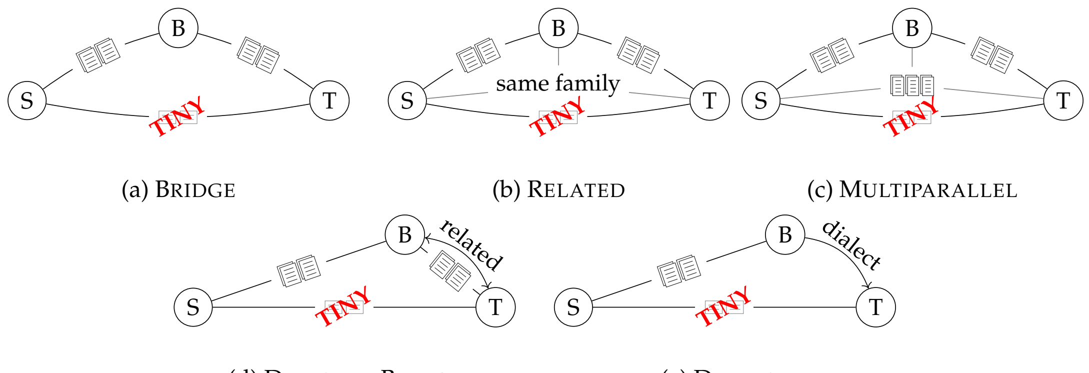
Scenarios for improving word alignment with cross-lingual transfer fall into two categories, depending on whether the source and target languages play a symmetric role.

I first consider the standard symmetric BRIDGE scenario (illustrated in the first line of Figure B.1): it involves two languages S and T, for which large bitexts with a 'bridge' language B exist, readily yielding reasonably-good alignment models for S-B and B-T. I am however specifically interested in the S-T pair, for which I only possess a small parallel corpus. This can happen either in the context of a bilingual task like translation or because word alignments are needed for cross-lingual transfer of a monolingual model. In this case, the purpose is to annotate the S-T data thanks to information contained in the high-quality S-B and B-T models. Two variants provide interesting refinement opportunities: MULTIPARALLEL, in which part or all the S-T parallel data is also aligned with sentences in B,[1] and RELATED, when all three languages belong to the same linguistic family. Taking advantage of such similarity however requires more expressive models than the IBM series, which only operate at the level of wordforms and are therefore agnostic to lexical or PoS-based similarities.

The RELATED scenario can be illustrated on the example of morphosyntactic model transfer from Italian to Romanian, using annotation projection. For lack of a large Italian-Romanian corpus to compute robust word alignments, an option is to use French as a bridge, collect large bitexts for Italian-French and for French-Romanian, and improve the quality of the Italian-Romanian word alignment model thanks to the Italian-French and French-Romanian models. Even though the resulting Italian-Romanian bitext is noisier and smaller than the French-Romanian one, which could also be used for transferring PoS labels, transfer to Romanian may still be more accurate when using Italian as an additional source, or even as a better source than French.

In the second type of scenarios (second line of Figure B.1), source and target languages play asymmetric roles regarding transfer. Bitexts for S-T (e.g. English and Ukrainian) come with small parallel data, but there exists a language $\tilde{T}$ related to T (and unrelated to S) for which large parallel data with S are available (e.g. Russian). I consequently consider transfer of S-$\tilde{T}$ word alignments to the S-T pair. This

[1] Such multiparallel corpora are typically drawn from minutes or official documents published by international organizations, like Europarl (Koehn, 2005) or MultiUN (Eisele and Chen, 2010).



(a) BRIDGE (b) RELATED (c) MULTIPARALLEL

(d) DIRECTED BRIDGE (e) DIALECT

FIGURE B.1: Typology of the main use cases for cross-lingual transfer of word alignments, depending on the available resources and on linguistic similarities. Alignment between languages S and T is improved thanks to language B.

can be interpreted as standard cross-lingual transfer from $\tilde{T}$ to T, but with the difference that the transferred knowledge is not monolingual but bilingual because of the interactions with S. With large data in both S-$\tilde{T}$ and $\tilde{T}$-T, I call this scenario DIRECTED BRIDGE. This is actually the context of Wang et al. (2006)'s works on English-Japanese, using Chinese as a bridge language, but their cross-language word similarity does not exploit Chinese-Japanese lexical similarity.

Finally, in the DIALECT scenario, T is a dialect of $\tilde{T}$, and even though parallel $\tilde{T}$-T data is not necessarily available, the transfer process can rely on the large number of common wordforms. This would, for instance, be the case with the alignment of English with MS Arabic and dialects. Thanks to the large linguistic overlap, and contrarily to the previous scenarios, here again methods from the domain adaptation literature (Wu et al., 2005) may also successfully apply.

Before closing this section, I would finally like to stress the fact that the motivations for transferring alignments can be many: one might want to get alignments for a small parallel bitext, to then transfer other annotations, or one might want to bootstrap an alignment model with transferred parameters, or even to train a small SMT, etc. Each such motivation may call for different strategies.

B.3 Concrete methods for transferring alignments

This section exemplifies, with simple systems, how general transfer methods can be instantiated for alignment transfer.

From now on, I focus on a DIRECTED BRIDGE scenario, further assuming that the task is to annotate a very small parallel corpus. This situation will be simulated, in Section B.4, for the English-Swedish pair, using Danish as a bridge language, as it is closely related to Swedish. Both English-Danish and Danish-Swedish pairs will be considered well-resourced. Six methods to address that scenario are proposed in the following; for clarity, they are all summarized in Table B.1 and illustrated in Figure B.2.

DATA SPACE	
CAT-B	concatenate S-B and test data; train
TR-B	word-for-word translate S-B data; concatenate with test data; train
B-TR	word-for-word translate test data in B; concatenate with S-B data; train
PARAMETER SPACE	
B	train an S-B model; apply on test data
GLOSS-B	train an S-B model; apply on test data word-for-word translated in B
PARAM-B	train an S-B model; translate the parameters; apply on test data

TABLE B.1: Summary of proposed methods for alignment transfer, for test data S-T and bridge language B.

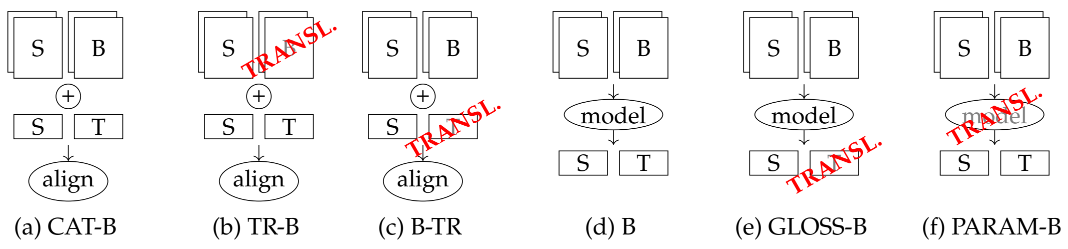


FIGURE B.2: Illustration of proposed methods for alignment transfer, for test data S-T and bridge language B.

I start with a straightforward baseline (CAT-DA), consisting in concatenating the English-Danish and the English-Swedish data before training (as done by Nakov and Ng (2012) for much closer languages). The underlying assumption is that both Danish and Swedish are subsumed by a Scandinavian meta-language. Despite their similarity, these languages only have few common wordforms (Zeman and Resnik, 2008) and their vocabularies overlap mostly on function words, numbers and proper nouns. Consequently, the resulting translation model will in fact consist in two quite separate submodels that hardly interact. A unique model is however trained for the unlexicalized parameters like distortion.

This approach corresponds to joint learning with parameter sharing. It can however be interpreted from another point of view: as the purpose of transfer at the data level is to produce noisy artificial training data, here I produce approximate English-Swedish sentence pairs, with the Danish set considered as a proxy to Swedish.

Pursuing along these lines, I propose a second method to produce fake Swedish data: first train an IBM 1 model on the Danish-Swedish corpus to extract a translation model, then replace every Danish token of the English-Danish data with its most likely translation (leaving OOV tokens unchanged). Then again the resulting bitext is concatenated with the English-Swedish corpus and standard training ensues. I notice that a symmetrical procedure can be straightforwardly implemented, by using a Swedish-Danish IBM 1 model to translate the test set in Danish: alignment is performed in the Danish domain, but since there is an exact Swedish-Danish token correspondence, the resulting word alignment can be directly used for the English-Swedish part. This is an example where English-Swedish IBM models will not be delivered, and can hardly be re-estimated from so few sentence pairs. I denote these methods TR-DA and DA-TR respectively.

The same intuitions can finally be applied for transferring in the parameter space: direct transfer is achieved by training an English-Danish model, then using it directly to annotate the English-Swedish pairs (E step of the EM algorithm). This approach (denoted DA) is rather naive and is expected to perform poorly, but it can be improved in a similar manner to the 'glosses' method of (Zeman and Resnik, 2008), by translating the Swedish tokens into Danish with a Swedish-Danish IBM 1 model (a method denoted GLOSS-DA). Using the converse translation model to translate the English-Danish lexicalized model parameters and thus produce a full English-Swedish model yields slightly different results (PARAM-DA).

Deriving other methods The purpose of those strategies is mostly to set baselines and conduct qualitative analyses: more complex alignment transfer methods can actually be designed, following the typology of Chapter 2. Two restrictions apply however.

First, annotation projection cannot be entertained in any scenario: while annotation projection for a monolingual task needs parallel data, for a bilingual model it would require multiparallel data. The converse is not true however, and the MULTIPARALLEL scenario can be successfully exploited without annotation projection (Kumar et al., 2007).

Second, the delexicalized approach for transfer causes a chicken-and-egg situation in real-life scenarios. Indeed, when the target language is low-resourced, one cannot assume the availability of a PoS tagger that is needed to compute delexicalized representations. Conversely, methods like Wisniewski et al. (2014)'s cross-lingual PoS tagger projection and Täckström et al. (2012)'s clusters are not applicable without a word-aligned corpus. Finding common, even coarse-grained, representations then becomes a huge obstacle in many scenarios where alignment transfer is needed, which makes this approach less relevant.

As a final note, I point out that the RELATED scenario can be simulated with two symmetric instances of DIRECTED BRIDGE interpolated in the data or parameter space. The straightforward strategies described here can therefore be extended to other scenarios.

B.4 Experiments

In this section, I experiment with the methods introduced above and compare them with standard unsupervised models of varying sizes.

Experimental setup I evaluate the proposed alignment transfer methods on the English-Swedish test set provided by Holmqvist and Ahrenberg (2011), which consists of 192 word-aligned sentence pairs extracted from the English-Swedish part of Europarl (Koehn, 2005). I score the methods according to the intrinsic Alignment Error Rate (AER) metric proposed by Och and Ney (2000).

As documented in a large body of literature (Lopez and Resnik, 2006; Fraser and Marcu, 2007; Lambert et al., 2009, 2010), AER poorly correlates with translation quality of the systems trained on the evaluated alignments, especially for large corpora, and extrinsic metrics like the BLEU score should be preferred, were MT training the final goal of alignment.

The concatenation methods proposed here are intended for very small data, with large imbalance between the target and the bridge sets, a data size for which the SMT application is not relevant. Consequently, for extrinsic evaluation, I rather use as metric the PoS accuracy of a cross-lingual tagger, weakly supervised by the resulting word alignments (Wisniewski et al., 2014). In such a system, each extra sentence pair brings valuable knowledge, while incorrect links strongly noise the system, making the accuracy a direct indicator of alignment quality. Besides, this step completes a realistic low-resource scenario where word alignments are needed for the sole purpose of cross-lingual transfer between both languages.

The English-Swedish bitext is used both as a test set[2] for intrinsic evaluation and as projection data to train cross-lingual taggers. PoS accuracies are computed on the coarse PoS tags of the Swedish test treebank of UD 1.2 and the source English tagger is trained on the training portion of the same corpus.

AER and PoS accuracy are measured for the three concatenation methods presented in Section B.3, with transfer through Danish: CAT-DA, TR-DA, DA-TR, and the three parameter transfer methods: DA, GLOSS-DA, PARAM-DA. To evaluate the benefits of using a related bridge language, I also run the concatenation experiments with transfer through Greek, which is only distantly related to Swedish and also uses a different alphabet (methods CAT-EL, TR-EL, EL-TR). Finally, the alignment performance is confronted with that of concatenation with English-Swedish data of various sizes, from no added pair (BASELINE) to full concatenation of the 1.8M sentence pairs in Europarl (CAT-SV).

[2]In every method where additional parallel data is required, I use Europarl without the Q4-2000 section, which is reserved for tests.

Results Table B.2 reports the AERs of the various methods when using IBM 1, HMM or IBM 4 models. The methods involving test set translation (da-TR and GLOSS-da) consistently yield the best cross-lingual accuracies for each model, with a relative error reduction of 45%, 52% and 59% respectively for da-TR and scores comparable to the full English-Swedish ones. Figure B.3 reports those AERs along the learning curves of English-Swedish models for increasing data sizes. It assesses that the da-TR method yields alignment quality comparable to unsupervised learning on 0.1M to 0.5M sentence pairs.

Table B.2 also reports PoS accuracy measures, which show a clear correlation of the most effective (in AER) transfer methods with high PoS accuracies. However, this measure does not allow to clearly rank the top few models (CAT-sv, TR-da, da-TR and GLOSS-da). Notably, here CAT-da and da-TR respectively outperform baseline and CAT-sv. It may be that word alignments obtained by transfer are less accurate but focus more on general cross-lingual structures which, in turn, enables a better annotation projection, or that these score differences are simply not significant. Coming up with a reliable interpretation of this issue however will require further experiments that are beyond the scope of this work.

		Swedish only		Danish data			Greek data			Danish parameters		
		baseline	CAT-sv	CAT-da	TR-da	da-TR	CAT-el	TR-el	el-TR	da	GLOSS-da	PARAM-da
AER	IBM 1	53.9	**26.5**	57.0	31.1	**29.6**	74.3	**35.9**	37.4	66.0	**28.3**	33.3
	HMM	35.3	**15.3**	41.9	20.5	**16.8**	58.3	26.9	**26.4**	46.7	**16.4**	25.8
	IBM 4	33.9	**12.3**	35.8	16.4	**14.0**	50.0	**20.6**	21.7	49.1	**14.8**	24.3
POS	IBM 1	68.7	**73.3**	58.7	73.8	**74.0**	47.4	**71.9**	71.5	67.0	**72.2**	71.1
	HMM	69.9	**73.8**	71.9	73.5	**73.6**	66.6	**73.4**	71.9	69.5	**73.4**	72.4
	IBM 4	73.0	**74.7**	74.0	73.9	**74.9**	72.0	73.4	**73.5**	66.7	**73.6**	72.0

TABLE B.2: AER and extrinsic cross-lingual PoS accuracies achieved by the proposed cross-lingual methods with Danish and Greek as bridge languages, compared to the baseline (unsupervised alignment on test data only) and the CAT-sv higher bound (addition of large English-Swedish data).

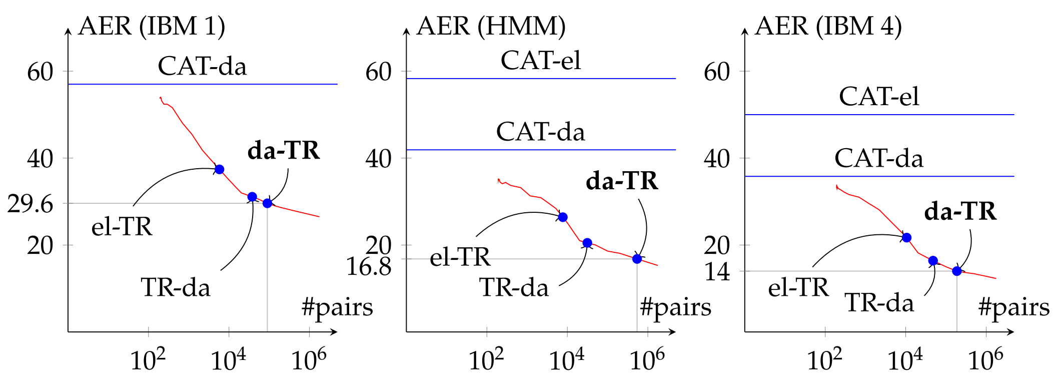


FIGURE B.3: AER on the English-Swedish test set for increasing data sizes (red curve) and some cross-lingual methods (reported in blue along the curve). The number of English-Swedish sentence pairs includes the 192 test pairs.

