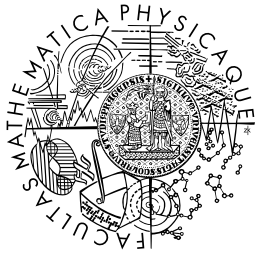

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

Rudolf Rosa

Discovering the structure of natural language sentences by semi-supervised methods

Institute of Formal and Applied Linguistics

Supervisor of the doctoral thesis:	doc. Ing. Zdeněk Žabokrtský, Ph.D.
Study programme:	Informatics
Study branch:	Mathematical Linguistics

Beroun 2018

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ..... signature of the author

Title: Discovering the structure of natural language sentences by semi-supervised methods

Author: Rudolf Rosa

Institute: Institute of Formal and Applied Linguistics

Supervisor: doc. Ing. Zdeněk Žabokrtský, Ph.D., Institute of Formal and Applied Linguistics

Abstract: In this thesis, we focus on the problem of automatically syntactically analyzing a language for which there is no syntactically annotated training data. We explore several methods for cross-lingual transfer of syntactic as well as morphological annotation, ultimately based on utilization of bilingual or multilingual sentence-aligned corpora and machine translation approaches. We pay particular attention to automatic estimation of the appropriateness of a source language for the analysis of a given target language, devising a novel measure based on the similarity of part-of-speech sequences frequent in the languages. The effectiveness of the presented methods has been confirmed by experiments conducted both by us as well as independently by other respectable researchers.

Keywords: dependency parsing, part-of-speech tagging, cross-lingual processing, multilingual processing

I dedicate this thesis to the Invisible Pink Unicorn goddess.

But really, I would like to thank religion, especially the Christian one, because no other force in the world has been capable of producing such massively multiparallel texts as the Christians with the Bible. And the Watchtower texts produced by Jehovah's Witnesses are just great (at least for multilingual natural language processing).

Cheers to the HamleDT group and the Universal Dependencies community for creating such great resources, without which this work would be totally impossible.

Big thanks to Milan Straka and his team for developing a range of great NLP tools, especially the Parsito parser and the UDPipe toolkit, with high-quality documentation as well as a fast-responding support.

Thanks to my dear colleagues with whom I had the pleasure to cooperate on my research, especially David Mareček, Dan Zeman, and Zdeněk Žabokrtský.

And thanks to my beloved ÚFAL, the Institute of Formal and Applied Linguistics, for being such a friendly place, for being so flexible about everything, and for allowing me to devote most of my time to this thesis in the last months, which ultimately made it possible for me to finish this work.

Contents

Introduction

The topic of this thesis is automatic linguistic analysis of written text, specifically syntactic dependency parsing, and, to some extent, morphological part-of-speech tagging.

Syntactic parsing is a classical Natural Language Processing (NLP) task, which often serves as a gateway for further and deeper language understanding. Part of Speech (POS) tagging has many uses, but we are only interested in it as a preprocessing step for parsing in this thesis.

In the classical fully supervised monolingual data-driven parsing (Chapter 2), a parser is trained on a syntactically annotated corpus, i.e. a treebank. To achieve a reasonable parsing accuracy, the treebank should contain thousands or tens of thousands of manually annotated sentences. However, such treebanks are expensive to create, and are thus available only for a few dozen languages; currently, less than 80 languages have at least a tiny treebank available. This renders approximately 99% of the world's languages *under-resourced* in terms of parsing resources, as the classical fully supervised approach to parsing cannot be applied for these languages.

This situation constitutes the main motivation for our work. While treebanks are not available for those languages, there is a belief that most or all languages in the world are similar to each other to some extent. Therefore, annotated resources for resource-rich languages might be utilized to learn knowledge useful for analyzing other languages, especially similar ones. Moreover, even if no treebank is available for a language, we might still exploit other resources, such as parallel or even monolingual plain-text data. We discuss the datasets potentially useful for parsing in Chapter 1.

One possible approach to use is the cross-lingual transfer of a delexicalized parser, which we introduce in Chapter 3. Here, the idea is that even if two languages differ in their lexicon, they might not differ that much in their grammar. Therefore, a parser trained without any lexical features on a treebank for a resource-rich *source* language (i.e. a delexicalized parser) might be applicable to a resource-poor *target* language. As the delexicalized parser transfer approach has been repeatedly shown to perform well, we take it as the basis for our research, and extend it in several ways.

There is already a wide range of resource-rich languages, which can be used as source languages in the parser transfer. However, automatically choosing the optimal source language is an important yet non-trivial task. Moreover, a clever combination of multiple sources might be an even better approach to take. We address both of these issues in Chapter 4, where we introduce our language similarity measure, $KL^{-4}_{cpos^3}$, and we port a monolingual multi-source parser combination method into the cross-lingual setting. This is the key chapter of this thesis.

By delexicalizing the parser, we are losing some accuracy. Fortunately, existing parallel text corpora can be utilized to lexicalize the cross-lingual parsing, either directly through word alignment links, or indirectly via Machine Translation (MT). In Chapter 5, we take the latter approach, investigating the potential of simpler word-based MT approaches and their advantages over state-of-the-art

phrase-based MT systems.

A problem we have been leaving unaddressed so far is the fact that we typically need to provide the parsers with a morphological annotation of the input sentences. Delexicalized parsers actually operate primarily on POS tags; and even for lexicalized parsers, POS tags constitute a very useful input feature. However, as we cannot reasonably assume to have supervised POS taggers available for all under-resourced target languages, we need to apply cross-lingual approaches even for tagging, which we investigate in Chapter 6.

We conclude the thesis by summarizing our findings in the form of step-by-step instructions for parsing an under-resourced language.

1. Datasets for Parsing

In this chapter, we deal with the data that we use in cross-lingual parsing.

The key resource for data-driven dependency parsing are dependency treebanks, i.e. corpora of sentences annotated with syntactic trees, which we review in Section 1.1. In the case of resource-rich target languages for which large treebanks are available, the task of parsing then consists of training an off-the-shelf parser on the treebank and applying it to the texts in the target language.

However, in our scenario, we assume the target languages to be resource-poor, with no annotated data available. Our approaches are thus based on exploitation of treebanks for different source languages, and transfer of the knowledge learned from those treebanks into the target languages. Unfortunately, treebanks tend to use a wide range of different styles of both morphological and syntactic annotation, which poses significant problems to any cross-lingual processing – we need the treebanks to be *harmonized*, i.e. to be annotated in an as much similar way as possible. While the harmonization of syntactic annotation is obviously crucial for cross-lingual parsing, we need the morphological annotation to be harmonized as well, as it constitutes a very important input feature for parsing (this is especially true for the POS tags). We deal with treebank harmonization in Section 1.2, and explicitly list the treebank datasets or their subsets that we use in our experiments in Section 1.3.

Another very important resource for any cross-lingual processing are parallel corpora, i.e. texts in one (source) language accompanied by their human-devised translations in another (target) language. These enable us to transfer annotation or knowledge from the source language into the target language, typically either by means of projection over word alignment on the parallel data, or by training an MT system on the parallel data. Fortunately, parallel data are "natural resources", available in the wild for harvesting and subsequent construction of parallel corpora. We discuss parallel corpora in Section 1.4, with a focus on parallel data that are typically available for under-resourced languages.

Other resources can also be useful for cross-lingual parsing; in the low-resource scenario, we would ideally like to utilize any resources that are available for the target language. We discuss exploitation of other resources in Section 1.5, with a focus on monolingual texts and linguistic catalogues.

We note that at least a small amount of monolingual text is an unavoidable requirement for the target language, as without any text to parse, the task of parsing has no sense for the language.

1.1 Syntactically annotated corpora

This section partially draws from [Jurafsky and Martin, 2017].

The two largest and most common classes of syntactic trees are phrase structure trees (also called constituency trees) and dependency trees. Both typically follow the treeness constraints, with each child node having exactly one parent node, except for the unique root node which has no parent, and with the tree being cycle-free, i.e. with no node being its own transitive descendant. Typically, the left-to-right ordering of the nodes locally or globally corresponds to the word

order in the underlying sentence.