Discussion Unsurprisingly, direct data transfer methods CAT-DA and CAT-EL perform at best on par with the baseline, and often worse. Indeed, because of mostly disjoint vocabularies, the translation model is globally not improved by the external knowledge, while training the Swedish parameters on few pairs compared to the whole data produces weak parameters that are easily subjects to any noise coming from the few common wordforms (see IBM 1 results). This approach could however perform better in a DIALECT scenario, assuming that the $\tilde{T}$ and T languages have large enough common vocabularies to structure the alignment sets, and that shared wordforms are not accidental but actually correspond to common words that can be reliably exploited to transfer knowledge. The absolute AER gap reduces when adding shared distortion and fertility models, but this does not allow to conclude on a positive or negative effect of unlexicalized parameter sharing. This suggests however the need for experimenting with more selective parameter sharing.

Compared to the Danish ones, I notice that the Greek systems yield smaller but still significant improvements over the baseline, even if (a) Greek and Swedish are only distantly related and have therefore less common structures and (b) the impact of untranslated OOV words is higher in Greek because of the change of alphabet: indeed, it is quite unlikely to find identical wordforms in Swedish and Greek. The Greek systems have also been trained on slightly smaller bitexts (1.2M pairs, compared to 1.9M for English-Danish), but this ratio corresponds to a loss of at most 1 AER point in standard learning, which remains negligible in comparison to the cost of choosing Greek over Danish. All in all, using Greek as a bridge language is comparable to training with 10,000 English-Swedish pairs, instead of 400,000 when using Danish. This suggests that my methods can be useful even for unrelated languages in the BRIDGE scenario, even though the return ratio is much lesser than when languages are related: for Greek, 1.2M parallel sentences used cross-linguistically yield the same accuracy as 10,000 parallel sentences of the targeted pair (a ratio of approximately 1%); for Danish the ratio is closer to 20% (1.9M sentences providing the same information as 400,000 English-Swedish sentence pairs).

Finally, the comparison of the TR-DA and DA-TR columns reveals that English-Swedish alignment biased by the English-Danish alignment is more accurate when performed in the Danish domain than in the Swedish one. Intuitively, the *noisy application* of a *valid model* performs better than a *valid application* of a *noisy model*. Columns PARAM-DA and GLOSS-DA also support that interpretation. Besides, the fact that among the evaluated strategies, the most refined data transfer models outperform the parameter ones shows that even from a very small piece of target data, it remains possible to extract valuable knowledge to guide model adaptation to a new language.

B.5 Conclusion

This work has shown how the main methods for cross-lingual transfer can be applied on a poorly addressed task, the transfer of bilingual knowledge. I have presented a few realistic scenarios where transfer of word alignment is needed and then focused on one of them in the frame of unsupervised word alignment, thereby proposing six cross-lingual methods that are easy to set up.

Experiments on an English-Swedish test set reveal that even straightforward methods can extract valuable information for cross-lingual supervision: from five sentence pairs in the bridge language, they are able to extract knowledge equivalent to one English-Swedish pair. Altogether I achieve up to 59% relative error reduction. Further analyses also provide precious hints for accurate designs of alignment transfer methods.

Regarding word alignment transfer, my main interest for future work is to further explore the benefits of language similarity in the RELATED, DIRECTED BRIDGE and DIALECT scenarios, along two tracks: weighting based on linguistic similarity during the EM training and selective transfer at the submodel level.

Appendix C

Leveraging lexical similarities: zero-resource tagger transfer

***

Contents

The work presented in this appendix questions the marked under-use of lexical similarities in the cross-lingual transfer literature;[1] it is in indeed somehow paradoxical to exploit a closely related language by either *delexicalizing* its data (thereby hiding any lexical similarity) or resorting to parallel data (which is completely agnostic to wordforms). Intuitively, such similarities can be especially useful for transferring to dialects, but they also concern many pairs of related languages, including cognates (the Spanish 'relación' translates as 'relation' in French) and shared morphemes (in Spanish, infinitive verbs end with '-ar', '-er' or '-ir', while in French their endings are '-er' and '-ir', more rarely '-re').

Therefore, I explore methods to leverage directly those similarities, in a context of PoS tagging, and it is shown in the following that substantial improvements can in fact be achieved with very simple means. This proof of concept raises my hopes to manage, with some further work, incorporating that information into the transfer of more complex models, like parsers.

[1] PoS tagging is actually one task where a few prior works along that line exist, in contrary to parsing for instance. However, such works are in general based on a finite list of heuristically identified cognate pairs, typically to map the corresponding model parameters (Hana et al., 2006; Feldman, 2006) or to annotate raw target data (Scherrer and Sagot, 2014). But unlike those works, the objective pursued here is to alter the training procedure to build a tagger that handles *by design* all lexically similar words (even the OOV ones), by transferring all source knowledge, including morphological knowledge, which is not handled by the cognate-based approaches. Others have considered rule-based transliterations of source data, like Currey et al. (2016) for language modeling, but this approach requires human expertise, which is supposed unavailable here. Nakov and Ng (2012) propose an interesting trade-off, with source transliteration supervised by heuristic cognate lists, but for transferring Machine Translation systems.

The approach I experiment with is direct *lexicalized* transfer. The underlying intuition is that, for languages as lexically close as dialects are, lexicalized features of appropriate granularity (typically, morphological endings) will be common to both languages, so that the lexicalized knowledge acquired through these features is readily usable on the target side. But as a matter of fact, I *partially delexicalize* the source tagger (that is, I drive it to prefer learning as much as possible in an unlexicalized manner), by modifying the feature representations during training, based on timing, on how lexicalized those features are, or on presumably shared wordforms and morphemes. Direct tagger transfer has already been explored by Yu et al. (2016), but with full delexicalization: in their proposal, taggers are trained using only unlexicalized features, like word length, frequency or context entropy.

One strong advantage of my proposal, which is not the case of Yu et al. (2016), is to address a truly zero-resource scenario, where no data at all is available on the target language, and no preprocessing is required;[2] for cases when raw data is available in the target language, I still propose an extension that leverages it. However, while Yu et al. (2016)'s method targets languages with large raw data (up to 20 million tokens, in order to compute accurate entropy), mine can be set up with target data as small as 400 tokens.

My method shares however an unfortunate property with delexicalized taggers: as improved as they are, the resulting accuracies (60% to 80% for quite similar pairs) remain too low to use the predicted tags as the basis of downstream tasks – Zeman et al. (2016) have notably reported their failure to pipeline a delexicalized tagger and a delexicalized parser. Nonetheless, this study still provides interesting insights for future work on lexical similarities.

C.1 Perceptron-based PoS tagging

The study is conducted using the feature-based tagger included in PanParser and based on the same structured prediction framework as the parser, described in Section 7.3. While this choice is not competitive with state-of-the-art PoS taggers (mostly neural and character-based), it provides a flexible experimental setting and therefore serves as a proof of concept for adapting the lexicalization of representations.

PoS tags are learned and predicted in a greedy way along the input sentence, using five kinds of contextual features: PoS-based features taken from earlier predictions (both words on the left, and their combination); the contextual words (two words in each direction); the word itself (and its combination with the tag of its left neighbour); all prefixes (1 to 6 characters) and suffixes (1 to 8) of the current word

[2]As a matter of fact, there remain two prerequisites, language identification and tokenization, but both can be done by language-independent means: language identification does not need to be very precise, just detecting its similarity with the source language is enough, and baseline tokenization can be achieved with a simple generic tokenizer, splitting on all punctuation marks.

and its context window; and finally the spelling pattern of the current word, stating how lowercase, uppercase and non-alphabetical characters follow one another (for instance proper nouns will appear as 'Xx').

Additionally, the default tagger implemented in PanParser builds explicitly a tag lexicon from training data, thereby keeping a record of known unambiguous words; it is used at test time to annotate such words in a deterministic way. However, in this context, despite their similarities the source and target vocabularies are still supposed to differ, and their ambiguities do not necessarily match; this functionality is consequently disabled. As a result, the monolingual accuracies reported here are slightly below the true PanParser baseline, especially for large training corpora, which would provide rich lexicons.

C.2 Modifying feature representations

The vanilla strategy for direct lexicalized transfer is completely straightforward: a standard model is trained in one language, and simply applied to input data of another language. The most interesting part is thus how to modify source training, so that the accuracy is not overly hurt by this change of language.

Overall, the source tagger should not learn shared knowledge based on source-specific features, which will not be available on the target side; it is thus tempting to discard such features at training time, otherwise the model will learn to rely on them, according to Section 6.3. But on the other hand, in cross-lingual contexts, they have also been shown to help generalization, by taking care of source-specific exceptions. Therefore, I consider keeping them during training, but with penalties that lead the model to use them only as a last resort. Two strategies are compared to this end: *scheduling* and *differentiated dropout*.

I start by classifying features according to their source specificity (or equivalently, to their lexicalization). The least lexicalized are obviously the PoS-based features (including bias parameters): if both languages are close enough, they have similar syntax, so that this information is reliable. As for exact wordforms, they will likely differ between both languages and are thus on the opposite end of the spectrum; but the most lexicalized features are those involving the current word, as the classifier may use them to encode a kind of internal lexicon, where PoS information is specified for each wordform independently (which would fail after transfer). In between remain the spelling pattern, which I consider almost unlexicalized, as it abstracts from the exact wordform but is not fully language-independent (in Japanese for instance, proper nouns cannot be identified by an initial capital), and prefixes and suffixes. The intuition regarding those is that between two very close languages, common lemmas or morphemes may occur; as such, information about them is surely lexicalized, but often remains transferable. Their usefulness depends on the feature: for instance for English-French transfer, information about '-ion' suffixes will be useful ('connection' translates as 'connexion'), but not '-cy' ('democracy' translates as 'démocracie').

The idea of a scheduling strategy is motivated by the fact, revealed in Section 6.3, that timing matters during training, because the earlier a given type of features is provided to the classifier, the more it will rely on that feature. Hence, if lexical information is hidden during the first epochs, the model will first exploit the unlexicalized parameters as much as possible, and then only consider the lexicalized ones, in cases of need. In practice, unlexicalized parameters are used from the first epoch,[3] spelling patterns are introduced at epoch 3, suffixes and prefixes at epoch 5, contextual words at epoch 7, and as a very last resort at epoch 9, the current word.[4]

The differentiated dropout strategy is based on another approach to prevent the classifier from relying on a given parameter: making it difficult to rely on by randomly discarding the corresponding feature. But here I want to penalize much more the (unreliable) lexicalized features than the purely unlexicalized ones, and thus use differentiated rates: dropout is set to 10% for the fully unlexicalized parameters, 30% for the spelling pattern, 50% for all suffixes and prefixes, 70% for the contextual wordforms, and 90% for the features involving the current wordform itself.

When target raw data is available, I experiment with an additional refinement of the dropout strategy, where the model is only allowed to train using features that may appear in target: features which seem non-transferable are filtered out (or in other terms, their dropout is set to 100%). In practice, I collect beforehand all wordforms, prefixes and suffixes appearing in the target raw corpus; then I modify the feature extraction function, so that it discards any feature involving prefixes, suffixes or wordforms that do not belong to that set.

C.3 Experiments

Experiments are done on a subset of UD 2.0 exhibiting interesting lexical similarities, which I seek to leverage. Three language families are considered: North Germanic,[5] Romance[6] and Slavic languages. However, because they do not use the same writing

[3]In contrary to the PanParser default, in that case greedy training is done *without* exploration during the first epoch. Indeed, when only PoS-based features are used, a clean prediction history is crucial to have at least a few reliable features.

[4]Note that by default in PanParser, the tagger is trained on batches of exponential sizes (1, 2, 4 configurations for epochs 1, 2 and 3, and so on). So, the most lexicalized features are unveiled at a time when not many update steps remain; this interaction probably affects training, notably in the languages with especially small data, but has not been further investigated.

[5]I use the Danish, Norwegian-Bokmaal (abbreviated as `no` in the results) and Swedish treebanks.

[6]The corresponding treebanks are Portuguese, Galician, Spanish, Catalan, French, Italian and Romanian.

Target:	da				no				sv				μ			
da	**92.7**	92.4	92.3		77.4	76.8	77.7	**78.1**	55.6	56.7	61.8	**64.2**	66.5	66.7	69.8	**71.1**
no	77.1	77.3	77.7	**78.9**	**95.9**	95.5	95.1		55.2	59.2	62.1	**64.8**	66.2	68.2	69.9	**71.9**
sv	56.8	57.8	60.2	**63.3**	59.8	59.6	**61.1**	60.5	**93.6**	93.0	93.5		58.3	58.7	60.7	**61.9**

Target:	ru				be				uk				bg				μ			
ru	**93.2**	92.9	92.9		56.2	55.2	55.7	**58.6**	55.6	54.2	**57.1**	55.0	54.5	54.1	55.8	**56.5**	55.4	54.5	56.2	**56.7**
be	48.9	**49.4**	47.8	46.5	81.2	80.2	**81.5**		47.8	46.2	47.6	**48.4**	39.8	**40.7**	40.1	38.5	**45.5**	45.5	45.2	44.5
uk	32.2	30.7	32.6	**34.6**	31.2	31.2	32.3	**35.7**	42.9	37.1	**43.9**		30.3	29.3	30.9	**35.0**	31.3	30.4	31.9	**35.1**
bg	58.6	**59.8**	59.5	57.9	49.3	**50.4**	47.8	46.1	47.6	47.7	**47.9**	43.0	**96.3**	95.9	96.3		51.8	**52.7**	51.7	49.0

Target:	cs				sk				sl				hr				pl				hsb				μ			
cs	**97.8**	97.6	97.7		75.7	74.5	78.6	**78.8**	51.1	51.2	**52.0**	49.7	54.5	54.5	**55.9**	53.8	48.0	47.9	49.2	**49.3**	52.2	52.7	**53.5**	43.5	56.3	56.2	**57.8**	55.0
sk	69.4	69.2	69.8	**70.2**	**91.9**	91.3	89.8		48.5	48.8	49.8	**50.7**	50.7	**51.2**	51.0	51.1	52.0	52.1	53.8	**56.4**	51.5	**52.4**	51.0	45.4	54.4	54.7	**55.1**	54.8
sl	55.4	55.4	57.8	**58.1**	58.5	58.6	**60.6**	59.9	**94.9**	94.1	94.5		63.3	63.1	**64.4**	63.9	44.8	45.1	45.4	**47.6**	52.1	52.2	**52.7**	45.5	54.8	54.9	**56.2**	55.0
hr	51.9	52.0	54.7	**55.1**	53.1	52.7	**54.4**	51.4	62.6	63.0	64.0	**65.0**	**94.8**	94.1	94.1		43.5	**45.0**	44.7	44.7	51.2	51.9	**53.1**	45.9	52.5	52.9	**54.2**	52.4
pl	46.4	**47.5**	47.2	47.1	50.5	**52.0**	51.0	51.0	38.1	**39.5**	38.9	39.4	47.9	48.4	47.3	**48.4**	**94.0**	92.9	93.2		57.8	**58.2**	57.0	54.2	48.1	**49.1**	48.3	48.0
hsb	**37.5**	31.3	35.3	33.4	**39.6**	33.9	38.2	34.9	**29.7**	25.6	27.8	25.1	**31.1**	27.0	27.9	29.3	**37.4**	30.6	35.6	34.7	**50.0**	42.3	47.6		**35.1**	29.7	32.9	31.5