Phrase structure trees [Chomsky, 1956] capture the composition of a sentence from larger phrases, which themselves are composed of smaller phrases, and so on, until getting to single words. Phrase structure trees contain two types of nodes – terminal (leaf) nodes correspond to words, and non-terminal (non-leaf) nodes correspond to phrases and bear phrase labels (e.g. *NP* for a nominal phrase, or *PP* for a prepositional phrase).

Dependency trees [Tesnière, 1959, 2015] capture syntactic relations between individual words. Typically, they only contain one type of nodes, which correspond to words (although additional nodes are added in some theories). The label of the relation between a head (parent, governor) node and a dependent (child, modifier) node typically corresponds to the function or role of the dependent node in relation to its parent node (e.g. a nominal node can be the *subject* to its verbal parent node, or an adjectival node can be an *adjectival modifier* to its nominal parent node).

A wide range of syntactic theories operate within these two classes of trees, because there are many linguistically sensible ways of representing the syntactic structure of a sentence by a tree – both in terms of the structure of the tree, and in terms of the labels of the phrases or the dependency relations.

Since the Penn Treebank (PennTB) of Marcus et al. [1993], the syntactic theories have been realized by annotating treebanks. In our work, we do not deal with the theories explicitly – as we use data-driven dependency parsers, they can be trained on any treebank annotated with standard dependency trees, regardless of the underlying theory. However, because of the multilingual setting of our research, we need all of the treebanks that we use to adhere to the same theory; or, rather, *annotation style*, as the practical treebank annotation may diverge to some extent from the underlying theory.

Authors of a treebank typically devise their own annotation style, defining the dependency structures and dependency relation labels, as well as the POSes and other morphological features, and possibly additional annotations. The annotation style is typically designed to suit well the language, but it is also inevitably influenced by the linguistic schools and traditions which the authors adhere to. Moreover, it usually incorporates many less systematic decisions that emerge during the annotation process to handle phenomena that were not covered by the initial annotation guidelines. It is also often at least partially fitted to the intended use of the dataset. It can be based on a pre-existing annotation style, such as that of the PennTB, but even in such cases, it is still usually adapted for the new treebank at least a bit. Thus, the original treebank annotations are extremely varied, both cross-lingually as well as even intra-lingually.

1.1.1 The Penn treebanks family

The first non-trivial treebank was the Penn Treebank (PennTB) of Marcus et al. [1993], created at the University of Pennsylvania. It is a treebank of English sentences annotated with POS tags and phrase structure trees. The creation of the treebank made it possible to easily automatically evaluate as well train automatic syntactic parsers.

The researchers at Penn University then continued by annotating constituency

treebanks for other languages – the Penn Korean Treebank [Han et al., 2002], the Penn Arabic Treebank [Maamouri et al., 2004], and the Penn Chinese Treebank [Xue et al., 2005]. While these in principle followed the same annotation style as the PennTB, specific features of the respective languages needed to be dealt with, leading to language-specific diversions for each of the treebanks. Of course, PennTB also inspired other treebanks, created by researchers at other institutes, which typically differ in the annotation style even more.

Due to the prominent position of the PennTB, many parsers, as well as other automated NLP tools, were directly tied to its annotation style, requiring the PennTB POS tags on the input and producing the PennTB-style parse trees on the output. Therefore, to process other languages than those covered by the Penn treebanks, it was quite common to use automatic conversions to obtain PennTB-style annotations even for other languages [Xia and Palmer, 2001, Collins, 2003]. We consider these to be the first attempts at treebank annotation harmonization.

Later on, as the dependency paradigm was becoming more and more popular for syntactic parsing, the usual direction of the conversions changed to converting PennTB-style treebanks into dependency representations.

1.1.2 The Prague treebanks family

The first *dependency* treebank was the Prague Dependency Treebank (PDT) [Hajič, 1998], created at the Charles University in Prague. It is a treebank of Czech sentences, with an annotation style inspired by the Functional Generative Description of Sgall [1967]. PDT is annotated on multiple layers; however, for our work, it is sufficient to limit ourselves to the layer of words (w-layer), morphology (m-layer), and surface syntax (a-layer).

The Prague treebanking group then continued by producing the Prague Czech-English Dependency Treebank [Čmejrek et al., 2004, Cuřín et al., 2004], the Prague Arabic Dependency Treebank [Hajič et al., 2004a,b], the Prague English Dependency Treebank [Hajič et al., 2009], and the Prague Tamil Dependency Treebank [Ramasamy and Žabokrtský, 2012].

Naturally, the annotation style used for creating these resources was mostly based on the PDT annotation style (sometimes also referred to as Prague Dependencies); mostly, it was only enriched with new labels and guidelines for phenomena non-existent in Czech (such as determiners). While the resulting treebanks were not yet fully harmonized in the modern sense, this gave the group the necessary experience and ground to start the HamleDT project, which will be introduced in Section 1.2.3.

Similarly the PennTB, PDT also inspired the creation of other treebanks – while the PennTB was the model for constituency treebanks and treebanking in general, PDT became a model for many dependency treebanks.

1.1.3 CoNLL treebanks

In the year 2005, there were already dependency or constituency treebanks available for a number of languages, even though their annotation styles varied considerably, as did the formats in which they were stored and distributed.

Moreover, two new trainable data-driven dependency parsers had been developed – the MaltParser of Nivre [2003], and the MSTParser of McDonald et al. [2005a].

This set the ground for two CoNLL shared tasks on multilingual dependency parsing [Buchholz and Marsi, 2006, Nivre et al., 2007], with multilingual parsing understood as training and evaluating a parser for more than one language, contrary to previous works which typically only focused on parsing one or two languages. Many of the existing treebanks were gathered for the task (13 for CoNLL 2006 and 10 for CoNLL 2007, with a partial overlap), both dependency based and constituency based, which were converted into dependencies in a semi-automatic way.

While the CoNLL treebank collections were an important milestone in multilingual dependency parsing, their annotations were not yet harmonized in the modern sense. The morphological annotation underwent a sort of rudimentary normalization, including both the original fine-grained annotations as well as simplified coarse-grained POS tags (CPOSTAG), but the particular values used were still taken over from the original treebanks, and the syntactic annotation (dependency structure and relation labels) was also kept from the original. The crucial novelty at that time was the fact that all of the treebanks were converted into the same simple text-based file format with a set of predefined fields, usually referred to as the CoNLL or CoNLL-X format. Moreover, many of the converted treebanks were now rather easily obtainable. This made it viable to train and evaluate monolingual parsers on multiple treebanks, bringing multilinguality (although not yet cross-linguality) into parsing.

When early attempts at cross-lingual parsing were made, the CoNLL collections were typically used, even though their limitations were clear, since there was no better resource in existence (yet). The annotation style differences had a clear impact on performance; even if the coarse POS tags were harmonized, it was often observed that typological similarity of the source and target languages used in cross-lingual parsing was of less importance than the similarity of the annotation styles [McDonald et al., 2011].

1.2 Treebank harmonization

Using a different annotation style for each language might make perfect sense monolingually. However, it is clear that the different annotation styles constitute a major obstacle to any cross-lingual or multilingual processing. Therefore, *treebank harmonization* emerged as a general approach of converting the original heterogeneous annotations into a common style, using a similar set of POS tags, dependency relation labels, syntactic structures, and other annotations, across multiple languages.

If we were to analyze only one target language with no annotation available by transferring the knowledge from annotated resources for only one source language, the harmonization might not be necessary; although, even in that case, the harmonization might help by avoiding annotations specific for the source language only, such as parts of speech or dependency relation labels that do not appear in the target language.

However, in practice, we typically work with multiple source and target languages, and the cross-lingual methods can therefore benefit from the harmonized annotations, allowing us to combine multiple sources together, as well as to use a more unified approach and tools across all of the languages.

We also typically operate in a simulated low-resource setting, applying our methods to target languages for which at least small treebanks are available. This is because for truly under-resourced languages, the cross-lingual methods can be applied, but cannot be automatically evaluated. Thus, for the automatic evaluation to be possible, we require even the target treebanks to be harmonized.

Moreover, in scenarios where we assume the availability of target POS tags, these obviously need to be harmonized, as they constitute the input for the parser.

1.2.1 The beginnings

The need for annotation harmonization in cross-lingual parsing has been clear since the founding work of Hwa et al. [2005], who projected annotations from English into Chinese and Spanish over parallel data.

The authors used the English PennTB for training and the Chinese PennTB for evaluation. Both of the treebanks use a very similar annotation style; the authors converted them from constituency to dependency syntax using the same algorithm, thus keeping the annotation harmonized. Moreover, they also apply a set of post-projection rules to sanitize the output, addressing some specifics of Chinese language and/or the Chinese treebank annotation both on the morphological and on the syntactic level.

For Spanish, the authors created a small artificial evaluation treebank by parsing text with a rule-based dependency-like parser and then hand-correcting the analyses, both to rectify missing or incorrect annotations as well as to match the intended dependency structures.

1.2.2 Interset

Zeman and Resnik [2008] introduced the idea of cross-lingual delexicalized parser transfer, removing the word forms from the training and evaluation treebanks and using only the morphological annotation as input for the parsers.

Similarly to Hwa et al. [2005], they converted the syntactic annotation of both of the treebanks into the same style, and applied a range of further normalization steps to ensure that the same phenomena are annotated in the same way (focusing e.g. on determiner attachment), which allowed them to perform a reliable evaluation of the automatically induced parse trees.

However, their parsers relied practically exclusively on the morphological tags as input. For their approach to be applicable, it was thus crucial to harmonize the annotation of POS and other morphological categories. For this purpose, they devised a rather sophisticated approach of automatically mapping the original morphological annotations to and from an interlingual representation, called Interset [Zeman, 2008].

The original idea of Interset was to simplify automatic conversions between various tagsets. To avoid the need of a special convertor for each pair of tagsets, Interset operates with the concept of *tagset drivers*. The driver for each tagset

contains a *decoder*, which transforms the tags into an intermediate representation, and an *encoder*, capable of transforming the interlingual representation into the tag. Of course, this is impossible to do completely correctly using a fully automatic delexicalized processing, as the annotations vary immensely in their granularity as well as their definitions of particular POS classes, let alone other morphological features [Zeman, 2010].[1] Nevertheless, the Interset quickly became a very useful tool for tagset harmonization, even if it is rather approximate in some cases, enabling easy transfer of delexicalized parsers across many language pairs. It is still in active development, currently supporting 70 tagsets.[2]

Interestingly, while the interlingual representation was originally intended as intermediate, it gradually became a useful format of tag representation in itself. As we will show, it later found its use in the HamleDT treebanks, and even later, in a modified form, in the Universal Dependencies treebanks.

A simple precursor to Interset can be seen in the work on cross-lingual POS tagging by Yarowsky et al. [2001], who projected POS tags from English to French, with the PennTB tags on the source side and a fine-grained French tagset on the target side. Both of the tagsets capture a range of phenomena specific for the underlying languages, which makes them unsuitable for cross-lingual projection. Therefore, the authors defined a mapping from these tagsets into a set of core POS tags, presumably containing 9 categories.[3]

Even earlier attempts at defining a unified tagset applicable to multiple languages were made by the Eagles advisory group [Leech and Wilson, 1996], the Multext project [Ide and Véronis, 1994], and the Multext-East project [Dimitrova et al., 1998], eventually producing corpora with harmonized morphological annotation.

1.2.3 HamleDT 1.0

Based on previous experience in cross-lingual parsing [Zeman and Resnik, 2008], the Prague group saw the potential utility in not only extending the CoNLL collection to include other treebanks which existed at that time, but also in *harmonizing* the treebanks, i.e. in converting their morphological and syntactic annotation into one common style, realized within the newly founded project HamleDT [Mareček et al., 2011, Zeman et al., 2012].

Naturally, the group decided to use Prague Dependencies as the universal annotation style, but this time using the same guidelines and sets of labels for all of the treebanks, avoiding language-specific modifications and extensions; interestingly but inevitably, this eventually led to the need to slightly modify the annotation of the HamleDT version of the PDT, even though this was the original source of the annotation style definition. To represent the vast amount of existing rich morphological annotations, the Interset of Zeman [2008] was employed.

The group applied automatic conversions of the original treebanks, opting for not using any additional manual annotation. This meant that information not

[1] For example, the definition of pronouns was found to be so inconsistent across languages that Zeman [2010] decided to remove the pronoun category from Interset completely.

[2] `https://ufal.mff.cuni.cz/interset`

[3] The paper is not clear in that respect, listing 9 POS categories as "examples"; so there might be more.

captured by the original annotation was not reflected in the harmonized annotation either, leading to a varying granularity of the annotation. On the other hand, it was possible to rerun the conversion scripts on any potential newer versions of the underlying treebanks. The conversion scripts were carefully manually crafted, featuring some universal parts, as well as components fitted to the individual original treebanks.

1.2.4 Universal Stanford Dependencies

At that time, however, the idea of Universal Stanford Dependencies (USD), based on Stanford Dependencies (SD) of De Marneffe and Manning [2008], was already forming and gaining traction. Several research groups were independently trying to produce a treebank collection covering several languages but using the same annotation style for all of them, inspired by the Stanford Dependencies. Similarly to HamleDT, this caused some initial problems – while the Prague Dependencies annotation was developed primarily for Czech, the SD initially focused on English.

The vague and slowly crystallizing idea was then given a key push by Google researchers. First, Petrov et al. [2012] devised the Universal Part of Speech Tagset (UPT), defining a set of 12 core POS tags, and devised one-way mappings from many existing treebank tagsets to UPT, which they published together with the paper. Subsequently, McDonald et al. [2013] not only devised the Google Stanford Dependencies (GSD) – a universal syntactic annotation style based on SD – but also released a set of 4 new treebanks annotated from scratch using the GSD annotation style, and 2 more treebanks obtained by automatic conversions; five more languages were added in version 2.0.[4] This was unprecedented both in the quality of the annotations, as automatic harmonizations can hardly compete with the manual annotations already being done harmonizedly, as well as in the principledness of the annotation guidelines – while the Prague group was multilingualizing the Prague Dependencies rather iteratively and additively, the Google group designed their annotation style (although based on SD) with multilingualism in mind from the start.

The GSD received a lot of attention from the treebanking and parsing researchers. The Stanford group quickly followed with explicitly introducing USD [de Marneffe et al., 2014], in an attempt to canonize an authoritative language-universal SD-based treebank annotation style, based on their original intentions and linguistic and theoretical considerations behind designing SD. Of course, from the practical point of view, they also took inspiration in existing harmonization efforts, as other researchers' experiences both from automatic harmonizations and from manual harmonized annotations are invaluable.

At the same time, we were working basically on the same thing in the Prague group, which eventually materialized in the Stanfordized HamleDT 2.0 [Rosa et al., 2014], with annotations automatically converted from the Prague Dependencies into our version of USD;[5] avoiding manual annotations, we were unable to reliably produce some of the USD labels, requiring us to abandon some and generalize over others. HamleDT 2.0 featured 30 treebanks in 30 languages anno-

[4] `https://github.com/ryanmcd/uni-dep-tb`

[5] This was joint work of several authors, with most contributions made by Daniel Zeman, Jan Mašek, and Rudolf Rosa.

tated both using Prague Dependencies and USD, thus being the largest treebank collection as well as the largest collection of Stanfordized treebanks at that time.

1.2.5 Universal Dependencies

As the goal of treebank harmonization is to have one style for all of the treebanks, the fact that several research groups were working on that problem independently actually went against the idea itself. Fortunately, this was quickly realized by the community, and the idea of joining forces emerged. After some negotiations and discussions, the Universal Dependencies (UD) group was formed under the lead of Joakim Nivre,[6] joining all of the interested researchers and merging the most significant existing ideas and approaches into one framework [Nivre, 2015, Nivre et al., 2016a];[7] gradually, practically all other harmonization activities ceased.