Target:	pt				gl				es				ca				fr				it				ro				μ			
pt	95.3	**95.4**	94.8		77.6	78.3	**79.6**	79.5	52.5	53.3	54.3	**54.7**	44.2	**45.3**	45.0	43.4	32.3	**32.9**	32.1	31.7	27.4	**30.1**	28.7	27.3	31.0	**33.5**	31.1	30.1	44.2	**45.6**	45.1	44.4
gl	**70.2**	68.9	69.8	69.8	**95.4**	95.0	95.3		57.8	57.3	**58.4**	58.0	44.5	**45.1**	44.6	43.8	38.1	37.8	**38.6**	38.3	**30.6**	30.1	30.1	29.3	35.3	**36.3**	36.2	36.2	46.1	45.9	**46.3**	45.9
es	55.1	57.6	57.9	**58.4**	66.2	**68.6**	67.9	68.6	95.1	95.1	**95.3**		61.4	65.4	66.2	**66.3**	35.7	**39.3**	37.5	37.3	40.1	**44.0**	43.5	43.4	25.1	31.1	29.3	**33.2**	47.3	51.0	50.4	**51.2**
ca	48.9	49.5	48.9	**50.5**	50.3	**52.0**	51.9	51.0	66.3	**68.8**	67.7	68.0	**97.1**	96.9	96.7		52.6	**54.4**	54.4	53.9	36.9	39.5	**40.0**	37.9	33.4	35.3	**36.7**	35.8	48.1	49.9	**49.9**	49.5
fr	39.4	**42.8**	41.0	38.3	42.8	**45.7**	44.4	43.7	49.5	**50.7**	49.9	48.5	56.1	**57.7**	55.8	54.1	95.5	**95.6**	95.3		45.4	**48.3**	45.3	43.7	33.8	**36.4**	36.0	35.3	44.5	**46.9**	45.4	43.9
it	44.5	**49.4**	43.5	42.9	41.5	**48.0**	41.1	42.1	50.6	**57.4**	51.1	49.7	55.0	**59.2**	57.0	55.2	48.3	**55.3**	48.4	48.2	**96.3**	96.0	96.3		33.7	**37.9**	36.8	37.5	45.6	**51.2**	46.3	45.9
ro	44.6	**46.6**	44.0	43.5	43	45.8	45.3	**46.1**	37.9	**38.7**	35.8	36.5	38.9	**39.2**	37.2	36.9	35.9	**36.6**	35.9	35.8	28.2	29.2	28.4	**29.5**	**95.0**	94.9	94.8		38.1	**39.4**	37.8	38.1

TABLE C.1: PoS accuracies after direct lexicalized transfer, among the North Germanic, Cyrillic-script Slavic, Latin-script Slavic and Romance language families. Rows represent the sources, columns the targets (presented in a geographically-consistent order); for each source, μ values are averages over all targets (except the source itself). Each cell contains baseline || scheduling || dropout || dropout+filtering values (without filtering for the monolingual evaluations). Gray rows correspond to unrealistic sources, due to tiny training data.

system (thereby hiding lexical similarities in written texts), the Slavic family is split into two groups.[7]

Among each group, all source-target combinations are considered, even though some languages have only tiny training data (400 to 4,000 tokens) and are thus not realistic sources. For those, very low accuracies are expected, but for reasons independent of lexical divergences, which makes the results hard to interpret – for the sake of completeness, they are still reported.

For each pair, the baseline method (vanilla direct lexicalized transfer) is compared with the three improvements I have proposed (scheduling, dropout and dropout with filtering). For the filtering refinement, the UD trainset of the target language, stripped of all annotations, is used as raw data; note that its size varies a lot among the considered languages, from 400 to 1 million tokens. All taggers are trained on 10 epochs, which is sufficient for convergence in all cases.

Table C.1 reports monolingual and cross-lingual measures of PoS accuracy, among each group of languages. The best method for each language pair is written in bold, and for each target the best source model is displayed in red (excluding the target itself, which is underlined).

[7]For the Latin script group, I retain the Czech, Slovak, Slovenian, Croatian, Polish and Upper Sorbian treebanks. For the Cyrillic script group, I retain the Russian, Belarusian, Ukrainian and Bulgarian treebanks.

Regarding baseline accuracies, they appear in fact relatively high (60-80%) in many cases: in the North Germanic group, but also in the Romance and Latin-script Slavic groups, when considering geographically close languages (Czech-Slovak, Slovenian-Croatian, Portuguese-Galician, Spanish-Catalan), whose lexical similarity is close to that of a dialect scenario. For most other Romance and Slavic pairs however, baseline accuracy is below 60%.[8]

When applying feature-level penalties, monolingual accuracies slightly decrease in general: lexicalized features contain indeed valuable information, whose loss hinders accurate tagging of the most specific words. But in the cross-lingual settings, all methods I propose appear beneficial overall, suggesting that more general knowledge is actually extracted; these results are in fact consistent with those of Section 6.3 on parsing. In particular, the single-best source is significantly improved: the Slavic groups yield modest improvements of +1.3 and +1.7, but the gains reach +3.1 on the best Romance source, +5.4 for the North Germanic one.

Substantial improvements are notably observed with the filtering refinement of the dropout technique: dropout often outperforms the scheduling strategy, and is indeed further improved with appropriate feature selection. Two languages stand out on that matter: Swedish, which particularly benefits from that method (with around +10 for both sources), and Upper Sorbian, which endures a severe adverse impact (down to -10), but other results are more consistent.

C.4 Conclusion

This appendix has reported a series of experiments on methods to leverage lexical similarity, using direct lexicalized transfer on taggers from closely related languages. While the resulting accuracies are not competitive enough to produce usable taggers in the target languages, the experiments themselves provide valuable insights and perspectives for future work.

Based on a classification of features along their degree of lexicalization, I have compared two approaches, scheduling and dropout, for preventing the source tagger from relying on the most lexicalized features.

With both strategies, the information learned by the tagger appears more language-independent, thereby degrading monolingual accuracy but improving the efficiency of transfer. Also, an additional refinement is proposed, where features are selected on the basis of target raw data: with that method, the source tagger extracts knowledge that is even more rewarding from the target point of view, and achieves significantly higher transfer accuracies.

[8]The results from and to Romanian are particularly intriguing in the light of its numerous cognates, but this transfer failure may be explained by the frequent use in Romanian of some diacritics which do not exist in other languages. The same holds, to a lesser extent, for French diacritics.

The main track for future work in lexicalized tagger transfer is to reproduce such experiments with state-of-the-art taggers based on neural networks and character-level information; in that case, an additional difficulty resides in the interpretation of neural parameters, or at least identifying those which convey unlexicalized knowledge, which is in fact quite challenging, in particular when using recurrent networks. Beside PoS tagging, lexicalized transfer could also be entertained in other tasks, typically parsing, but as the interpretation of features is much less straightforward in parsing than in tagging, the method to enable such transfer remains to be devised.

Résumé détaillé

***

Les techniques dites de transfert cross-lingue permettent de pallier au manque de ressources dans une langue donnée, en exploitant des ressources disponibles dans d'autres langues mieux dotées (dites sources) pour construire des systèmes dans la langue peu dotée (dite cible). Or pour être performantes, ces méthodes restent soumises à la disponibilité de ressources cross-lingues de qualité, typiquement de larges corpus parallèles. L'approche cross-lingue permet ainsi d'augmenter légèrement la couverture linguistique des méthodes modernes de Traitement Automatique des Langues (TAL), coûteuses en ressources, mais la majorité des 7.000 langues existant au monde reste inaccessible.

Cette thèse explore plusieurs voies pour assouplir les restrictions liées aux ressources cross-lingues : cette diversification des ressources exploitables augmente significativement le nombre de langues pour lesquelles le TAL moderne est applicable. En premier lieu, de nouveaux types de ressources utilisables en contexte cross-lingue sont identifiés et exploités, tels que l'information typologique. Il est également proposé d'exploiter des sources distantes, y compris lorsque des sources plus proches sont disponibles : d'une part de nombreuses erreurs de transfert sont éliminées grâce à une technique de correction systématique des divergences les plus grossières, d'autre part des métriques dédiées permettent de catégoriser l'information extraite de ces sources distantes, afin de cibler les mots pour lesquels elles apportent une réelle plus-value. Cette thèse revisite en outre l'apport d'un petit corpus cible, en évaluant la quantité de données à partir de laquelle le transfert devient moins performant que l'approche monolingue classique ; c'est néanmoins la complémentarité des deux approches qui est finalement mise en évidence et modélisée. Enfin, une nouvelle architecture de transfert en cascade est développée pour exploiter de manière transparente la complémentarité de plusieurs approches, sources ou types de ressources.

Tout ce travail, instancié sur le cas d'étude de l'analyse en dépendance et accompagné de nombreux exemples linguistiques, est doublé d'un travail de fond sur les

analyseurs syntaxiques, afin d'assouplir leur cadre d'utilisation et ainsi rendre possibles des architectures de transfert plus diverses. Ces travaux revisitent notamment l'utilisation d'oracles dits dynamiques lors de l'apprentissage, initialement prévus pour mieux sélectionner les parties de l'espace de recherche à entraîner : il s'avère que leurs nombreux bénéfices dépassent ce cadre. Ces innovations techniques sont concrétisées par le développement de PanParser, un analyseur par transition open source dont l'accent est mis sur la modularité, dans le but de faciliter les expérimentations scientifiques avec un cadre le plus équitable et systématique possible.

Afin d'élargir les résultats obtenus en analyse syntaxique, tout en s'assurant que les langues peu dotées disposent effectivement de tous les éléments nécessaires au transfert, d'autres tâches sont également étudiées dans le cadre de projets connexes (transfert de parties du discours, transfert de modèles d'alignement de mots). Les réflexions menées sont aussi enrichies par des travaux en traduction automatique, domaine dont les problématiques de divergences entre langues rejoignent celles du transfert.

Un accent particulier est mis tout au long de cette thèse sur l'analyse des résultats empiriques, avec notamment une étude fine des capacités actuelles des analyseurs syntaxiques. Il en ressort une meilleure compréhension de leur fonctionnement interne, par exemple leur difficulté à généraliser l'information syntaxique lorsque l'information lexicalisée intervient tôt durant l'apprentissage, ou le déséquilibre apporté par les quelques prédictions difficiles qu'ils doivent opérer, par rapport aux nombreuses prédictions faciles voire déterministes, comme rattacher un mot à son voisin. Ces éléments ouvrent une série de nouvelles perspectives de recherche, y compris pour le traitement monolingue de langues bien dotées.

List of publications

***

Alexandre ALLAUZEN, Lauriane AUFRANT, Franck BURLOT, Ophélie LACROIX, Elena KNYAZEVA, Thomas LAVERGNE, Guillaume WISNIEWSKI, & François YVON (2016). LIMSI@WMT'16: Machine Translation of News. In *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, pp. 239–245. URL: http://www.aclweb.org/anthology/W16-2304.

Lauriane AUFRANT, & Guillaume WISNIEWSKI (2016). *PanParser: a Modular Implementation for Efficient Transition-Based Dependency Parsing*. Technical Report. LIMSI-CNRS. URL: https://hal.archives-ouvertes.fr/hal-01295590.

Lauriane AUFRANT, & Guillaume WISNIEWSKI (2017). LIMSI@CoNLL'17: UD Shared Task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Vancouver, Canada: Association for Computational Linguistics, pp. 163–173. URL: http://www.aclweb.org/anthology/K17-3017.

Lauriane AUFRANT, Guillaume WISNIEWSKI, & François YVON (2016a). Cross-lingual alignment transfer: a chicken-and-egg story? In *Proceedings of the Workshop on Multilingual and Cross-lingual Methods in NLP*. San Diego, California: Association for Computational Linguistics, pp. 35–44. URL: http://www.aclweb.org/anthology/W16-1205.

Lauriane AUFRANT, Guillaume WISNIEWSKI, & François YVON (2016b). Cross-lingual and Supervised Models for Morphosyntactic Annotation: a Comparison on Romanian. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Ed. by Nicoletta Calzolari (Conference CHAIR), Khalid CHOUKRI, Thierry DECLERCK, Sara GOGGI, Marko GROBELNIK, Bente MAEGAARD, Joseph MARIANI, Helene MAZO, Asuncion MORENO, Jan ODIJK, & Stelios PIPERIDIS. Portorož, Slovenia: European Language Resources Association (ELRA). ISBN: 978-2-9517408-9-1.

Lauriane AUFRANT, Guillaume WISNIEWSKI, & François YVON (2016c). Ne nous arrêtons pas en si bon chemin: améliorations de l'apprentissage global d'analyseurs en dépendances par transition. In *Actes de la 23e conférence sur le Traitement Automatique des Langues Naturelles*, pp. 248–261.

Lauriane AUFRANT, Guillaume WISNIEWSKI, & François YVON (2016d). Zero-resource Dependency Parsing: Boosting Delexicalized Cross-lingual Transfer with

Linguistic Knowledge. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, pp. 119–130. URL: `http://aclweb.org/anthology/C16-1012`.

Lauriane AUFRANT, Guillaume WISNIEWSKI, & François YVON (2017). Don't Stop Me Now! Using Global Dynamic Oracles to Correct Training Biases of Transition-Based Dependency Parsers. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 318–323. URL: `http://www.aclweb.org/anthology/E17-2051`.

Lauriane AUFRANT, Guillaume WISNIEWSKI, & François YVON (2018a). Exploiting Dynamic Oracles to Train Projective Dependency Parsers on Non-Projective Trees. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 413–419. URL: `http://www.aclweb.org/anthology/N18-2066`.

Lauriane AUFRANT, Guillaume WISNIEWSKI, & François YVON (2018b). Quantifying training challenges of dependency parsers. In *Proceedings of COLING 2018, the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico: The COLING 2018 Organizing Committee.

Ophélie LACROIX, Lauriane AUFRANT, Guillaume WISNIEWSKI, & François YVON (2016a). Apprentissage d'analyseur en dépendances cross-lingue par projection partielle de dépendances. In *Actes de la 23e conférence sur le Traitement Automatique des Langues Naturelles*, pp. 1–14.

Ophélie LACROIX, Lauriane AUFRANT, Guillaume WISNIEWSKI, & François YVON (2016b). Frustratingly Easy Cross-Lingual Transfer for Transition-Based Dependency Parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, pp. 1058–1063. URL: `http://www.aclweb.org/anthology/N16-1121`.

Jan-Thorsten PETER, Tamer ALKHOULI, Hermann NEY, Matthias HUCK, Fabienne BRAUNE, Alexander FRASER, Aleš TAMCHYNA, Ondřej BOJAR, Barry HADDOW, Rico SENNRICH, Frédéric BLAIN, Lucia SPECIA, Jan NIEHUES, Alex WAIBEL, Alexandre ALLAUZEN, Lauriane AUFRANT, Franck BURLOT, Elena KNYAZEVA, Thomas LAVERGNE, François YVON, Mārcis PINNIS, & Stella FRANK (2016). The QT21/HimL Combined Machine Translation System. In *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, pp. 344–355. URL: `http://www.aclweb.org/anthology/W16-2320`.

Bibliography

***

Željko AGIĆ (2017). Cross-Lingual Parser Selection for Low-Resource Languages. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*. Gothenburg, Sweden: Association for Computational Linguistics, pp. 1–10. URL: http://www.aclweb.org/anthology/W17-0401.

Željko AGIĆ, Jörg TIEDEMANN, Danijela MERKLER, Simon KREK, Kaja DOBROVOLJC, & Sara MOZE (2014). Cross-lingual Dependency Parsing of Related Languages with Rich Morphosyntactic Tagsets. In *Proceedings of the EMNLP'2014 Workshop on Language Technology for Closely Related Languages and Language Variants*. Doha, Qatar: Association for Computational Linguistics, pp. 13–24. URL: http://www.aclweb.org/anthology/W14-4203.

Željko AGIĆ, Dirk HOVY, & Anders SØGAARD (2015). If all you have is a bit of the Bible: Learning POS taggers for truly low-resource languages. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China: Association for Computational Linguistics, pp. 268–272. URL: http://www.aclweb.org/anthology/P15-2044.

Željko AGIĆ, Anders JOHANNSEN, Barbara PLANK, Héctor MARTÍNEZ ALONSO, Natalie SCHLUTER, & Anders SØGAARD (2016). Multilingual Projection for Parsing Truly Low-Resource Languages. In *Transactions of the Association for Computational Linguistics* 4, pp. 301–312. ISSN: 2307-387X. URL: https://transacl.org/ojs/index.php/tacl/article/view/869.

Ethem ALPAYDIN (1997). REx: Learning a rule and exceptions. In *International Computer Science Institute TR-97-040 Berkeley*.

Ethem ALPAYDIN, & Cenk KAYNAK (1998). Cascading classifiers. In *Kybernetika* 34.4, pp. 369–374.

Waleed AMMAR, George MULCAIRE, Miguel BALLESTEROS, Chris DYER, & Noah A. SMITH (2016a). Many Languages, One Parser. In *Transactions of the Association for Computational Linguistics* 4, pp. 431–444. ISSN: 2307-387X. URL: https://www.transacl.org/ojs/index.php/tacl/article/view/892.

Waleed AMMAR, George MULCAIRE, Yulia TSVETKOV, Guillaume LAMPLE, Chris DYER, & Noah A. SMITH (2016b). Massively multilingual word embeddings. In *arXiv preprint arXiv:1602.01925*.

Daniel ANDOR, Chris ALBERTI, David WEISS, Aliaksei SEVERYN, Alessandro PRESTA, Kuzman GANCHEV, Slav PETROV, & Michael COLLINS (2016). Globally Normalized Transition-Based Neural Networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 2442–2452. URL: http://www.aclweb.org/anthology/P16-1231.

Mikel ARTETXE, Gorka LABAKA, & Eneko AGIRRE (2017). Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 451–462. URL: http://aclweb.org/anthology/P17-1042.

Miguel BALLESTEROS, & Joakim NIVRE (2013). Going to the Roots of Dependency Parsing. In *Computational Linguistics* 39.1, pp. 5–13. URL: http://www.aclweb.org/anthology/J/J13/J13-1002.pdf.

Miguel BALLESTEROS, Chris DYER, & Noah A. SMITH (2015). Improved Transition-based Parsing by Modeling Characters instead of Words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 349–359. URL: http://aclweb.org/anthology/D15-1041.

Miguel BALLESTEROS, Yoav GOLDBERG, Chris DYER, & Noah A. SMITH (2016). Training with Exploration Improves a Greedy Stack LSTM Parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 2005–2010. URL: https://aclweb.org/anthology/D16-1211.

Carmen BANEA, Rada MIHALCEA, Janyce WIEBE, & Samer HASSAN (2008). Multilingual Subjectivity Analysis Using Machine Translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Honolulu, Hawaii: Association for Computational Linguistics, pp. 127–135. URL: http://www.aclweb.org/anthology/D08-1014.

Éléonor BARTENLIAN, Margot LACOUR, Matthieu LABEAU, Alexandre ALLAUZEN, Guillaume WISNIEWSKI, & François YVON (2016). Adaptation au domaine pour l'analyse morpho-syntaxique. In *Actes de la 24e conférence sur le Traitement Automatique des Langues Naturelles*, pp. 134–141.

Shai BEN-DAVID, John BLITZER, Koby CRAMMER, & Fernando PEREIRA (2007). Analysis of Representations for Domain Adaptation. In *Advances in Neural Information Processing Systems 19*. Ed. by P. B. SCHÖLKOPF, J. C. PLATT, & T. HOFFMAN. MIT Press, pp. 137–144. URL: http://papers.nips.cc/paper/2983-analysis-of-representations-for-domain-adaptation.pdf.

Emily M. BENDER (2009). Linguistically Naïve != Language Independent: Why NLP Needs Linguistic Typology. In *Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*

Athens, Greece: Association for Computational Linguistics, pp. 26–32. URL: http://www.aclweb.org/anthology/W09-0106.

Emily M. BENDER, Dan FLICKINGER, Stephan OEPEN, & Yi ZHANG (2011). Parser Evaluation over Local and Non-Local Deep Dependencies in a Large Corpus. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, pp. 397–408. URL: http://www.aclweb.org/anthology/D11-1037.

Taylor BERG-KIRKPATRICK, & Dan KLEIN (2010). Phylogenetic Grammar Induction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden: Association for Computational Linguistics, pp. 1288–1297. URL: http://www.aclweb.org/anthology/P10-1131.

Taylor BERG-KIRKPATRICK, Alexandre BOUCHARD-CÔTÉ, John DENERO, & Dan KLEIN (2010). Painless Unsupervised Learning with Features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California: Association for Computational Linguistics, pp. 582–590. URL: http://www.aclweb.org/anthology/N10-1083.

Luca BERTINETTO, João F. HENRIQUES, Jack VALMADRE, Philip TORR, & Andrea VEDALDI (2016). Learning feed-forward one-shot learners. In *Advances in Neural Information Processing Systems 29*. Ed. by D. D. LEE, M. SUGIYAMA, U. V. LUXBURG, I. GUYON, & R. GARNETT. Curran Associates, Inc., pp. 523–531. URL: http://papers.nips.cc/paper/6068-learning-feed-forward-one-shot-learners.pdf.

Arianna BISAZZA, & Marcello FEDERICO (2016). A Survey of Word Reordering in Statistical Machine Translation: Computational Models and Language Phenomena. In *Computational linguistics* 42.2, pp. 163–205.

Anders BJÖRKELUND, & Joakim NIVRE (2015). Non-Deterministic Oracles for Unrestricted Non-Projective Transition-Based Dependency Parsing. In *Proceedings of the 14th International Conference on Parsing Technologies*. Bilbao, Spain: Association for Computational Linguistics, pp. 76–86. URL: http://www.aclweb.org/anthology/W15-2210.

John BLITZER, Ryan MCDONALD, & Fernando PEREIRA (2006). Domain Adaptation with Structural Correspondence Learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Sydney, Australia: Association for Computational Linguistics, pp. 120–128. URL: http://www.aclweb.org/anthology/W/W06/W06-1615.

Andreea BODNARI (2014). Joint multilingual learning for coreference resolution. PhD thesis. Massachusetts Institute of Technology.

Cristina BOSCO, Felice DELL'ORLETTA, Simonetta MONTEMAGNI, Manuela SANGUINETTI, & Maria SIMI (2014). The EVALITA 2014 dependency parsing task. In *EVALITA 2014 Evaluation of NLP and Speech Tools for Italian*. Pisa University Press, pp. 1–8.

Léon BOTTOU (1991). Une Approche théorique de l'Apprentissage Connexionniste: Applications à la Reconnaissance de la Parole. PhD thesis. Orsay, France: Université de Paris XI. URL: http://leon.bottou.org/papers/bottou-91a.

Peter F. BROWN, Peter V. DESOUZA, Robert L. MERCER, Vincent J. DELLA PIETRA, & Jenifer C. LAI (1992). Class-Based n-gram Models of Natural Language. In *Computational linguistics* 18.4, pp. 467–479.

Peter F. BROWN, Vincent J. DELLA PIETRA, Stephen A. DELLA PIETRA, & Robert L. MERCER (1993). The Mathematics of Statistical Machine Translation: Parameter Estimation. In *Computational linguistics* 19.2, pp. 263–311.

David BURKETT, Slav PETROV, John BLITZER, & Dan KLEIN (2010). Learning Better Monolingual Models with Unannotated Bilingual Text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Uppsala, Sweden: Association for Computational Linguistics, pp. 46–54. URL: http://www.aclweb.org/anthology/W10-2906.

Rich CARUANA (1997). Multitask Learning. PhD thesis. Carnegie Mellon University Pittsburgh, PA.

Danqi CHEN, & Christopher MANNING (2014). A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 740–750. URL: http://www.aclweb.org/anthology/D14-1082.

Jinho D. CHOI, & Martha PALMER (2011). Getting the Most out of Transition-based Dependency Parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, pp. 687–692. URL: http://www.aclweb.org/anthology/P11-2121.

Christos CHRISTODOULOUPOULOS, & Mark STEEDMAN (2015). A massively parallel corpus: the Bible in 100 languages. In *Language resources and evaluation* 49.2, pp. 375–395.

Maximin COAVOUX, & Benoit CRABBÉ (2016). Neural Greedy Constituent Parsing with Dynamic Oracles. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 172–182. URL: http://www.aclweb.org/anthology/P16-1017.

Shay B. COHEN, Dipanjan DAS, & Noah A. SMITH (2011). Unsupervised Structure Prediction with Non-Parallel Multilingual Guidance. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, pp. 50–61. URL: http://www.aclweb.org/anthology/D11-1005.

Reed COKE, Ben KING, & Dragomir R. RADEV (2016). Classifying Syntactic Regularities for Hundreds of Languages. In *arXiv preprint arXiv:1603.08016*. URL: http://arxiv.org/abs/1603.08016.

Michael COLLINS (2002). Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 1–8. DOI: 10.3115/1118693.1118694. URL: http://www.aclweb.org/anthology/W02-1001.

Michael COLLINS, & Brian ROARK (2004). Incremental Parsing with the Perceptron Algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*. Barcelona, Spain, pp. 111–118. DOI: 10.3115/1218955.1218970. URL: http://www.aclweb.org/anthology/P04-1015.

Ronan COLLOBERT, & Jason WESTON (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. ACM, pp. 160–167.

Lisa CONATHAN, & Jeff GOOD (2000). Morphosyntactic reduplication in Chechen and Ingush. In *The proceedings from the panels of the Chicago Linguistic Society's thirty-sixth meeting*, pp. 36–2.

Alexandra CORNILESCU (2016). The Low Definite Article and the Evolution of the Romanian DP*. In *Generative Grammar in Geneva* 9, pp. 3–26.

Michael A. COVINGTON (2001). A Fundamental Algorithm for Dependency Parsing. In *Proceedings of the 39th annual ACM southeast conference*, pp. 95–102.

Brooke COWAN, Ivona KUČEROVÁ, & Michael COLLINS (2006). A Discriminative Model for Tree-to-Tree Translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Sydney, Australia: Association for Computational Linguistics, pp. 232–241. URL: http://www.aclweb.org/anthology/W/W06/W06-1628.

Koby CRAMMER, Mark DREDZE, & Alex KULESZA (2009). Multi-Class Confidence Weighted Algorithms. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Singapore: Association for Computational Linguistics, pp. 496–504. URL: http://www.aclweb.org/anthology/D/D09/D09-1052.

William CROFT (2002). *Typology and universals*. Cambridge University Press.

Anna CURREY, Alina KARAKANTA, & Jon DEHDARI (2016). Using Related Languages to Enhance Statistical Language Models. In *Proceedings of the NAACL Student Research Workshop*. San Diego, California: Association for Computational Linguistics, pp. 116–123. URL: http://www.aclweb.org/anthology/N16-2017.

Dipanjan DAS, & Slav PETROV (2011). Unsupervised Part-of-Speech Tagging with Bilingual Graph-Based Projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, pp. 600–609. URL: http://www.aclweb.org/anthology/P11-1061.

Hal DAUMÉ III, & Daniel MARCU (2005). Learning as Search Optimization: Approximate Large Margin Methods for Structured Prediction. In *Proceedings of the 22nd international conference on Machine learning*. ACM, pp. 169–176.

Hal DAUME III, & Daniel MARCU (2006). Domain adaptation for statistical classifiers. In *Journal of Artificial Intelligence Research* 26, pp. 101–126.

Hal DAUMÉ III, John LANGFORD, & Daniel MARCU (2009). Search-based Structured Prediction. In *Machine Learning* 75.3, pp. 297–325.

Shai Ben DAVID, Tyler LU, Teresa LUU, & Dávid PÁL (2010). Impossibility theorems for domain adaptation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 129–136.

Ludmila DIMITROVA, Nancy IDE, Vladimir PETKEVIC, Tomaz ERJAVEC, Heiki Jaan KAALEP, & Dan TUFIS (1998). Multext-East: Parallel and Comparable Corpora and Lexicons for Six Central and Eastern European Languages. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, pp. 315–319.

Gabriela Pană DINDELEGAN, & Martin MAIDEN (2013). *The grammar of Romanian*. Oxford University Press.

Pedro DOMINGOS (2000). A unified bias-variance decomposition for zero-one and squared loss. In *AAAI/IAAI* 2000, pp. 564–569.

Daxiang DONG, Hua WU, Wei HE, Dianhai YU, & Haifeng WANG (2015). Multi-Task Learning for Multiple Language Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, pp. 1723–1732. URL: http://www.aclweb.org/anthology/P15-1166.

Bonnie J. DORR (1994). Machine translation divergences: A formal description and proposed solution. In *Computational Linguistics* 20.4, pp. 597–633.

Markus DREYER, Keith HALL, & Sanjeev KHUDANPUR (2007). Comparing Reordering Constraints for SMT Using Efficient BLEU Oracle Computation. In *Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation*. Rochester, New York: Association for Computational Linguistics, pp. 103–110. URL: http://www.aclweb.org/anthology/W/W07/W07-0414.

Gregory DRUCK, Burr SETTLES, & Andrew MCCALLUM (2009). Active Learning by Labeling Features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Singapore: Association for Computational Linguistics, pp. 81–90. URL: http://www.aclweb.org/anthology/D/D09/D09-1009.

Matthew S. DRYER, & Martin HASPELMATH, eds. (2013). *WALS Online*. Leipzig: Max Planck Institute for Evolutionary Anthropology. URL: http://wals.info/.

Kevin DUH, Akinori FUJINO, & Masaaki NAGATA (2011). Is Machine Translation Ripe for Cross-Lingual Sentiment Classification? In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, pp. 429–433. URL: http://www.aclweb.org/anthology/P11-2075.

Long DUONG, Trevor COHN, Steven BIRD, & Paul COOK (2015a). A Neural Network Model for Low-Resource Universal Dependency Parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 339–348. URL: `http://aclweb.org/anthology/D15-1040`.

Long DUONG, Trevor COHN, Steven BIRD, & Paul COOK (2015b). Low Resource Dependency Parsing: Cross-lingual Parameter Sharing in a Neural Network Parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China: Association for Computational Linguistics, pp. 845–850. URL: `http://www.aclweb.org/anthology/P15-2139`.

Greg DURRETT, Adam PAULS, & Dan KLEIN (2012). Syntactic Transfer Using a Bilingual Lexicon. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Jeju Island, Korea: Association for Computational Linguistics, pp. 1–11. URL: `http://www.aclweb.org/anthology/D12-1001`.

Chris DYER, Victor CHAHUNEAU, & Noah A. SMITH (2013). A Simple, Fast, and Effective Reparameterization of IBM Model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, pp. 644–648. URL: `http://www.aclweb.org/anthology/N13-1073`.

Chris DYER, Miguel BALLESTEROS, Wang LING, Austin MATTHEWS, & Noah A. SMITH (2015). Transition-Based Dependency Parsing with Stack Long Short-Term Memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, pp. 334–343. URL: `http://www.aclweb.org/anthology/P15-1033`.

Andreas EISELE, & Yu CHEN (2010). MultiUN: A Multilingual Corpus from United Nation Documents. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. Ed. by Nicoletta Calzolari (Conference CHAIR), Khalid CHOUKRI, Bente MAEGAARD, Joseph MARIANI, Jan ODIJK, Stelios PIPERIDIS, Mike ROSNER, & Daniel TAPIAS. Valletta, Malta: European Language Resources Association (ELRA). ISBN: 2-9517408-6-7.

Jason EISNER, & Noah A. SMITH (2005). Parsing with Soft and Hard Constraints on Dependency Length. In *Proceedings of the Ninth International Workshop on Parsing Technology*. Vancouver, British Columbia: Association for Computational Linguistics, pp. 30–41. URL: `http://www.aclweb.org/anthology/W/W05/W05-1504`.

Akiko ERIGUCHI, Kazuma HASHIMOTO, & Yoshimasa TSURUOKA (2016). Tree-to-Sequence Attentional Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 823–833. URL: `http://www.aclweb.org/anthology/P16-1078`.