The key pillars of UD treebanks annotation are:

- Universal Part of Speech (UPOS) tags, based on UPT,
- Universal morphological features, based on the Interset,
- Universal dependency structure and universal dependency labels, based primarily on SD, but including notions from USD, HamleDT, and GSD.

Detailed annotation style descriptions, with a large number of practical examples in many languages, are maintained online.[8]

While HamleDT tried to capture mostly all information annotated in the original treebanks, the focus of UD is a bit different, trying to capture only the important information, but not necessarily everything. More specifically, UD annotations especially try to capture phenomena manifested in multiple languages, but avoiding phenomena that only exist in one or two languages – indeed, it would not make much sense to attempt to devise a language-independent way of annotating something which would be only ever annotated in one language (and it would also not be useful for any cross-lingual experiments, although this is not the primary target for UD). Language-specific extensions are basically limited to the option of appending a language-specific subtype to the universal relation label.[9] However, even here, there is some effort to limit the number of these, and to semi-standardize them by discouraging the introduction of a new language-specific label subtype for a phenomenon for which there already is a subtype defined in another language.[10]

The first set of 10 UD-harmonized treebanks, UD v1.0, was released in January

[6] Who, interestingly, was a member of both the GSD team and of the USD group, under a double-affiliation of Google and Uppsala University, and, as the only such person, was the most logical leader of the joined project; of course, another important factor was his vast expertise in both treebanking and parsing.

[7] `http://universaldependencies.org/`

[8] `http://universaldependencies.org/guidelines.html`

[9] For example, according to `http://universaldependencies.org/fi/dep/nmod-own.html`, the nominal modifier ("nmod") can be subtyped as "nmod:own" in Finnish, which marks that the modifier is in fact an owner of the related subject entity. As this is a frequent and regular construction in Finnish, it makes sense to capture this special type of "nmod", especially as the "nmod" itself is both very general and very frequent, and the ownership information would be lost by only using the universal label.

[10] Thus, "nmod:own" is now defined for Buryat, Erzya, and Finnish; although the annotation guidelines do not make it clear how similarly or differently the label is used in each of those languages.

2015 [Nivre et al., 2015]. A new version of the collection is released every 6 months, adding both conversions of existing treebanks (done manually, semi-automatically, or automatically, potentially with checks and post-corrections), as well as new treebanks annotated in the UD style from scratch. At the time of writing, the latest release is UD v2.1 [Nivre et al., 2017], containing 102 treebanks for 60 languages; the annotation style was partially modified in the transition to the version 2.0.[11] Practically all of the UD treebanks are easily available for download under permissive licences – except for a few, where the texts/word forms cannot be distributed together with the treebanks due to licensing issues, and have to be obtained separately. Thanks to all of the aforementioned characteristics, UD annotation style and datasets have quickly become the current *de facto* standard for most of the work on treebanking, dependency parsing, as well as POS tagging, both monolingual and cross-lingual.

Although the degree of harmonization is very high in UD, there are some areas where the cross-lingual consistency is still rather low. Most notably, in universal morphological features, there is much more variation than we would expect based on just the properties of the languages. A lot of the information is harvested automatically from pre-existing treebanks, which have a varied degree of granularity of the annotations. Therefore, some information is simply not available without manual post-processing.

Unfortunately, even for newly annotated data, incoherences are still common. This is, to some extent, caused by the fact that the guidelines are not sufficiently specific and allow some freedom. The official position of the UD project is that this is on purpose, since too strict guidelines would lead to nonsensical annotations. The UD project thus follows the approach of Google researchers, used for creating their universal treebanks. The idea is to create only rough initial guidelines, then let the annotators work mostly independently on each language, and then review the annotations and refine the initial guidelines to find some universal guidelines by consensus.

In [Rosa et al., 2017], we indeed found that targeting systematic differences in the annotations by a handful of simple additional harmonization steps can bring further notable improvements in cross-lingual parsing.[12] However, it constituted a small but non-trivial amount of manual work for each pair of languages to bring their annotations closer together, which does not scale well to experiments performed on dozens of languages.

In an earlier work, Mašek [2015] tried to automatically detect and correct inconsistencies in the annotations of the HamleDT collection. He found that the automatic error detection is viable to some extent, but the automatic correction of errors turned out to be a very hard task, with no clearly positive results being reported in that work.

1.3 Treebank datasets used in our experiments

As our research was done over the course of several years, during which a lot changed in the field of treebank collections, we did not keep our dataset fixed

[11] `http://universaldependencies.org/v2/summary.html`

[12] The additional harmonizations were devised by Daniel Zeman.

throughout the whole time. For the earlier experiments, we used the Stanfordized HamleDT 2.0 treebanks (Section 1.3.1); later on, we switched to using Universal Dependencies v1.4 (Section 1.3.2).

1.3.1 HamleDT 2.0 dataset

When the research for this thesis commenced, the HamleDT 2.0 collection [Rosa et al., 2014, Zeman et al., 2014] was by far the largest as well as most harmonized existing treebank collection and thus the logical, or probably even the only reasonable, choice of dataset. Our work was thus among the first ones to be applied to a really large harmonized treebank collection. HamleDT 2.0 featured 30 treebanks in 30 languages in both the Prague Dependencies and the Universal Stanford Dependencies annotation style – and although unfortunately only some of them were freely available to the public, we had access to all of them, giving us a great advantage then. Thus, the experiments done early on in our research are performed and evaluated using the Stanfordized HamleDT 2.0 dataset.

The Stanfordized treebanks are annotated with a set of 33 dependency relation labels inspired by the USD of de Marneffe et al. [2014], and with the 12 UPT tags as defined by Petrov et al. [2012], which we reproduce here from the official website:[13]

VERB	verbs (all tenses and modes)
NOUN	nouns (common and proper)
PRON	pronouns
ADJ	adjectives
ADV	adverbs
ADP	adpositions (prepositions and postpositions)
CONJ	conjunctions
DET	determiners
NUM	cardinal numbers
PRT	particles or other function words
X	other: foreign words, typos, abbreviations
.	punctuation

As we initially focused solely on parsing, we use the gold-standard UPT tags in all our experiments conducted on the HamleDT dataset. The treebanks also contain fine-grained Interset morphological annotations, but we did not use these in our experiments.

We list all of the treebanks in Table 1.1, together with the sizes of their training and test sections in thousands of tokens, and a reference to the original source of the treebank, as all HamleDT treebanks are semi-automatic conversions of pre-existing treebanks. For details about the syntactic annotation and the harmonization process, please refer to the original publication of Rosa et al. [2014].

We used these treebanks in development and tuning of our cross-lingual parser transfer methods. Therefore, to avoid overtuning to the particular set of treebanks, we split them into 12 *development treebanks* and 18 *evaluation treebanks*, and initially evaluated our methods only on the development treebanks; the evaluation treebanks were only used in final evaluations, once the hyperparameters

[13] `https://github.com/slavpetrov/universal-pos-tags`

	Language	Treebank (kTokens) Train	Test	Reference
Development treebanks				
ar	Arabic	250	28	[Smrž et al., 2008]
bg	Bulgarian	191	6	[Simov and Osenova, 2005]
ca	Catalan	391	54	[Taulé et al., 2008]
el	Greek	66	5	[Prokopidis et al., 2005]
es	Spanish	428	51	[Taulé et al., 2008]
et	Estonian	9	1	[Bick et al., 2004]
fa	Persian	183	7	[Rasooli et al., 2011]
fi	Finnish	54	6	[Haverinen et al., 2010]
hi	Hindi	269	27	[Husain et al., 2010]
hu	Hungarian	132	8	[Csendes et al., 2005]
it	Italian	72	6	[Montemagni et al., 2003]
ja	Japanese	152	6	[Kawata and Bartels, 2000]
Evaluation treebanks				
bn	Bengali	7	1	[Husain et al., 2010]
cs	Czech	1,331	174	[Hajič et al., 2006]
da	Danish	95	6	[Kromann et al., 2004]
de	German	649	33	[Brants et al., 2004]
en	English	447	6	[Surdeanu et al., 2008]
eu	Basque	138	15	[Aduriz et al., 2003]
grc	Ancient Greek	304	6	[Bamman and Crane, 2011]
la	Latin	49	5	[Bamman and Crane, 2011]
nl	Dutch	196	6	[van der Beek et al., 2002]
pt	Portuguese	207	6	[Afonso et al., 2002]
ro	Romanian	34	3	[Călăcean, 2008]
ru	Russian	495	4	[Boguslavsky et al., 2000]
sk	Slovak	816	86	[Šimková and Garabík, 2006]
sl	Slovenian	29	7	[Džeroski et al., 2006]
sv	Swedish	192	6	[Nilsson et al., 2005]
ta	Tamil	8	2	[Ramasamy and Žabokrtský, 2012]
te	Telugu	6	1	[Husain et al., 2010]
tr	Turkish	66	5	[Atalay et al., 2003]

Table 1.1: List of HamleDT 2.0 treebanks.

of our methods were fixed. However, we do not split the treebanks into sources and targets; to parse a given target language, any of the remaining treebanks can be used as the source, i.e. we use the leave-one-out approach.

To tune our methods to perform well in many different situations, we chose the development set to contain both smaller and larger treebanks (but not the largest ones, to enable faster development), a pair of very close languages (*ca, es*), a very solitary language (*ja*), multiple members of several language families (Uralic, Romance), and both primarily left-branching (*bg, el*) and right-branching (*ar, ja*) languages.[14]

Note that the treebanks for some languages are very small (*bn, et, ta, te*), bringing us quite close to the intended under-resourced setting.

1.3.2 Universal Dependencies 1.4 subset

In our more recent experiments, we switched from the HamleDT 2.0 dataset to UD 1.4 [Nivre et al., 2016b], as this was the newest UD release available at the time of the switch, featuring 64 treebanks for 47 languages.

The UD 1.4 are annotated using the set of 17 UPOS tags of UD v1, which we reproduce here from the official website:[15]

ADJ	adjective
ADP	adposition
ADV	adverb
AUX	auxiliary verb
CONJ	coordinating conjunction[16]
DET	determiner
INTJ	interjection
NOUN	noun
NUM	numeral
PART	particle
PRON	pronoun
PROPN	proper noun
PUNCT	punctuation
SCONJ	subordinating conjunction
SYM	symbol
VERB	verb
X	other

Most treebanks are also annotated with Universal features, but we did not use these in most experiments.

The syntactic annotation of UD 1.4 uses a set of 40 universal dependency relation labels, which we list in Attachment A. Furthermore, many languages use language-specific relation subtypes, such as *nmod:poss*, with the universal label

[14] We use the terms *left-branching* and *right-branching* in the usual way, where for languages which are written right-to-left, the terms assume that we reorder the language into the left-to-right order; i.e. *left-branching* corresponds to *head-final* and *right-branching* corresponds to *head-initial* when processing the sentence from its beginning to its end.

[15] `http://universaldependencies.org/docsv1/u/pos/index.html`

[16] Changed to *CCONJ* in UD v2.

	Language	Treebank (kT) Train	Dev	Para data (kS)
Target languages				
da	Danish	89	5.9	120
el	Greek	47	6.0	102
hu	Hungarian	33	4.8	118
id	Indonesian	98	12.6	130
ja	Japanese	80	9.1	20
kk	Kazakh	5	0.7	60
lv	Latvian	13	3.6	95
pl	Polish	69	6.9	119
sk	Slovak	81	12.4	117
ta	Tamil	6	1.3	92
tr	Turkish	41	8.9	117
uk	Ukrainian	1	0.2	98
vi	Vietnamese	32	6.0	119
Source languages				
ar	Arabic	226	28	111
bg	Bulgarian	124	16	101
ca	Catalan	429	58	22
cs	Czech	1,672	175	121
de	German	270	12	117
en	English	271	33	150
es	Spanish	836	95	109
et	Estonian	188	23	115
fa	Farsi	121	16	63
fi	Finnish	290	25	114
fr	French	356	39	113
he	Hebrew	135	11	101
hi	Hindi	281	35	89
hr	Croatian	128	5	117
it	Italian	271	11	123
nl	Dutch	286	11	124
no	Norwegian	244	36	121
pt	Portuguese	478	217	129
ro	Romanian	163	28	122
ru	Russian	930	120	97
sl	Slovene	136	17	117
sv	Swedish	131	17	115

Table 1.2: UD 1.4 dataset as used in our experiments. We separate the languages into source languages and target languages. For each language, we report the size of its training and development treebank in thousands of tokens, and the size of the WTC parallel data aligned to English in thousands of sentences.

and the subtype label separated by a colon. While these subtypes are partially harmonized, they are not, by definition, sufficiently language-independent, and we therefore remove them from both the training and evaluation treebanks, keeping only the universal parts of the labels.

We do not use all of the treebanks that are available in UD 1.4 – we remove treebanks for which parallel data are not available in the WTC (see Section 1.4.2), as we need those for our experiments. This eliminates the treebanks of dead languages (Ancient Greek, Church Slavic, Coptic, Gothic, Latin, and Sanskrit), several minority languages (Basque, Galician, Irish, and Uighur), the Swedish sign language, and Chinese. If multiple treebanks exist for a given language, we concatenate them all into one. We list the resulting 35 treebanks in Table 1.2, showing their sizes in thousands of tokens for both the *train* section, which we use for training, and the *dev* section, which we use for evaluation. We also list the number of parallel sentences available in the WTC, in thousands – this number is different for each language pair, but we list the number of sentences parallel with English, as we pivot the sentence alignments through English.

We separated the languages into two non-overlapping groups: the source languages and the target languages. The source languages are assumed to be resource-rich, with the *train* sections of the source language treebanks being used for training taggers and parsers. The target languages are used to simulate low-resource languages; we do not use the *train* sections of these treebanks, and we only use the *dev* sections for evaluation of our methods (see Section 2.3.4). The languages were split into the two groups based on the size of their training treebanks: if the treebank contains more than 100,000 tokens, we use the language as source; if it is smaller, we designate it a target language. This leads to a set of 22 source languages and 13 target languages; as some of the target language treebanks are really small, treating the languages as under-resourced seems at least partially justified.

VarDial subset

In the VarDial cross-lingual parsing shared task [Zampieri et al., 2017], the organizers specified treebanks and parallel corpora to use, including a specification of source and target languages. As the focus of the shared task was on close language pairs, a set of only three very close language pairs was selected. The task thus consisted of parsing Slovak using Czech resources, parsing Croatian using Slovene resources, and parsing Norwegian using Danish and Swedish resources; the last language "pair" is thus actually a triplet, but, following suggestions provided by the organizers, we simply concatenated the Danish and Swedish data into one Dano-Swedish resource, and further treated this as a single source language.

We list the dataset sizes on the first 3 lines of Table 1.3. The treebanks come from the UD 1.4 dataset. The parallel data come from the OpenSubtitles2016 corpus (Section 1.4.1) and are thus much larger than the WTC; therefore, we report their sizes in millions of sentences instead of thousands.

Extended VarDial subset

For further exploratory experiments, we then extended the VarDial dataset by 7 more languages, organized into 9 further language pairs; the whole Extended

	Languages		Treebank sizes (kT)		Para data
	source	target	source train	target dev	size (MS)
Source language very similar to target					
cs→sk	Czech	Slovak	1,173	12	5.7
da+sv→no	Danish+Swedish	Norwegian	156	36	9.1
sl→hr	Slovene	Croatian	112	5	12.8
fr→es	French	Spanish	356	41	32.5
es→fr	Spanish	French	382	39	32.5
cs→pl	Czech	Polish	1,173	7	24.2
Source language less similar to target					
it→ro	Italian	Romanian	271	28	22.1
en→sv	English	Swedish	205	10	13.3
en→de	English	German	205	12	15.4
de→sv	German	Swedish	270	10	6.4
de→en	German	English	270	25	15.4
fr→en	French	English	356	25	37.3

Table 1.3: Overview of the VarDial dataset (first 3 lines) and Extended VarDial dataset (all lines), listing the source and target languages, the treebank sizes in thousands of tokens, and the sizes of parallel data in millions of sentences.