Tomaž ERJAVEC (2010). MULTEXT-East Version 4: Multilingual Morphosyntactic Specifications, Lexicons and Corpora. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. Ed. by Nicoletta Calzolari (Conference CHAIR), Khalid CHOUKRI, Bente MAEGAARD, Joseph MARIANI, Jan ODIJK, Stelios PIPERIDIS, Mike ROSNER, & Daniel TAPIAS. Valletta, Malta: European Language Resources Association (ELRA). ISBN: 2-9517408-6-7.

Nicholas EVANS, & Stephen C. LEVINSON (2009). The myth of language universals: Language diversity and its importance for cognitive science. In *Behavioral and brain sciences* 32.05, pp. 429–448.

Daniel L. EVERETT (1986). Pirahã. In *Handbook of Amazonian languages*. Ed. by Desmond C. DERBYSHIRE, & Geoffrey K. PULLUM. Vol. 1. Berlin: Mouton de Gruyter, pp. 200–325.

Anna FELDMAN (2006). Portable language technology: a resource-light approach to morpho-syntactic tagging. PhD thesis. The Ohio State University.

Alejandro Moreo FERNÁNDEZ, Andrea ESULI, & Fabrizio SEBASTIANI (2015). Distributional correspondence indexing for cross-lingual and cross-domain sentiment classification. In *Journal of artificial intelligence research* 55, pp. 131–163.

Daniel FERNÁNDEZ-GONZÁLEZ, & Carlos GÓMEZ-RODRÍGUEZ (2012). Improving Transition-Based Dependency Parsing with Buffer Transitions. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Jeju Island, Korea: Association for Computational Linguistics, pp. 308–319. URL: http://www.aclweb.org/anthology/D12-1029.

Orhan FIRAT, Kyunghyun CHO, & Yoshua BENGIO (2016a). Multi-Way, Multilingual Neural Machine Translation with a Shared Attention Mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, pp. 866–875. URL: http://www.aclweb.org/anthology/N16-1101.

Orhan FIRAT, Baskaran SANKARAN, Yaser AL-ONAIZAN, Fatos T. YARMAN VURAL, & Kyunghyun CHO (2016b). Zero-Resource Translation with Multi-Lingual Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 268–277. URL: https://aclweb.org/anthology/D16-1026.

Victoria FOSSUM, & Steven ABNEY (2005). Automatically inducing a part-of-speech tagger by projecting from multiple source languages across aligned corpora. In *Lecture notes in computer science* 3651, p. 862.

Alexander FRASER, & Daniel MARCU (2007). Measuring word alignment quality for statistical machine translation. In *Computational Linguistics* 33.3, pp. 293–303.

Lyn FRAZIER (1979). On Comprehending Sentences: Syntactic Parsing Strategies. PhD thesis. University of Connecticut.

Yoav FREUND, & Robert E. SCHAPIRE (1999). Large margin classification using the perceptron algorithm. In *Machine learning* 37.3, pp. 277–296.

João GAMA, & Pavel BRAZDIL (2000). Cascade generalization. In *Machine learning* 41.3, pp. 315–343.

Kuzman GANCHEV, Jennifer GILLENWATER, & Ben TASKAR (2009). Dependency Grammar Induction via Bitext Projection Constraints. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Association for Computational Linguistics, pp. 369–377. URL: http://www.aclweb.org/anthology/P/P09/P09-1042.

Dan GARRETTE, & Jason BALDRIDGE (2013). Learning a Part-of-Speech Tagger from Two Hours of Annotation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, pp. 138–147. URL: http://www.aclweb.org/anthology/N13-1014.

Ryan GEORGI, Fei XIA, & William LEWIS (2010). Comparing Language Similarity across Genetic and Typologically-Based Groupings. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Beijing, China: Coling 2010 Organizing Committee, pp. 385–393. URL: http://www.aclweb.org/anthology/C10-1044.

Ryan GEORGI, Fei XIA, & William LEWIS (2012). Measuring the Divergence of Dependency Structures Cross-Linguistically to Improve Syntactic Projection Algorithms. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. Ed. by Nicoletta Calzolari (Conference CHAIR), Khalid CHOUKRI, Thierry DECLERCK, Mehmet Uğur DOĞAN, Bente MAEGAARD, Joseph MARIANI, Asuncion MORENO, Jan ODIJK, & Stelios PIPERIDIS. Istanbul, Turkey: European Language Resources Association (ELRA). ISBN: 978-2-9517408-7-7.

Arnab GHOSHAL, Pawel SWIETOJANSKI, & Steve RENALS (2013). Multilingual training of deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, pp. 7319–7323.

Kevin GIMPEL, & Noah A. SMITH (2014). Phrase dependency machine translation with quasi-synchronous tree-to-tree features. In *Computational Linguistics* 40.2, pp. 349–401.

Yoav GOLDBERG, & Michael ELHADAD (2010). An Efficient Algorithm for Easy-First Non-Directional Dependency Parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California: Association for Computational Linguistics, pp. 742–750. URL: http://www.aclweb.org/anthology/N10-1115.

Yoav GOLDBERG, & Joakim NIVRE (2012). A Dynamic Oracle for Arc-Eager Dependency Parsing. In *Proceedings of COLING 2012*. Mumbai, India: The COLING 2012 Organizing Committee, pp. 959–976. URL: http://www.aclweb.org/anthology/C12-1059.

Yoav GOLDBERG, & Joakim NIVRE (2013). Training Deterministic Parsers with Non-Deterministic Oracles. In *Transactions of the Association for Computational Linguistics* 1, pp. 403–414. ISSN: 2307-387X. URL: https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/145.

Yoav GOLDBERG, Francesco SARTORIO, & Giorgio SATTA (2014). A Tabular Method for Dynamic Oracles in Transition-Based Parsing. In *Transactions of the Association for Computational Linguistics* 2, pp. 119–130. ISSN: 2307-387X. URL: https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/302.

Carlos GÓMEZ-RODRÍGUEZ, & Daniel FERNÁNDEZ-GONZÁLEZ (2015). An Efficient Dynamic Oracle for Unrestricted Non-Projective Parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China: Association for Computational Linguistics, pp. 256–261. URL: http://www.aclweb.org/anthology/P15-2042.

Carlos GÓMEZ-RODRÍGUEZ, & Joakim NIVRE (2010). A Transition-Based Parser for 2-Planar Dependency Structures. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden: Association for Computational Linguistics, pp. 1492–1501. URL: http://www.aclweb.org/anthology/P10-1151.

Carlos GÓMEZ-RODRÍGUEZ, Iago ALONSO-ALONSO, & David VILARES (2017). How Important is Syntactic Parsing Accuracy? An Empirical Evaluation on Rule-Based Sentiment Analysis. In *Artificial Intelligence Review*, pp. 1–17.

Jeff GOOD (2003). Clause combining in Chechen. In *Studies in Language. International Journal sponsored by the Foundation 'Foundations of Language'* 27.1, pp. 113–170.

Stephan GOUWS, Yoshua BENGIO, & Greg CORRADO (2015). Bilbowa: Fast bilingual distributed representations without word alignments. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 748–756.

Joseph H. GREENBERG (1963). Some universals of grammar with particular reference to the order of meaningful elements. In *Universals of language* 2, pp. 73–113.

Jiang GUO, Wanxiang CHE, David YAROWSKY, Haifeng WANG, & Ting LIU (2015). Cross-lingual Dependency Parsing Based on Distributed Representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, pp. 1234–1244. URL: http://www.aclweb.org/anthology/P15-1119.

Jiang GUO, Wanxiang CHE, David YAROWSKY, Haifeng WANG, & Ting LIU (2016). A Representation Learning Framework for Multi-Source Transfer Parsing. In *AAAI*, pp. 2734–2740.

Thanh-Le HA, Jan NIEHUES, & Alexander WAIBEL (2016). Toward multilingual neural machine translation with universal encoder and decoder. In *arXiv preprint arXiv:1611.04798*.

Jirka HANA, Anna FELDMAN, Chris BREW, & Luiz AMARAL (2006). Tagging Portuguese with a Spanish tagger using cognates. In *Proceedings of the International Workshop on Cross-Language Knowledge Induction*. Association for Computational Linguistics, pp. 33–40.

Alice C. HARRIS, & Lyle CAMPBELL (1995). *Historical syntax in cross-linguistic perspective*. Vol. 74. Cambridge University Press.

Martin HASPELMATH, Matthew S. DRYER, David GIL, & Bernard COMRIE (2005). *The World Atlas of Language Structures*. Oxford Univ. Press.

Martin HAULRICH (2010). Transition-Based Parsing with Confidence-Weighted Classification. In *Proceedings of the ACL 2010 Student Research Workshop*. Uppsala, Sweden: Association for Computational Linguistics, pp. 55–60. URL: http://www.aclweb.org/anthology/P10-3010.

John A. HAWKINS (1983). *Word order universals: Quantitative analyses of linguistic structure*.

John A. HAWKINS (1994). *A performance theory of order and constituency*. Vol. 73. Cambridge University Press.

James HENDERSON (2004). Discriminative Training of a Neural Network Statistical Parser. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*. Barcelona, Spain, pp. 95–102. DOI: 10.3115/1218955.1218968. URL: http://www.aclweb.org/anthology/P04-1013.

Karl Moritz HERMANN, & Phil BLUNSOM (2013). A Simple Model for Learning Multilingual Compositional Semantics. In *arXiv preprint arXiv:1312.6173*. URL: http://arxiv.org/abs/1312.6173.

Tin HO, Mitra BASU, & Martin LAW (2006). Measures of geometrical complexity in classification problems. In *Data complexity in pattern recognition*, pp. 1–23.

Tin Kam HO, & Mitra BASU (2000). Measuring the complexity of classification problems. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*. Vol. 2. IEEE, pp. 43–47.

Maria HOLMQVIST, & Lars AHRENBERG (2011). A gold standard for English–Swedish word alignment. In *Proceedings of the 18th Nordic Conference of Computational Linguistics NODALIDA*. Vol. 11, pp. 106–113.

Matthew HONNIBAL, & Mark JOHNSON (2015). An Improved Non-monotonic Transition System for Dependency Parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1373–1378. URL: http://aclweb.org/anthology/D15-1162.

Matthew HONNIBAL, Yoav GOLDBERG, & Mark JOHNSON (2013). A Non-Monotonic Arc-Eager Transition System for Dependency Parsing. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 163–172. URL: http://www.aclweb.org/anthology/W13-3518.

Jui-Ting HUANG, Jinyu LI, Dong YU, Li DENG, & Yifan GONG (2013). Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, pp. 7304–7308.

Kejun HUANG, Matt GARDNER, Evangelos PAPALEXAKIS, Christos FALOUTSOS, Nikos SIDIROPOULOS, Tom MITCHELL, Partha P. TALUKDAR, & Xiao FU (2015). Translation Invariant Word Embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1084–1088. URL: http://aclweb.org/anthology/D15-1127.

Liang HUANG, & Kenji SAGAE (2010). Dynamic Programming for Linear-Time Incremental Parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden: Association for Computational Linguistics, pp. 1077–1086. URL: http://www.aclweb.org/anthology/P10-1110.

Liang HUANG, Suphan FAYONG, & Yang GUO (2012). Structured Perceptron with Inexact Search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Montréal, Canada: Association for Computational Linguistics, pp. 142–151. URL: http://www.aclweb.org/anthology/N12-1015.

Rebecca HWA, Philip RESNIK, Amy WEINBERG, & Okan KOLAK (2002). Evaluating Translational Correspondence using Annotation Projection. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, pp. 392–399. DOI: 10.3115/1073083.1073149. URL: http://www.aclweb.org/anthology/P02-1050.

Rebecca HWA, Philip RESNIK, Amy WEINBERG, Clara CABEZAS, & Okan KOLAK (2005). Bootstrapping parsers via syntactic projection across parallel texts. In *Natural language engineering* 11.03, pp. 311–325.

Richard JOHANSSON, & Pierre NUGUES (2006). Investigating Multilingual Dependency Parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*. New York City: Association for Computational Linguistics, pp. 206–210. URL: http://www.aclweb.org/anthology/W/W06/W06-2930.

Melvin JOHNSON, Mike SCHUSTER, Quoc LE, Maxim KRIKUN, Yonghui WU, Zhifeng CHEN, Nikhil THORAT, Fernand a VIÉGAS, Martin WATTENBERG, Greg CORRADO, Macduff HUGHES, & Jeffrey DEAN (2017). Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. In *Transactions of the Association for Computational Linguistics* 5, pp. 339–351. ISSN: 2307-387X. URL: https://transacl.org/ojs/index.php/tacl/article/view/1081.

David KAMHOLZ, Jonathan POOL, & Susan COLOWICK (2014). PanLex: Building a Resource for Panlingual Lexical Translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Ed. by Nicoletta Calzolari (Conference CHAIR), Khalid CHOUKRI, Thierry DECLERCK, Hrafn LOFTSSON, Bente MAEGAARD, Joseph MARIANI, Asuncion MORENO, Jan ODIJK, & Stelios

PIPERIDIS. Reykjavik, Iceland: European Language Resources Association (ELRA). ISBN: 978-2-9517408-8-4.

Cenk KAYNAK, & Ethem ALPAYDIN (2000). MultiStage Cascading of Multiple Classifiers: One Man's Noise is Another Man's Data. In *Proceedings of the Seventeenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., pp. 455–462.

Alexandre KLEMENTIEV, Ivan TITOV, & Binod BHATTARAI (2012). Inducing Crosslingual Distributed Representations of Words. In *Proceedings of COLING 2012*. Mumbai, India: The COLING 2012 Organizing Committee, pp. 1459–1474. URL: http://www.aclweb.org/anthology/C12-1089.

Philipp KOEHN (2005). Europarl: A parallel corpus for statistical machine translation. In *MT summit*. Vol. 5, pp. 79–86.

Arne KÖHN, & Wolfgang MENZEL (2014). Incremental Predictive Parsing with TurboParser. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Baltimore, Maryland: Association for Computational Linguistics, pp. 803–808. URL: http://www.aclweb.org/anthology/P14-2130.

Grzegorz KONDRAK, Daniel MARCU, & Kevin KNIGHT (2003). Cognates Can Improve Statistical Translation Models. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2*. Association for Computational Linguistics, pp. 46–48.

Mikhail KOZHEVNIKOV, & Ivan TITOV (2014). Cross-lingual Model Transfer Using Feature Representation Projection. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Baltimore, Maryland: Association for Computational Linguistics, pp. 579–585. URL: http://www.aclweb.org/anthology/P14-2095.

Miroslav KUBAT, & Stan MATWIN (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *ICML*. Vol. 97. Nashville, USA, pp. 179–186.

Marco KUHLMANN, Carlos GÓMEZ-RODRÍGUEZ, & Giorgio SATTA (2011). Dynamic Programming Algorithms for Transition-Based Dependency Parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, pp. 673–682. URL: http://www.aclweb.org/anthology/P11-1068.

Shankar KUMAR, & William BYRNE (2005). Local Phrase Reordering Models for Statistical Machine Translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Vancouver, British Columbia, Canada: Association for Computational Linguistics, pp. 161–168. URL: http://www.aclweb.org/anthology/H/H05/H05-1021.

Shankar KUMAR, Franz J. OCH, & Wolfgang MACHEREY (2007). Improving Word Alignment with Bridge Languages. In *Proceedings of the 2007 Joint Conference on*

Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL). Prague, Czech Republic: Association for Computational Linguistics, pp. 42–50. URL: http://www.aclweb.org/anthology/D/D07/D07-1005.

Ludmila I. KUNCHEVA (2004). *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons.

John LAFFERTY, Andrew MCCALLUM, & Fernando C.N. PEREIRA (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference on Machine Learning*. Vol. 951, pp. 282–289.

Patrik LAMBERT, Yanjun MA, Sylwia OZDOWSKA, & Andy WAY (2009). Tracking Relevant Alignment Characteristics for Machine Translation. In *MT Summit XII-The twelfth Machine Translation Summit*, pp. 26–30.