VarDial dataset thus contains the 12 language pairs listed in Table 1.3. This time, we did not concatenate multiple treebanks for the same language – we simply always used only the largest one. For the parallel data, we again used OpenSubtitles2016.

Following the approach of the organizers of the VarDial shared task, we preselected source-target language pairs which we believed to be rather close, based on our linguistic intuition. Moreover, we subdivided the language pairs into two groups, the first containing pairs of languages which we believe to be very similar, and the second one consisting of language pairs that still belong to the same typological genera, but we believe them to be more distant than the language pairs in the first group.

1.4 Parallel corpora

A parallel text corpus is a resource consisting of a text in one language and its translation in another language. Parallel texts are "natural resources", produced by human translators and published for various reasons – we can often easily get religious texts, international laws, film subtitles, etc. Parallel corpora are often freely available for download, or can be compiled from parallel data harvested from the internet. Still, for under-resourced languages, even the amount of available parallel data is usually lower than for resource-rich languages.

Parallel corpora are typically used in NLP to train Statistical Machine Translation (SMT) systems, which can be useful for many tasks, including cross-lingual parsing. In the cross-lingual projection approach, parallel data are even used directly to project annotations from its one side to the other side, without using an SMT system.

In many cases, translations of the same texts are available in multiple lan-

Corpus	Available languages easily	potentially	Typical number of sentences
OpenSubtitles	62	78	5M – 30M
Watchtower	135	300	100k – 150k
Bible	100	1200/4000	10k – 30k
UDHR	400	544	60 – 70

Table 1.4: Overview of some parallel and multiparallel corpora, with the number of languages for which it is easily or at least potentially available, and a typical size in number of sentences.

guages; such resources are usually referred to as multiparallel corpora, and can be even more useful for cross-lingual processing.

In the canonical format, the corpora are sentence-aligned, i.e. there is always a pair of one source sentence and one target sentence that correspond to each other. Some data already more or less arrive in this format (e.g. the Bible), but usually, the alignment has to be estimated. Fortunately, this is generally not a difficult tasks, as high performance sentence aligners exist, such as the Hunalign of Varga et al. [2007].

Moreover, many parallel corpora can be downloaded from linguistic repositories, such as the OPUS collection of Tiedemann [2012],[17] which publish them in a preprocessed format, usually including sentence segmentation and sentence alignment, and often also tokenization.

In Table 1.4, we present an overview of parallel corpora which are available for a large number of languages (in fact, all of the listed corpora are actually multiparallel, at least to some extent).

In our experiments, we have only used the first two, OpenSubtitles and WTC. However, we also review the other two, Bible and Universal Declaration of Human Rights (UDHR), as they are available for an even larger number of languages than the first two, thus broadening the potential scope of cross-lingual parsing methods. We discus all of these corpora in more detail further on.

1.4.1 OpenSubtitles

The OpenSubtitles corpora are film and TV series subtitles and their translations provided by volunteers through the OpenSubtitles web portal.[18] While the translations are of varying quality, they have been repeatedly successfully used by many researchers. The data are typically sufficiently large, making it possible to train high-quality SMT systems, while also being available in a respectable number of languages. Unfortunately, these are mostly resource-rich languages; for resource-poor languages, little or no data are often available in this corpus, which gravely limits the usefulness of this dataset in the intended use case of cross-lingual parsing.

Nevertheless, we employed the OpenSubtitles data in some of our experiments, in particular using the OpenSubtitles2016 version, published by Lison and Tiedemann [2016]. We report the sizes of the parallel data which we used together with the particular languages in Section 1.3.2. We always split of the first 10,000

[17] `http://opus.nlpl.eu/`

[18] `http://www.opensubtitles.org/`

sentences from the dataset as development data, used for tuning the MT systems, and the last 10,000 sentences as test data, used to intrinsically evaluate the quality of the MT systems.

1.4.2 Watchtower

Agić et al. [2016] introduced a much more realistic resource for cross-lingual parsing: the Watchtower Corpus (WTC).[19] It consists of texts of the Watchtower magazine, published by Jehovah's Witnesses via the Watch Tower Bible and Tract Society of Pennsylvania in a large number of languages, including many under-resourced ones. The texts are available on the Watchtower Online website,[20] from which they were scraped by Agić et al. [2016] and compiled into the WTC.

The WTC contains texts in 135 languages. For each language, the corpus contains at least 27,000 and no more than 167,000 sentences; the average number of sentences is 116,000, the median is 127,000. These are thus drastically smaller data than the OpenSubtitles, inevitably leading to considerably worse results. However, in line with Agić et al. [2016], we believe this to be a much more realistic setting for under-resourced languages, leading to more plausible estimates of the parsing accuracies – for real under-resourced languages, really large parallel corpora are typically simply not available. We thus use OpenSubtitles for several rather exploratory experiments, but ultimately apply WTC in our final setups. However, given the small size of the WTC, we typically split of a lower absolute number of sentence pairs for tuning (dev) and for evaluation (test) of the MT systems than we did for OpenSubtitles, never taking away more than 20% of the corpus.[21]

On the plus side, the WTC data are massively multiparallel, as they consist of translations of the same texts. The texts in WTC are tokenized on punctuation symbols by a trivial tokenizer. This means that languages which do not separate words by spaces, such as Japanese, are not properly tokenized; the results which we report for Japanese thus suffer from this, but we find it useful to investigate what the results are under such settings. The texts are also segmented into sentences using a similar approach, with one sentence per line. The average number of tokens in an English sentence is 16.5 in WTC.

For some language pairs, automatic sentence alignment is part of the corpus, while for other it is not, thus requiring us to rerun Hunalign on the data. As running Hunalign for all language pairs was rather computationally demanding, we instead took a pivoting approach, sentence-aligning each language to English as the pivot. We then construct sentence-alignment for any pair of languages from their alignments to English, which inevitably leads to omitting sentences that appear in both of these languages but seem to be missing in the English text. However, as the English text is the source for all of the translations, we did not observe a large ratio of such omissions in practice. We believe that, due to the nature of the data, the pivoting approach might actually lead to better

[19]The WTC was made available to us by Željko Agić via direct e-mail contact.

[20]`https://wol.jw.org/`

[21]Specifically, we take 10,000 sentence pairs for dev and another 10,000 sentence pairs for test only if there is at least 100,000 parallel sentences for the language pair. Otherwise, we take 1000, 100, or even only 10, so that at least 80% of the sentences are left for training.

results than pairwise alignments; but we have not measured that in any way.

We list all of the 135 languages present in the WTC in Section B.1 of Attachment B. However, it seems that many more languages are available on the Watchtower website; at the time of writing, it advertises texts in 301 languages, which suggests that the scope of the corpus (and, subsequently, of the presented cross-lingual methods) could still be considerably extended. We include a listing of all these languages in Section B.2.

1.4.3 Bible

While Watchtower covers a respectable number of languages, it is still clearly surpassed by the Bible, which seems to be freely available for download in 1,200 languages at Bible.com.[22] Furthermore, WorldBibles[23] claim to provide access to the text of Bible in more than 4,000 languages; however, the portal only contains links to websites that claim to have the text of the Bible, which might be available for free download, for purchase, in printed form, or even not at all; the lower number of 1,200 thus seems somewhat more realistic.

Interestingly, the largest precompiled corpus of Bible texts that we were able to find is the Edinburgh Bible Corpus (EBC)[24] of Christodouloupoulos and Steedman [2015], covering only 100 languages, i.e. less than the WTC. Moreover, according to Agić et al. [2016], the WTC texts are more useful for cross-lingual NLP, as they better correspond to current everyday language than the Bible. On the other hand, Watchtower is only published in living languages, while EBC also contains several extinct languages, such as Coptic or Latin. Nevertheless, we did not use the Bible in our experiments.