Patrik LAMBERT, Simon PETITRENAUD, Yanjun MA, & Andy WAY (2010). Statistical Analysis of Alignment Characteristics for Phrase-based Machine Translation. In *Proceedings of the 14th European Association for Machine Translation*.

Tomer LEVINBOIM, & David CHIANG (2015). Multi-Task Word Alignment Triangulation for Low-Resource Languages. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, pp. 1221–1226. URL: http://www.aclweb.org/anthology/N15-1129.

Omer LEVY, Anders SØGAARD, & Yoav GOLDBERG (2017). A Strong Baseline for Learning Cross-Lingual Word Embeddings from Sentence Alignments. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 765–774. URL: http://www.aclweb.org/anthology/E17-1072.

William D. LEWIS, & Fei XIA (2008). Automatically Identifying Computationally Relevant Typological Features. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*, pp. 685–690.

Miryam DE LHONEUX, & Joakim NIVRE (2016). Should Have, Would Have, Could Have. Investigating Verb Group Representations for Parsing with Universal Dependencies. In *Proceedings of the Workshop on Multilingual and Cross-lingual Methods in NLP*. San Diego, California: Association for Computational Linguistics, pp. 10–19. URL: http://www.aclweb.org/anthology/W16-1202.

Miryam DE LHONEUX, Sara STYMNE, & Joakim NIVRE (2017). Old School vs. New School: Comparing Transition-Based Parsers with and without Neural Network Enhancement. In *TLT*, pp. 99–110.

Jiwei LI, Xinlei CHEN, Eduard HOVY, & Dan JURAFSKY (2016). Visualizing and Understanding Neural Models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, pp. 681–691. URL: http://www.aclweb.org/anthology/N16-1082.

Ling Li, & Yaser S. Abu-Mostafa (2006). *Data Complexity in Machine Learning*. Tech. rep. California Institute of Technology.

Zhenghua Li, Min Zhang, & Wenliang Chen (2014). Soft Cross-lingual Syntax Projection for Dependency Parsing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, pp. 783–793. URL: http://www.aclweb.org/anthology/C14-1075.

Adam Lopez (2009). Translation as Weighted Deduction. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*. Athens, Greece: Association for Computational Linguistics, pp. 532–540. URL: http://www.aclweb.org/anthology/E09-1061.

Adam Lopez, & Philip Resnik (2006). Word-Based Alignment, Phrase-Based Translation: What's the Link. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pp. 90–99.

Bin Lu, Chenhao Tan, Claire Cardie, & Benjamin K. Tsou (2011). Joint Bilingual Sentiment Classification with Unlabeled Parallel Corpora. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, pp. 320–330. URL: http://www.aclweb.org/anthology/P11-1033.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, & Lukasz Kaiser (2016). Multi-task Sequence to Sequence Learning. In *International Conference on Learning Representations (ICLR)*. San Juan, Puerto Rico.

Teresa Lynn, Ozlem Cetinoglu, Jennifer Foster, Elaine Uí Dhonnchadha, Mark Dras, & Josef van Genabith (2012). Irish Treebanking and Parsing: A Preliminary Evaluation. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. Ed. by Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, & Stelios Piperidis. Istanbul, Turkey: European Language Resources Association (ELRA). ISBN: 978-2-9517408-7-7.

Xuezhe Ma, & Fei Xia (2014). Unsupervised Dependency Parsing with Transferring Distribution via Parallel Guidance and Entropy Regularization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, pp. 1337–1348. URL: http://www.aclweb.org/anthology/P14-1126.

Yanjun Ma, Nicolas Stroppa, & Andy Way (2007). Bootstrapping Word Alignment via Word Packing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Prague, Czech Republic: Association for Computational Linguistics, pp. 304–311. URL: http://www.aclweb.org/anthology/P07-1039.

Christopher Manning (2011). Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *Computational Linguistics and Intelligent Text Processing*, pp. 171–189.

David MAREČEK, & Milan STRAKA (2013). Stop-probability estimates computed on a large corpus improve Unsupervised Dependency Parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 281–290. URL: http://www.aclweb.org/anthology/P13-1028.

Héctor MARTÍNEZ ALONSO, Djamé SEDDAH, & Benoît SAGOT (2016). From Noisy Questions to Minecraft Texts: Annotation Challenges in Extreme Syntax Scenario. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*. Osaka, Japan: The COLING 2016 Organizing Committee, pp. 13–23. URL: http://aclweb.org/anthology/W16-3905.

Héctor MARTÍNEZ ALONSO, Željko AGIĆ, Barbara PLANK, & Anders SØGAARD (2017). Parsing Universal Dependencies without training. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 230–240. URL: http://www.aclweb.org/anthology/E17-1022.

André F. T. MARTINS (2015). Transferring Coreference Resolvers with Posterior Regularization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, pp. 1427–1437. URL: http://www.aclweb.org/anthology/P15-1138.

Thomas MAYER, & Michael CYSOUW (2014). Creating a Massively Parallel Bible Corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Ed. by Nicoletta Calzolari (Conference CHAIR), Khalid CHOUKRI, Thierry DECLERCK, Hrafn LOFTSSON, Bente MAEGAARD, Joseph MARIANI, Asuncion MORENO, Jan ODIJK, & Stelios PIPERIDIS. Reykjavik, Iceland: European Language Resources Association (ELRA). ISBN: 978-2-9517408-8-4.

David MCCLOSKY, Eugene CHARNIAK, & Mark JOHNSON (2006). Effective Self-Training for Parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. New York City, USA: Association for Computational Linguistics, pp. 152–159. URL: http://www.aclweb.org/anthology/N/N06/N06-1020.

Ryan MCDONALD, & Joakim NIVRE (2007). Characterizing the Errors of Data-Driven Dependency Parsing Models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: Association for Computational Linguistics, pp. 122–131. URL: http://www.aclweb.org/anthology/D/D07/D07-1013.

Ryan MCDONALD, Fernando PEREIRA, Kiril RIBAROV, & Jan HAJIC (2005). Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Vancouver, British Columbia, Canada: Association for Computational Linguistics, pp. 523–530. URL: http://www.aclweb.org/anthology/H/H05/H05-1066.

Ryan McDONALD, Slav PETROV, & Keith HALL (2011). Multi-Source Transfer of Delexicalized Dependency Parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, pp. 62–72. URL: http://www.aclweb.org/anthology/D11-1006.

Ryan McDONALD, Joakim NIVRE, Yvonne QUIRMBACH-BRUNDAGE, Yoav GOLDBERG, Dipanjan DAS, Kuzman GANCHEV, Keith HALL, Slav PETROV, Hao ZHANG, Oscar TÄCKSTRÖM, Claudia BEDINI, Núria BERTOMEU CASTELLÓ, & Jungmee LEE (2013). Universal Dependency Annotation for Multilingual Parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 92–97. URL: http://www.aclweb.org/anthology/P13-2017.

Avihai MEJER, & Koby CRAMMER (2010). Confidence in Structured-Prediction Using Confidence-Weighted Models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Cambridge, MA: Association for Computational Linguistics, pp. 971–981. URL: http://www.aclweb.org/anthology/D10-1095.

Tomas MIKOLOV, Quoc V. LE, & Ilya SUTSKEVER (2013). Exploiting Similarities among Languages for Machine Translation. In *arXiv preprint arXiv:1309.4168*. URL: http://arxiv.org/abs/1309.4168.

George A. MILLER (1995). WordNet: a lexical database for English. In *Communications of the ACM* 38.11, pp. 39–41.

Seyed Abolghasem MIRROSHANDEL, & Alexis NASR (2011). Active Learning for Dependency Parsing Using Partially Annotated Sentences. In *Proceedings of the 12th International Conference on Parsing Technologies*. Dublin, Ireland: Association for Computational Linguistics, pp. 140–149. URL: http://www.aclweb.org/anthology/W11-2917.

Verginica Barbu MITITELU, & Radu ION (2005). Cross-Language Transfer Of Syntactic Relations Using Parallel Corpora. In *Cross-Language Knowledge Induction Workshop, Romania*.

Robert C. MOORE (2005). A Discriminative Framework for Bilingual Word Alignment. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Vancouver, British Columbia, Canada: Association for Computational Linguistics, pp. 81–88. URL: http://www.aclweb.org/anthology/H/H05/H05-1011.

Preslav NAKOV, & Hwee Tou NG (2012). Improving statistical machine translation for a resource-poor language using related resource-rich languages. In *Journal of Artificial Intelligence Research* 44, pp. 179–222.

Tahira NASEEM, Regina BARZILAY, & Amir GLOBERSON (2012). Selective Sharing for Multilingual Dependency Parsing. In *Proceedings of the 50th Annual Meeting* *of the Association for Computational Linguistics (Volume 1: Long Papers)*. Jeju Island, Korea: Association for Computational Linguistics, pp. 629–637. URL: http://www.aclweb.org/anthology/P12-1066.

Vincent NG, Sajib DASGUPTA, & S. M. Niaz ARIFIN (2006). Examining the Role of Linguistic Knowledge Sources in the Automatic Identification and Classification of Reviews. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*. Sydney, Australia: Association for Computational Linguistics, pp. 611–618. URL: http://www.aclweb.org/anthology/P/P06/P06-2079.

Joakim NIVRE (2003). An Efficient Algorithm for Projective Dependency Parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies*. IWPT 2003. Nancy, France.

Joakim NIVRE (2004). Incrementality in Deterministic Dependency Parsing. In *Proceedings of the ACL Workshop Incremental Parsing: Bringing Engineering and Cognition Together*. Ed. by Frank KELLER, Stephen CLARK, Matthew CROCKER, & Mark STEEDMAN. Barcelona, Spain: Association for Computational Linguistics, pp. 50–57.

Joakim NIVRE (2009). Non-Projective Dependency Parsing in Expected Linear Time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Association for Computational Linguistics, pp. 351–359. URL: http://www.aclweb.org/anthology/P/P09/P09-1040.

Joakim NIVRE, & Chiao-Ting FANG (2017). Universal Dependency Evaluation. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*. Gothenburg, Sweden: Association for Computational Linguistics, pp. 86–95. URL: http://www.aclweb.org/anthology/W17-0411.

Joakim NIVRE, & Daniel FERNÁNDEZ-GONZÁLEZ (2014). Arc-eager parsing with the tree constraint. In *Computational linguistics* 40.2, pp. 259–267.

Joakim NIVRE, & Ryan MCDONALD (2008). Integrating Graph-Based and Transition-Based Dependency Parsers. In *Proceedings of ACL-08: HLT*. Columbus, Ohio: Association for Computational Linguistics, pp. 950–958. URL: http://www.aclweb.org/anthology/P/P08/P08-1108.

Joakim NIVRE, & Jens NILSSON (2005). Pseudo-Projective Dependency Parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Ann Arbor, Michigan: Association for Computational Linguistics, pp. 99–106. DOI: 10.3115/1219840.1219853. URL: http://www.aclweb.org/anthology/P05-1013.

Joakim NIVRE, Johan HALL, & Jens NILSSON (2006). MaltParser: A Data-Driven Parser-Generator for Dependency Parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*. Vol. 6, pp. 2216–2219.

Joakim NIVRE, Laura RIMELL, Ryan MCDONALD, & Carlos GÓMEZ RODRÍGUEZ (2010). Evaluation of Dependency Parsers on Unbounded Dependencies. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling*

2010). Beijing, China: Coling 2010 Organizing Committee, pp. 833–841. URL: http://www.aclweb.org/anthology/C10-1094.

Joakim NIVRE, Yoav GOLDBERG, & Ryan MCDONALD (2014). Constrained arc-eager dependency parsing. In *Computational Linguistics* 40.2, pp. 249–527.

Joakim NIVRE, Marie-Catherine DE MARNEFFE, Filip GINTER, Yoav GOLDBERG, Jan HAJIC, Christopher D. MANNING, Ryan MCDONALD, Slav PETROV, Sampo PYYSALO, Natalia SILVEIRA, Reut TSARFATY, & Daniel ZEMAN (2016). Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Ed. by Nicoletta Calzolari (Conference CHAIR), Khalid CHOUKRI, Thierry DECLERCK, Sara GOGGI, Marko GROBELNIK, Bente MAEGAARD, Joseph MARIANI, Helene MAZO, Asuncion MORENO, Jan ODIJK, & Stelios PIPERIDIS. Portorož, Slovenia: European Language Resources Association (ELRA). ISBN: 978-2-9517408-9-1.

Joakim NIVRE, Željko AGIĆ, Lars AHRENBERG, et al. (2017a). *Universal Dependencies 2.0*. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University, Prague. URL: http://hdl.handle.net/11234/1-1983.

Joakim NIVRE, Željko AGIĆ, Lars AHRENBERG, Lene ANTONSEN, Maria Jesus ARANZABE, Masayuki ASAHARA, Luma ATEYAH, Mohammed ATTIA, Aitziber ATUTXA, Elena BADMAEVA, Miguel BALLESTEROS, Esha BANERJEE, Sebastian BANK, John BAUER, Kepa BENGOETXEA, Riyaz Ahmad BHAT, Eckhard BICK, Cristina BOSCO, Gosse BOUMA, Sam BOWMAN, Aljoscha BURCHARDT, Marie CANDITO, Gauthier CARON, Gülşen CEBIROĞLU ERYIĞIT, Giuseppe G. A. CELANO, Savas CETIN, Fabricio CHALUB, Jinho CHOI, Yongseok CHO, Silvie CINKOVÁ, Çağrı ÇÖLTEKIN, Miriam CONNOR, Marie-Catherine DE MARNEFFE, Valeria DE PAIVA, Arantza DIAZ DE ILARRAZA, Kaja DOBROVOLJC, Timothy DOZAT, Kira DROGANOVA, Marhaba ELI, Ali ELKAHKY, Tomaž ERJAVEC, Richárd FARKAS, Hector FERNANDEZ ALCALDE, Jennifer FOSTER, Cláudia FREITAS, Katarína GAJDOŠOVÁ, Daniel GALBRAITH, Marcos GARCIA, Filip GINTER, Iakes GOENAGA, Koldo GOJENOLA, Memduh GÖKRMAK, Yoav GOLDBERG, Xavier GÓMEZ GUINOVART, Berta GONZÁLES SAAVEDRA, Matias GRIONI, Normunds GRŪZĪTIS, Bruno GUILLAUME, Nizar HABASH, Jan HAJIČ, Jan HAJIČ JR., Linh HÀ MỸ, Kim HARRIS, Dag HAUG, Barbora HLADKÁ, Jaroslava HLAVÁČOVÁ, Petter HOHLE, Radu ION, Elena IRIMIA, Anders JOHANNSEN, Fredrik JØRGENSEN, Hüner KAŞKARA, Hiroshi KANAYAMA, Jenna KANERVA, Tolga KAYADELEN, Václava KETTNEROVÁ, Jesse KIRCHNER, Natalia KOTSYBA, Simon KREK, Sookyoung KWAK, Veronika LAIPPALA, Lorenzo LAMBERTINO, Tatiana LANDO, Phương LÊ HỒNG, Alessandro LENCI, Saran LERTPRADIT, Herman LEUNG, Cheuk Ying LI, Josie LI, Nikola LJUBEŠIĆ, Olga LOGINOVA, Olga LYASHEVSKAYA, Teresa LYNN, Vivien MACKETANZ, Aibek MAKAZHANOV, Michael MANDL, Christopher MANNING, Ruli MANURUNG, Cătălina MĂRĂNDUC, David MAREČEK, Katrin MARHEINECKE, Héctor MARTÍNEZ ALONSO, André MARTINS, Jan MAŠEK, Yuji MATSUMOTO, Ryan McDonald, Gustavo Mendonça, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Pinkey Nainwani, Anna Nedoluzhko, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Livy Real, Siva Reddy, Georg Rehm, Larissa Rinaldi, Laura Rituma, Rudolf Rosa, Davide Rovati, Shadi Saleh, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Antonio Stella, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jonathan North Washington, Zhuoran Yu, Zdeněk Žabokrtský, Daniel Zeman, & Hanzhi Zhu (2017b). *Universal Dependencies 2.0 – CoNLL 2017 Shared Task Development and Test Data*. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University. URL: http://hdl.handle.net/11234/1-2184.

Franz Josef Och, & Hermann Ney (2000). A comparison of alignment models for statistical machine translation. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, pp. 1086–1090.

Franz Josef Och, & Hermann Ney (2003). A systematic comparison of various statistical alignment models. In *Computational linguistics* 29.1, pp. 19–51.

Robert Östling (2015). Word Order Typology through Multilingual Word Alignment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China: Association for Computational Linguistics, pp. 205–211. URL: http://www.aclweb.org/anthology/P15-2034.

Nikolaos Pappas, & Andrei Popescu-Belis (2017). Multilingual Hierarchical Attention Networks for Document Classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Taipei, Taiwan: Asian Federation of Natural Language Processing, pp. 1015–1025. URL: http://www.aclweb.org/anthology/I17-1102.

Judea Pearl (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Nicolas Pécheux, Alexandre Allauzen, Thomas Lavergne, Guillaume Wisniewski, & François Yvon (2015). Oublier ce qu'on sait, pour mieux apprendre ce qu'on ne sait pas : une étude sur les contraintes de type dans les modèles CRF. In *Actes de la 22e conférence sur le Traitement Automatique des Langues Naturelles*. Caen, France: Association pour le Traitement Automatique des Langues, pp. 37–48. URL: http://www.atala.org/taln_archives/TALN/TALN-2015/taln-2015-long-004.

Cenel-Augusto PEREZ (2014). Linguistic Resources for Natural Language Processing. PhD thesis. Al. I. Cuza University, Iași.

Slav PETROV, Dipanjan DAS, & Ryan MCDONALD (2012). A Universal Part-of-Speech Tagset. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. Ed. by Nicoletta Calzolari (Conference CHAIR), Khalid CHOUKRI, Thierry DECLERCK, Mehmet Uğur DOĞAN, Bente MAEGAARD, Joseph MARIANI, Asuncion MORENO, Jan ODIJK, & Stelios PIPERIDIS. Istanbul, Turkey: European Language Resources Association (ELRA). ISBN: 978-2-9517408-7-7.

Slav Orlinov PETROV (2009). Coarse-to-Fine Natural Language Processing. PhD thesis. University of California, Berkeley.

Emily PITLER, & Ryan MCDONALD (2015). A Linear-Time Transition System for Crossing Interval Trees. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, pp. 662–671. URL: http://www.aclweb.org/anthology/N15-1068.

Barbara PLANK, & Gertjan VAN NOORD (2011). Effective Measures of Domain Similarity for Parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, pp. 1566–1576. URL: http://www.aclweb.org/anthology/P11-1157.

Barbara PLANK, Héctor MARTÍNEZ ALONSO, Željko AGIĆ, Danijela MERKLER, & Anders SØGAARD (2015). Do dependency parsing metrics correlate with human judgments? In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Beijing, China: Association for Computational Linguistics, pp. 315–320. URL: http://www.aclweb.org/anthology/K15-1033.

Martin POPEL, David MAREČEK, Nathan GREEN, & Zdenek ZABOKRTSKY (2011). Influence of Parser Choice on Dependency-Based MT. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Edinburgh, Scotland: Association for Computational Linguistics, pp. 433–439. URL: http://www.aclweb.org/anthology/W11-2153.

Peter PRETTENHOFER, & Benno STEIN (2010). Cross-Language Text Classification Using Structural Correspondence Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden: Association for Computational Linguistics, pp. 1118–1127. URL: http://www.aclweb.org/anthology/P10-1114.

Peter PRETTENHOFER, & Benno STEIN (2011). Cross-lingual adaptation using structural correspondence learning. In *ACM Transactions on Intelligent Systems and Technology (TIST)* 3.1, p. 13.

Ricardo BC PRUDÊNCIO, & Carlos CASTOR (2014). Cost-sensitive measures of instance hardness. In *First International Workshop on Learning over Multiple Contexts in ECML 2014*. Vol. 1.

Peng QI, & Christopher D. MANNING (2017). Arc-swift: A Novel Transition System for Dependency Parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 110–117. URL: `http://aclweb.org/anthology/P17-2018`.

Loganathan RAMASAMY, David MAREČEK, & Zdenčk ŽABOKRTSKỲ (2014). Multilingual dependency parsing: Using machine translated texts instead of parallel corpora. In *The Prague Bulletin of Mathematical Linguistics* 102.1, pp. 93–104.

Loganathan RAMASAMY, & Zdeněk ŽABOKRTSKÝ (2012). Prague Dependency Style Treebank for Tamil. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. Ed. by Nicoletta Calzolari (Conference CHAIR), Khalid CHOUKRI, Thierry DECLERCK, Mehmet Uğur DOĞAN, Bente MAEGAARD, Joseph MARIANI, Asuncion MORENO, Jan ODIJK, & Stelios PIPERIDIS. Istanbul, Turkey: European Language Resources Association (ELRA). ISBN: 978-2-9517408-7-7.

Mohammad Sadegh RASOOLI, & Michael COLLINS (2015). Density-Driven Cross-Lingual Transfer of Dependency Parsers. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 328–338. URL: `http://aclweb.org/anthology/D15-1039`.

Mohammad Sadegh RASOOLI, & Joel R. TETREAULT (2015). Yara Parser: A Fast and Accurate Dependency Parser. In *arXiv preprint arXiv:1503.06733*. URL: `http://arxiv.org/abs/1503.06733`.

Leonardo RIGUTINI, Marco MAGGINI, & Bing LIU (2005). An EM based training algorithm for cross-language text categorization. In *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on*. IEEE, pp. 529–535.

Laura RIMELL, Stephen CLARK, & Mark STEEDMAN (2009). Unbounded Dependency Recovery for Parser Evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Singapore: Association for Computational Linguistics, pp. 813–821. URL: `http://www.aclweb.org/anthology/D/D09/D09-1085`.

Annette RIOS (2015). A Basic Language Technology Toolkit for Quechua. PhD thesis. University of Zurich.

Rudolf ROSA (2015a). Multi-source Cross-lingual Delexicalized Parser Transfer: Prague or Stanford? In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*. Uppsala, Sweden: Uppsala University, Uppsala, Sweden, pp. 281–290. URL: http://www.aclweb.org/anthology/W15-2131.

Rudolf ROSA (2015b). Parsing Natural Language Sentences by Semi-supervised Methods. In *arXiv preprint arXiv:1506.04897*. URL: http://arxiv.org/abs/1506.04897.

Rudolf ROSA, & Zdenek ZABOKRTSKY (2015). KLcpos3 - a Language Similarity Measure for Delexicalized Parser Transfer. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China: Association for Computational Linguistics, pp. 243–249. URL: http://www.aclweb.org/anthology/P15-2040.

Rudolf ROSA, Jan MAŠEK, David MAREČEK, Martin POPEL, Daniel ZEMAN, & Zdeněk ŽABOKRTSKÝ (2014). HamleDT 2.0: Thirty Dependency Treebanks Stanfordized. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Ed. by Nicoletta Calzolari (Conference CHAIR), Khalid CHOUKRI, Thierry DECLERCK, Hrafn LOFTSSON, Bente MAEGAARD, Joseph MARIANI, Asuncion MORENO, Jan ODIJK, & Stelios PIPERIDIS. Reykjavik, Iceland: European Language Resources Association (ELRA). ISBN: 978-2-9517408-8-4.

Sebastian RUDER (2017). A survey of cross-lingual embedding models. In *arXiv preprint arXiv:1706.04902*. URL: http://arxiv.org/abs/1706.04902.

Kenji SAGAE, & Alon LAVIE (2006). Parser Combination by Reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*. New York City, USA: Association for Computational Linguistics, pp. 129–132. URL: http://www.aclweb.org/anthology/N/N06/N06-2033.

Adam SANTORO, Sergey BARTUNOV, Matthew BOTVINICK, Daan WIERSTRA, & Timothy P. LILLICRAP (2016). One-shot Learning with Memory-Augmented Neural Networks. In *arXiv preprint arXiv:1605.06065*. URL: http://arxiv.org/abs/1605.06065.

Francesco SARTORIO (2015). Improvements in Transition Based Systems for Dependency Parsing. PhD thesis. Università degli studi di Padova.

Yves SCHERRER, & Benoît SAGOT (2014). A Language-independent and fully Unsupervised Approach to Lexicon Induction and Part-of-Speech Tagging for Closely Related Languages. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Ed. by Nicoletta Calzolari (Conference CHAIR), Khalid CHOUKRI, Thierry DECLERCK, Hrafn LOFTSSON, Bente MAEGAARD, Joseph MARIANI, Asuncion MORENO, Jan ODIJK, & Stelios PIPERIDIS. Reykjavik, Iceland: European Language Resources Association (ELRA). ISBN: 978-2-9517408-8-4.

Michael SCHLICHTKRULL, & Anders SØGAARD (2017). Cross-Lingual Dependency Parsing with Late Decoding for Truly Low-Resource Languages. In *Proceedings of* *the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 220–229. URL: http://www.aclweb.org/anthology/E17-1021.

Roy SCHWARTZ, Omri ABEND, & Ari RAPPOPORT (2012). Learnability-Based Syntactic Annotation Design. In *Proceedings of COLING 2012*. Mumbai, India: The COLING 2012 Organizing Committee, pp. 2405–2422. URL: http://www.aclweb.org/anthology/C12-1147.

Holger SCHWENK, & Matthijs DOUZE (2017). Learning Joint Multilingual Sentence Representations with Neural Machine Translation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Vancouver, Canada: Association for Computational Linguistics, pp. 157–167. URL: http://www.aclweb.org/anthology/W17-2619.

Djamé SEDDAH, Reut TSARFATY, Sandra KÜBLER, Marie CANDITO, Jinho D. CHOI, Richárd FARKAS, Jennifer FOSTER, Iakes GOENAGA, Koldo GOJENOLA GALLETEBEITIA, Yoav GOLDBERG, Spence GREEN, Nizar HABASH, Marco KUHLMANN, Wolfgang MAIER, Joakim NIVRE, Adam PRZEPIÓRKOWSKI, Ryan ROTH, Wolfgang SEEKER, Yannick VERSLEY, Veronika VINCZE, Marcin WOLIŃSKI, Alina WRÓBLEWSKA, & Eric Villemonte DE LA CLERGERIE (2013). Overview of the SPMRL 2013 Shared Task: A Cross-Framework Evaluation of Parsing Morphologically Rich Languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*. Seattle, Washington, USA: Association for Computational Linguistics, pp. 146–182. URL: http://www.aclweb.org/anthology/W13-4917.

Yu SHEN, Chenhui CHU, Fabien CROMIERES, & Sadao KUROHASHI (2016). Cross-language Projection of Dependency Trees with Constrained Partial Parsing for Tree-to-Tree Machine Translation. In *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, pp. 1–11. URL: http://www.aclweb.org/anthology/W16-2201.

Masayoshi SHIBATANI (1990). *The languages of Japan*. Cambridge University Press.

Hidetoshi SHIMODAIRA (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. In *Journal of statistical planning and inference* 90.2, pp. 227–244.

Michael R. SMITH, Tony MARTINEZ, & Christophe GIRAUD-CARRIER (2014). An instance level analysis of data complexity. In *Machine learning* 95.2, pp. 225–256.

Anders SØGAARD (2011). Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, pp. 682–686. URL: http://www.aclweb.org/anthology/P11-2120.

Kathrin SPREYER, & Jonas KUHN (2009). Data-Driven Dependency Parsing of New Languages Using Incomplete and Noisy Training Data. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*. Boulder, Colorado: Association for Computational Linguistics, pp. 12–20. URL: http://www.aclweb.org/anthology/W09-1104.

Kathrin SPREYER, Lilja ØVRELID, & Jonas KUHN (2010). Training Parsers on Partial Trees: A Cross-language Comparison. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. Ed. by Nicoletta Calzolari (Conference CHAIR), Khalid CHOUKRI, Bente MAEGAARD, Joseph MARIANI, Jan ODIJK, Stelios PIPERIDIS, Mike ROSNER, & Daniel TAPIAS. Valletta, Malta: European Language Resources Association (ELRA). ISBN: 2-9517408-6-7.

Milan STRAKA (2017). *CoNLL 2017 Shared Task - UDPipe Baseline Models and Supplementary Materials*. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University. URL: http://hdl.handle.net/11234/1-1990.

Milan STRAKA, & Jana STRAKOVÁ (2017). Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Vancouver, Canada: Association for Computational Linguistics, pp. 88–99. URL: http://www.aclweb.org/anthology/K17-3009.

Milan STRAKA, Jan HAJIČ, Jana STRAKOVÁ, & Jan HAJIČ JR. (2015). Parsing Universal Dependency Treebanks Using Neural Networks and Search-Based Oracle. In *International Workshop on Treebanks and Linguistic Theories (TLT14)*, pp. 208–220.

Mihai SURDEANU, & Christopher D. MANNING (2010). Ensemble Models for Dependency Parsing: Cheap and Good? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California: Association for Computational Linguistics, pp. 649–652. URL: http://www.aclweb.org/anthology/N10-1091.

Raymond Hendy SUSANTO, & Wei LU (2017). Neural Architectures for Multilingual Semantic Parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 38–44. URL: http://aclweb.org/anthology/P17-2007.

Anders SØGAARD, & Julie WULFF (2012). An Empirical Study of Non-Lexical Extensions to Delexicalized Transfer. In *Proceedings of COLING 2012: Posters*. Mumbai, India: The COLING 2012 Organizing Committee, pp. 1181–1190. URL: http://www.aclweb.org/anthology/C12-2115.

Oscar TÄCKSTRÖM (2012). Nudging the Envelope of Direct Transfer Methods for Multilingual Named Entity Recognition. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*. Montréal, Canada: Association for Computational Linguistics, pp. 55–63. URL: http://www.aclweb.org/anthology/W12-1908.

Oscar TÄCKSTRÖM, Ryan MCDONALD, & Jakob USZKOREIT (2012). Cross-lingual Word Clusters for Direct Transfer of Linguistic Structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Montréal, Canada: Association for Computational Linguistics, pp. 477–487. URL: http://www.aclweb.org/anthology/N12-1052.

Oscar TÄCKSTRÖM, Ryan MCDONALD, & Joakim NIVRE (2013). Target Language Adaptation of Discriminative Transfer Parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, pp. 1061–1071. URL: http://www.aclweb.org/anthology/N13-1126.

Oscar TÄCKSTRÖM, Dipanjan DAS, Slav PETROV, Ryan MCDONALD, & Joakim NIVRE (2013). Token and Type Constraints for Cross-Lingual Part-of-Speech Tagging. In *Transactions of the Association for Computational Linguistics* 1, pp. 1–12.

David TEMPERLEY (2007). Minimization of dependency length in written English. In *Cognition* 105.2, pp. 300–333.

Jörg TIEDEMANN (2014). Rediscovering Annotation Projection for Cross-Lingual Parser Induction. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, pp. 1854–1864. URL: http://www.aclweb.org/anthology/C14-1175.

Jörg TIEDEMANN (2015a). Cross-Lingual Dependency Parsing with Universal Dependencies and Predicted PoS Labels. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*. Uppsala, Sweden: Uppsala University, Uppsala, Sweden, pp. 340–349. URL: http://www.aclweb.org/anthology/W15-2137.

Jörg TIEDEMANN (2015b). Improving the Cross-Lingual Projection of Syntactic Dependencies. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*. Vilnius, Lithuania: Linköping University Electronic Press, Sweden, pp. 191–199. URL: http://www.aclweb.org/anthology/W15-1824.

Jörg TIEDEMANN, & Željko AGIĆ (2016). Synthetic Treebanking for Cross-Lingual Dependency Parsing. In *Journal of Artificial Intelligence Research* 55, pp. 209–248.

Jörg TIEDEMANN, & Lonneke VAN DER PLAS (2016). *Bootstrapping a dependency parser for Maltese – a real-world test case*.