1.4.4 Universal Declaration of Human Rights

The text of the UDHR itself is very tiny – its 30 articles are typically formulated in approximately 65 sentences. Therefore, we do not actually use it in our experiments.

However, it stands out among the other resources by being easily available for the largest number of languages. It is made available by the "UDHR in Unicode" project, and can be directly downloaded from the webpage of The Unicode Consortium[25] as a ZIP file containing 455 Unicode text files. Even if we omit unidentified languages and multiple variants of the same language, we still get the text in 400 languages.

The original and authoritative source, which publishes a slightly larger number of translations of the UDHR, is the Office of the United Nations High Commissioner for Human Rights.[26]

[22] `http://bible.com`
[23] `http://worldbibles.org/`
[24] `http://christos-c.com/bible/`
[25] `https://unicode.org/udhr/downloads.html`
[26] `http://www.ohchr.org/EN/UDHR/Pages/SearchByLang.aspx`

1.5 Other data

1.5.1 Monolingual plaintext data

Typically, plaintext monolingual data are available in a larger amount than parallel data, let alone annotated data. Therefore, it makes sense to try to leverage them, as such data can be very useful for learning about the language. More specifically, we can typically use monolingual texts to train the language models for an SMT system (Section 5.5), or to pre-train word embeddings for a neural parser (Section 5.3). Other possible uses exist as well, such as training a language identifier (Section 4.1.5), or attempting machine translation without parallel data [Rosa, 2017, Conneau et al., 2017]. Moreover, if we have no other resources than plaintext monolingual data, we can still perform unsupervised parsing [Klein and Manning, 2004, Mareček, 2016a] as a backoff – we compare our results to unsupervised parsing in Section 5.6.4.

Interestingly, for the least-resourced languages, the situation is often somewhat reversed, as in many cases, the largest datasets available for a resource-poor language tend to actually be multiparallel datasets – the Bible, Watchtower texts, and/or the UDHR.

In the realm of monolingual texts, one of the massively multilingual yet easily available corpora are the texts of Wikipedia, which are free to download for any of its approximately 300 language editions.[27] Most of the editions contain several thousands of articles; some of the articles consist of several paragraphs of text, while most are *stubs*, only containing a handful of sentences. A median Wikipedia edition thus typically contains something between tens of thousands and millions of words.

Another good option may be the W2C corpus of Majliš and Žabokrtský [2012], which contains texts from Wikipedia as well as from the Web in 120 languages, offering more than one million of words for most of the languages.[28]

In any case, the target-language side of available parallel data can always be used as monolingual data, either in combination or instead of monolingual texts. For simplicity, in our experiments, we always use the target side of the parallel data instead of monolingual data even in cases where larger monolingual data are available.

1.5.2 Linguistic catalogues

The World Atlas of Language Structures (WALS) of Dryer and Haspelmath [2013] is one of the most well-known and respectable sources of information about world's languages. It is a manually curated database, gathering typological information about a wide range of languages and organized in a structured way. The information itself comes from numerous studies, which are linked from the atlas. Importantly, the database is freely accessible both on the web and as a downloadable data resource, making it very useful and popular; we are unaware of any larger or otherwise better database available for free download in such an easy-to-use and machine-readable format.

[27] `https://en.wikipedia.org/wiki/List_of_Wikipedias#Detailed_list`
[28] `http://ufal.mff.cuni.cz/w2c`

At the time of writing, WALS contains 2679 entries,[29] listing up to 192 linguistic features for each of the languages (or 202, if we also count language codes and names, and genealogical and areal information). The features are assigned names and identifiers (e.g. "81A Order of Subject, Object and Verb"), and each feature has a fixed and limited set of possible values (e.g. "1 SOV", "2 SVO"... "6 OSV", and "7 No dominant order"). The features are grouped into several thematic areas, such as phonology, morphology, word order, or lexicon. Unfortunately, most languages are not covered by most of the features, i.e. the vast majority of the feature values are blank – this is often not because the features would be irrelevant or unknown for the languages, but simply because their values are not covered by the primary sources upon which the database is built.

We review the use of WALS for cross-lingual parsing in Section 4.1.3.

[29]There are multiple entries for some languages, corresponding to different dialects or varieties of the language; the total number of languages is thus lower, around 2400.

2. Dependency Parsing

In computational linguistics, parsing, or syntactic analysis, is the act of revealing the structural (syntactic) relations between words in a sentence, and presenting them in the form of a graph, usually an ordered rooted parse tree. Syntactic parsing is a classical NLP task, with the tool that performs this task being called a parser. The input to a parser is typically a tokenized and morphologically annotated sentence, and the output is a syntactic parse tree of the sentence.

As we explained Section 1.1, there is a range of linguistic theories and annotation styles that define a particular syntactic representation. However, typical modern data-driven parsers are mostly theory-oblivious, as long as the theory is realized in a treebank annotated with standard dependency or constituency trees; the only important distinction is between phrase-structure parsing and dependency parsing, which call for different parsing algorithms. The syntactic theory which underlies the annotated treebank is external to the parsers, as they simply learn to reproduce the annotators' decisions. For cross-lingual parsing, we need the annotations to be harmonized, as we explained in Section 1.2; however, it is typically not crucial which particular annotation style is used, as long as it is the same one for all of the datasets.

In this thesis, we focus on the dependency parsing paradigm, which has become the *de facto* standard in recent years, especially in the multilingual setting, with large collections of harmonized treebanks being available for dozens of languages.

Specifically, we have used two parsers throughout our work. In earlier experiments, we use the MSTperl parser, which is a representative of the graph-based parsers (Section 2.1). In later experiments, we use the UDPipe/Parsito parser, which is transition-based (Section 2.2).

We partially draw from [Jurafsky and Martin, 2017] in this chapter.

2.1 Graph-based parsing

In the graph-based approach to dependency parsing, the parse tree of a tokenized sentence is obtained by the following steps:

1. construct a complete directed graph with the tokens as nodes,
2. weight each edge by its score predicted by a trained model,
3. find the directed Maximum Spanning Tree (MST) of the graph.

Or, using the linguistic rather than graph-theoretic notions:

1. construct all potential dependency edges,
2. estimate the quality of each potential edge,
3. construct the highest quality parse tree, where the quality of a parse tree is measured by the total quality of the edges in it.

Graph-based dependency parsing was introduced by McDonald et al. [2005a], based on the work of Eisner [1996], in the form of the MSTParser. The parser was first developed as a projective one, i.e. unable to produce parse trees with crossing

edges; its non-projective variant was then introduced by McDonald et al. [2005b]. The parser is an unlabelled one by default, producing only the tree structure without dependency relation labels; the labelling is to be done independently in a second stage [McDonald et al., 2006].

We review the MSTParser in more detail, as understanding its internals will be useful in Section 4.3.1 and Section 4.3.2.

2.1.1 First-order edge factorization

In the base *first-order* variant of MSTParser, each edge is scored independently, and the score of the final parse tree is simply obtained as the sum of the scores of the individual edges.

This has the advantage of the parser efficiently exploring all possible parse trees: if we assume that the edge scores predicted by the trained model are optimal, the algorithm is guaranteed to return the optimal parse tree in terms of maximizing its total score.

The clear drawback is that the parsing decisions cannot inform each other – for example, if there are two good candidates for the subject of a verb, the parser has no means of trying to select one or the other but not both. This was later addressed in second-order and higher-order parsing [McDonald and Pereira, 2006, Carreras, 2007].

2.1.2 MSTParser model and training

The MSTParser model uses a set of binary features F that are assigned weights w_f by training on a treebank.

When parsing a sentence, the parser assigns each potential edge e a score s_e which is the sum of weights of features that are active for that edge:

$$s_e = \sum_{\forall f \in F} f(e) \cdot w_f \,, \tag{2.1}$$

where $f(e)$ is a binary feature function, returning 1 if the feature is active for edge e and 0 if it is not.

The MSTParser is trained with Margin-infused Relaxed Algorithm (MIRA), an ultra-conservative machine learning algorithm of Crammer and Singer [2003], similar in principle to the Perceptron algorithm.

2.1.3 The MST algorithms

To find the MST of the weighted oriented graph, there is a choice of two algorithms.