Jörg TIEDEMANN, Željko AGIĆ, & Joakim NIVRE (2014). Treebank Translation for Cross-Lingual Parser Induction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*. Ann Arbor, Michigan: Association for Computational Linguistics, pp. 130–140. URL: http://www.aclweb.org/anthology/W14-1614.

Jörg TIEDEMANN (2012). Parallel Data, Tools and Interfaces in OPUS. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*.

Ed. by Nicoletta Calzolari (Conference CHAIR), Khalid CHOUKRI, Thierry DECLERCK, Mehmet Uğur DOĞAN, Bente MAEGAARD, Joseph MARIANI, Asuncion MORENO, Jan ODIJK, & Stelios PIPERIDIS. Istanbul, Turkey: European Language Resources Association (ELRA). ISBN: 978-2-9517408-7-7.

Reut TSARFATY (2013). A Unified Morpho-Syntactic Scheme of Stanford Dependencies. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 578–584. URL: http://www.aclweb.org/anthology/P13-2103.

Yulia TSVETKOV (2015). Cross-Lingual Transfer of Linguistic and Metalinguistic Knowledge via Lexical Borrowing. Thesis Proposal. Georgia Institute of Technology.

Francis M. TYERS, & Jacques A. PIENAAR (2008). Extracting bilingual word pairs from Wikipedia. In *Collaboration: interoperability between people in the creation of language resources for less-resourced languages* 19, pp. 19–22.

Shyam UPADHYAY, Manaal FARUQUI, Chris DYER, & Dan ROTH (2016). Cross-lingual Models of Word Embeddings: An Empirical Comparison. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 1661–1670. URL: http://www.aclweb.org/anthology/P16-1157.

Stephan VOGEL, Hermann NEY, & Christoph TILLMANN (1996). HMM-Based Word Alignment in Statistical Translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, pp. 836–841.

Alexander VOLOKH (2013). Performance-oriented dependency parsing. PhD thesis.

Alexander VOLOKH, & Günter NEUMANN (2012). Task-oriented dependency parsing evaluation methodology. In *Information Reuse and Integration (IRI), 2012 IEEE 13th International Conference on*. IEEE, pp. 132–137.

Xiaojun WAN (2009). Co-Training for Cross-Lingual Sentiment Classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Association for Computational Linguistics, pp. 235–243. URL: http://www.aclweb.org/anthology/P/P09/P09-1027.

Chuan WANG, Nianwen XUE, & Sameer PRADHAN (2015). A Transition-based Algorithm for AMR Parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, pp. 366–375. URL: http://www.aclweb.org/anthology/N15-1040.

Dingquan WANG, & Jason EISNER (2017). Fine-Grained Prediction of Syntactic Typology: Discovering Latent Structure with Supervised Learning. In *Transactions of the Association for Computational Linguistics* 5, pp. 147–161. ISSN: 2307-387X. URL: https://transacl.org/ojs/index.php/tacl/article/view/1060.

Haifeng WANG, Hua WU, & Zhanyi LIU (2006). Word Alignment for Languages with Scarce Resources Using Bilingual Corpora of Other Language Pairs. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*. Sydney, Australia: Association for Computational Linguistics, pp. 874–881. URL: http://www.aclweb.org/anthology/P/P06/P06-2112.

Mengqiu WANG, & Christopher D. MANNING (2014). Cross-lingual Projected Expectation Regularization for Weakly Supervised Learning. In *Transactions of the Association of Computational Linguistics* 2.1, pp. 55–66.

Mengqiu WANG, Wanxiang CHE, & Christopher D. MANNING (2013). Effective Bilingual Constraints for Semi-Supervised Learning of Named Entity Recognizers. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Bin WEI, & Christopher PAL (2010). Cross Lingual Adaptation: An Experiment on Sentiment Classifications. In *Proceedings of the ACL 2010 Conference Short Papers*. Uppsala, Sweden: Association for Computational Linguistics, pp. 258–262. URL: http://www.aclweb.org/anthology/P10-2048.

David WEISS, & Benjamin TASKAR (2010). Structured prediction cascades. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 916–923.

Guillaume WISNIEWSKI, & Ophélie LACROIX (2017). A Systematic Comparison of Syntactic Representations of Dependency Parsing. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*. Gothenburg, Sweden: Association for Computational Linguistics, pp. 146–152. URL: http://www.aclweb.org/anthology/W17-0419.

Guillaume WISNIEWSKI, Nicolas PÉCHEUX, Souhir GAHBICHE-BRAHAM, & François YVON (2014). Cross-Lingual Part-of-Speech Tagging through Ambiguous Learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1779–1785. URL: http://www.aclweb.org/anthology/D14-1187.

Alina WRÓBLEWSKA, & Adam PRZEPIÓRKOWSKI (2014). Towards a Weighted Induction Method of Dependency Annotation. In *International Conference on Natural Language Processing*. Springer, pp. 164–176.

Hua WU, Haifeng WANG, & Zhanyi LIU (2005). Alignment Model Adaptation for Domain-Specific Word Alignment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Ann Arbor, Michigan: Association for Computational Linguistics, pp. 467–474. DOI: 10.3115/1219840.1219898. URL: http://www.aclweb.org/anthology/P05-1058.

Chenhai XI, & Rebecca HWA (2005). A Backoff Model for Bootstrapping Resources for Non-English Languages. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Vancouver, British Columbia, Canada: Association for Computational Linguistics, pp. 851–858. URL: http://www.aclweb.org/anthology/H/H05/H05-1107.

Fei XIA, & William LEWIS (2007). Multilingual Structural Projection across Interlinear Text. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. Rochester, New York: Association for Computational Linguistics, pp. 452–459. URL: http://www.aclweb.org/anthology/N/N07/N07-1057.

Min XIAO, & Yuhong GUO (2014). Distributed Word Representation Learning for Cross-Lingual Dependency Parsing. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*. Ann Arbor, Michigan: Association for Computational Linguistics, pp. 119–129. URL: http://www.aclweb.org/anthology/W14-1613.

Kenji YAMADA, & Kevin KNIGHT (2001). A Syntax-based Statistical Translation Model. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*. Toulouse, France: Association for Computational Linguistics, pp. 523–530. DOI: 10.3115/1073012.1073079. URL: http://www.aclweb.org/anthology/P01-1067.

David YAROWSKY, & Grace NGAI (2001). Inducing multilingual POS taggers and NP brackets via robust projection across aligned corpora. In *Proceedings of NAACL*.

David YAROWSKY, Grace NGAI, & Richard WICENTOWSKI (2001). Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research*. Association for Computational Linguistics, pp. 1–8.

Zhiwei YU, David MAREČEK, Zdeněk ŽABOKRTSKÝ, & Daniel ZEMAN (2016). If You Even Don't Have a Bit of Bible: Learning Delexicalized POS Taggers. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Ed. by Nicoletta Calzolari (Conference CHAIR), Khalid CHOUKRI, Thierry DECLERCK, Sara GOGGI, Marko GROBELNIK, Bente MAEGAARD, Joseph MARIANI, Helene MAZO, Asuncion MORENO, Jan ODIJK, & Stelios PIPERIDIS. Portorož, Slovenia: European Language Resources Association (ELRA). ISBN: 978-2-9517408-9-1.

Deniz YURET, Aydin HAN, & Zehra TURGUT (2010). SemEval-2010 Task 12: Parser Evaluation Using Textual Entailments. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Uppsala, Sweden: Association for Computational Linguistics, pp. 51–56. URL: http://www.aclweb.org/anthology/S10-1009.

Daniel ZEMAN, & Philip RESNIK (2008). Cross-Language Parser Adaptation between Related Languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*, pp. 35–42.

Daniel ZEMAN, David MAREČEK, Zhiwei YU, & Zdeněk ŽABOKRTSKÝ (2016). Planting Trees in the Desert: Delexicalized Tagging and Parsing Combined. In *Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation*. Seoul, Korea: Kyung Hee University, pp. 199–207. ISBN: 978-89-6817-428-5.

Daniel ZEMAN, Martin POPEL, Milan STRAKA, Jan HAJIC, Joakim NIVRE, Filip GINTER, Juhani LUOTOLAHTI, Sampo PYYSALO, Slav PETROV, Martin POTTHAST, Francis TYERS, Elena BADMAEVA, Memduh GOKIRMAK, Anna NEDOLUZHKO, Silvie CINKOVA, Jan HAJIC JR., Jaroslava HLAVACOVA, Václava KETTNEROVÁ, Zdenka URESOVA, Jenna KANERVA, Stina OJALA, Anna MISSILÄ, Christopher D. MANNING, Sebastian SCHUSTER, Siva REDDY, Dima TAJI, Nizar HABASH, Herman LEUNG, Marie-Catherine DE MARNEFFE, Manuela SANGUINETTI, Maria SIMI, Hiroshi KANAYAMA, Valeria DEPAIVA, Kira DROGANOVA, Héctor MARTÍNEZ ALONSO, Çağrı ÇÖLTEKIN, Umut SULUBACAK, Hans USZKOREIT, Vivien MACKETANZ, Aljoscha BURCHARDT, Kim HARRIS, Katrin MARHEINECKE, Georg REHM, Tolga KAYADELEN, Mohammed ATTIA, Ali ELKAHKY, Zhuoran YU, Emily PITLER, Saran LERTPRADIT, Michael MANDL, Jesse KIRCHNER, Hector Fernandez ALCALDE, Jana STRNADOVÁ, Esha BANERJEE, Ruli MANURUNG, Antonio STELLA, Atsuko SHIMADA, Sookyoung KWAK, Gustavo MENDONCA, Tatiana LANDO, Rattima NITISAROJ, & Josie LI (2017). CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Vancouver, Canada: Association for Computational Linguistics, pp. 1–19. URL: http://www.aclweb.org/anthology/K17-3001.

Yuan ZHANG, & Regina BARZILAY (2015). Hierarchical Low-Rank Tensors for Multilingual Transfer Parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1857–1867. URL: http://aclweb.org/anthology/D15-1213.

Yuan ZHANG, Roi REICHART, Regina BARZILAY, & Amir GLOBERSON (2012). Learning to Map into a Universal POS Tagset. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Jeju Island, Korea: Association for Computational Linguistics, pp. 1368–1378. URL: http://www.aclweb.org/anthology/D12-1125.

Yuan ZHANG, David GADDY, Regina BARZILAY, & Tommi JAAKKOLA (2016). Ten Pairs to Tag – Multilingual POS Tagging via Coarse Mapping between Embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, pp. 1307–1317. URL: http://www.aclweb.org/anthology/N16-1156.

Yue ZHANG, & Stephen CLARK (2008). A Tale of Two Parsers: Investigating and Combining Graph-based and Transition-based Dependency Parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Honolulu, Hawaii: Association for Computational Linguistics, pp. 562–571. URL: http://www.aclweb.org/anthology/D08-1059.

Yue ZHANG, & Joakim NIVRE (2011). Transition-based Dependency Parsing with Rich Non-local Features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, pp. 188–193. URL: http://www.aclweb.org/anthology/P11-2033.

Yue ZHANG, & Joakim NIVRE (2012). Analyzing the Effect of Global Learning and Beam-Search on Transition-Based Dependency Parsing. In *Proceedings of COLING 2012: Posters*. Mumbai, India: The COLING 2012 Organizing Committee, pp. 1391–1400. URL: http://www.aclweb.org/anthology/C12-2136.

Hai ZHAO, Yan SONG, Chunyu KIT, & Guodong ZHOU (2009). Cross Language Dependency Parsing using a Bilingual Lexicon. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Association for Computational Linguistics, pp. 55–63. URL: http://www.aclweb.org/anthology/P/P09/P09-1007.

Julian ZUBEK, & Dariusz M. PLEWCZYNSKI (2016). Complexity curve: a graphical measure of data complexity and classifier performance. In *PeerJ Computer Science* 2, e76.

Titre : Apprentissage d'analyseurs syntaxiques pour les langues peu dotées : amélioration du transfert cross-lingue grâce à des connaissances monolingues

Mots-clefs : Traitement automatique des langues, langues peu dotées, transfert cross-lingue, connaissances linguistiques, analyse en dépendance, prédiction structurée

Résumé : Le récent essor des algorithmes d'apprentissage automatique a rendu les méthodes de Traitement Automatique des Langues d'autant plus sensibles à leur facteur le plus limitant : la qualité des systèmes repose entièrement sur la disponibilité de grandes quantités de données, ce qui n'est pourtant le cas que d'une minorité parmi les 7.000 langues existant au monde.
La stratégie dite du transfert cross-lingue permet de contourner cette limitation : une langue peu dotée en ressources (la cible) peut être traitée en exploitant les ressources disponibles dans une autre langue (la source). Les progrès accomplis sur ce plan se limitent néanmoins à des scénarios idéalisés, avec des ressources cross-lingues prédéfinies et de bonne qualité, de sorte que le transfert reste inapplicable aux cas réels de langues peu dotées, qui n'ont pas ces garanties.
Cette thèse vise donc à tirer parti d'une multitude de sources et ressources cross-lingues, en opérant une combinaison sélective : il s'agit d'évaluer, pour chaque aspect du traitement cible, la pertinence de chaque ressource. L'étude est menée en utilisant l'analyse en dépendance par transition comme cadre applicatif.
Le cœur de ce travail est l'élaboration d'un nouveau méta-algorithme de transfert, dont l'architecture en cascade permet la combinaison fine des diverses ressources, en ciblant leur exploitation à l'échelle du mot. L'approche cross-lingue pure n'étant en l'état pas compétitive avec la simple annotation de quelques phrases cibles, c'est avant tout la complémentarité de ces méthodes que souligne l'analyse empirique. Une série de nouvelles métriques permet une caractérisation fine des similarités cross-lingues et des spécificités syntaxiques de chaque langue, de même que de la valeur ajoutée de l'information cross-lingue par rapport au cadre monolingue. L'exploitation d'informations typologiques s'avère également particulièrement fructueuse.
Ces contributions reposent largement sur des innovations techniques en analyse syntaxique, concrétisées par la publication en *open source* du logiciel PanParser, qui exploite et généralise la méthode dite des oracles dynamiques. Cette thèse contribue sur le plan monolingue à plusieurs autres égards, comme le concept de cascades monolingues, pouvant traiter par exemple d'abord toutes les dépendances faciles, puis seulement les difficiles.

Title: Training parsers for low-resourced languages: improving cross-lingual transfer with monolingual knowledge

Keywords: Natural language processing, low-resourced languages, cross-lingual transfer, linguistic knowledge, dependency parsing, structured prediction

Abstract: As a result of the recent blossoming of Machine Learning techniques, the Natural Language Processing field faces an increasingly thorny bottleneck: the most efficient algorithms entirely rely on the availability of large training data. These technological advances remain consequently unavailable for the 7,000 languages in the world, out of which most are low-resourced.
One way to bypass this limitation is the approach of cross-lingual transfer, whereby resources available in another (source) language are leveraged to help building accurate systems in the desired (target) language. However, despite promising results in research settings, the standard transfer techniques lack the flexibility regarding cross-lingual resources needed to be fully usable in real-world scenarios: exploiting very sparse resources, or assorted arrays of resources. This limitation strongly diminishes the applicability of that approach.
This thesis consequently proposes to combine multiple sources and resources for transfer, with an emphasis on selectivity: can we estimate which resource of which language is useful for which input? This strategy is put into practice in the frame of transition-based dependency parsing.
To this end, a new transfer framework is designed, with a cascading architecture: it enables the desired combination, while ensuring better targeted exploitation of each resource, down to the level of the word. Empirical evaluation dampens indeed the enthusiasm for the purely cross-lingual approach – it remains in general preferable to annotate just a few target sentences – but also highlights its complementarity with other approaches. Several metrics are developed to characterize precisely cross-lingual similarities, syntactic idiosyncrasies, and the added value of cross-lingual information compared to monolingual training. The substantial benefits of typological knowledge are also explored.
The whole study relies on a series of technical improvements regarding the parsing framework: this work includes the release of a new open source software, PanParser, which revisits the so-called dynamic oracles to extend their use cases. Several purely monolingual contributions complete this work, including an exploration of monolingual cascading, which offers promising perspectives with easy-then-hard strategies.