In the non-projective variant of the parser [McDonald et al., 2005b], the algorithm of Chu and Liu [1965] and Edmonds [1967] is used. The algorithm is $O(n^2)$, but can only be efficiently used with first-order edge factorization, as already second-order non-projective parsing is NP-hard [McDonald and Pereira, 2006].

The projective parser [McDonald et al., 2005a] uses the algorithm of Eisner [1996], which is $O(n^3)$. However, its advantage is that it can be extended

to higher-order edge factorizations; for second-order parsing, this is even without an increase of the computational complexity. Therefore, in later versions of MSTParser, the Eisner algorithm is used, using an approximate algorithm for non-projective parsing.

The approach of finding the MST of a weighted graph has also been exploited for combining multiple parsers in the work of Sagae and Lavie [2006], which we build upon in Section 4.3.1. For this task, there is no clear use of higher-order features, and the exact Chu-Liu-Edmonds algorithm thus can be used.

Zeman and Žabokrtský [2005] took a simpler approach to the problem, using a greedy heuristical search instead of an exact algorithm to try to find the MST; however, their results seem to be rather competitive, suggesting that even such approaches might be sufficient in practice.

2.1.4 MSTperl

In the earlier stages of our research, especially in Chapter 3 and Chapter 4, we use the MSTperl parser.

MSTperl [Rosa et al., 2012a, Rosa, 2015a] is our reimplementation of the MSTParser of McDonald et al. [2005b]; as the name suggests, the parser is implemented in Perl. We decided to reimplement the parser for research purposes, as this allowed us to experiment with modifying its internals.[1]

We originally implemented MSTperl for our previous work on automatic post-editing of machine translation outputs [Rosa et al., 2012b, Rosa, 2013]. When we moved to cross-lingual parsing, we found it particularly useful to use a parser where we can easily access its internals. We benefited from this mainly in two ways. First, it made it easy for us to use the inference component of the MSTParser (i.e. the MST algorithm) to combine parse trees produced by multiple parsers (Section 4.3.1). And second, it enabled us to directly work with and manipulate the trained parser models, which we tried to use for parser model interpolation in Section 4.3.2.

The MSTParser can come in various flavours; in implementing MSTperl, we used the following particular setup of the parser:

Non-projective The parser is not constrained to producing projective parse trees, using the Chu-Liu-Edmonds MST algorithm for inference. This variant was selected in particular because we were originally mainly interested in parsing Czech sentences, which contain a rather large number of non-projective edges.

First-order The parser only takes first order features into account, i.e. all edges are scored independently. This is implied by using the non-projective variant of the parser (the projective variant can be extended to use second-order features, where the model scores pairs of edges rather than individual edges).

Single-best The parser only produces the top scoring parse tree. Variants of MSTParser that perform k-best parsing also exist, and this might even be beneficial in our setting; however, we did not experiment with that.

[1] While even the original implementation of the parser is open-source, we found its source code to be rather difficult to work with.

Unlabelled The original MSTParser is defined as an unlabelled one, producing only the bare dependency tree without dependency relation labels. McDonald et al. [2006] suggest to label the parse tree with an independent second-stage labeller, which we did in [Rosa and Mareček, 2012] and used in the translation post-editing task. However, for cross-lingual parsing, we only worked with the unlabelled dependency trees produced by MSTperl.

Closed-form update McDonald et al. [2005a] suggest using k-best parsing in the training, applying a quadratic programming algorithm to find the minimal update of the model weights necessary to separate the correct parse tree from high-scoring incorrect parse trees by a margin proportional to the number of incorrectly produced edges. In MSTperl, we only do a closed form single-best update, adjusting the model weights uniformly to separate only the single top-scoring parse tree from the correct one by the error margin.

Feature set

Our feature set is based on [McDonald et al., 2005a], and consists of various conjunctions of the following features:

POS We use the 12-value UPT tags of Petrov et al. [2012]. For each potential edge, we use the POS tag of the head token, the dependent token, their linear neighbours, and all of the tokens that appear in the sentence between the head and the dependent tokens.

Distance We compare the positions of the head and dependent tokens in the sentence by measuring their token-based distance, in a signed way to also incorporate the information about their order: $position_{head} - position_{dependent}$; to avoid data sparseness, the distance is bucketed into the following buckets: $+1;\ +2;\ +3;\ +4;\ \geq+5;\ \geq+11; -1;\ -2;\ -3;\ -4;\ \leq-5;\ \leq-11$.

Form We use the word form of the head and dependent tokens.

Lemma We use the morphological lemma of the head and dependent tokens.

We use exactly the same settings of the parser in all experiments; for delexicalized parsing, the form and lemma features are removed, as will be explained in Chapter 3. The configuration files that contain the feature sets and other settings, as well as the scripts we used to conduct our experiments, are available in [Rosa, 2015b].

Technical considerations

In our cross-lingual parsing experiments, we trained the parser using only 3 passes over the training data, instead of the more usual 10 or so, as it has been suggested to us that this makes the parser perform better in this setting. The idea is that with more iterations, the parser becomes "overtrained" to the language of the training data, making it less robust to the transfer to a different language.

Moreover, when training the parser over extremely large datasets, such as a concatenation of a larger number of treebanks, we only used one iteration over the training data, since even then the parser took many days to train.

As the MSTperl parser is particularly weak in some respects (see the setup description above), the accuracies obtained when using it are very likely lower than they could be if a stronger parser was used. However, at that point of our research, we focused on comparing various setups to each other, rather than trying to achieve the best absolute accuracies.

Of course, there is a risk that some of the conclusions that we made when using the MSTperl parser might not be universally valid for other parsers. However, when we later moved to a much newer state-of-the-art transition-based parser (Section 2.2), we saw very similar results relatively, suggesting that using one or another particular parser is not of crucial importance to the general approach. Still, we did not replicate all of the intermediary experiments, so it is indeed possible that some of our choices which were based on the performance of the MSTperl parser might not be optimal for other parsers.

2.2 Transition-based parsing

The approach of transition-based dependency parsing is motivated by the classical shift-reduce parsing of Aho and Ullman [1972], but replacing the "reduce" operation with "left-arc" and "right-arc" operations; it is thus also sometimes referred to as shift-arc parsing. It was introduced in the MaltParser by Nivre [2003], based on the work of Covington [2001].

2.2.1 Arc-standard transition-based parsing

A transition-based parser operates with a stack and a buffer; at the start of the algorithm, the buffer contains the tokens of the sentence to be parsed, while the stack is empty (or contains a technical root node). The algorithm then proceeds by reading in the input sentence token-by-token, shifting its tokens onto the stack, and building partial dependency trees on the stack, eventually finishing with an empty buffer and the final parse tree on the stack.

Each tree on the stack is represented by its root node; other nodes of the tree than the root node are inaccessible for the algorithm. In each step, the algorithm decides to perform one of the following operations:

shift Take the token from the top of the buffer and push it onto the top of the stack as a new root node with no children.

left-arc Pop the top two root nodes from the stack, create a dependency edge between them, governed by the first node (i.e. the node that was on top of the stack), and push the resulting tree (i.e. the first node) back onto the stack.

right-arc Pop the top two root nodes from the stack, create a dependency edge between them, governed by the second node, and push the resulting tree (i.e. the second node) back onto the stack.

This base variant of MaltParser is usually referred to as *arc-standard*, in contrast to the *arc-eager* variant of Nivre [2004].

The algorithm decides which action to take based on a trained multiclass classifier,[2] using both the contents of the buffer and of the stack to provide input features. In this way, the algorithm is informed both by its past decisions and by the input sentence, making the MaltParser stronger than the MSTParser in this respect.

On the other hand, the parser does not take all possible parse trees into account, but rather constructs the parse tree greedily, with each action further limiting its search space – if the parser makes a wrong decision at some point, it is unable to recover from it later. This makes MaltParser faster than MSTParser (linear instead of quadratic or cubic), but at the same time making it weaker in this respect. However, this can be partially alleviated by the use of beam search [Zhang and Clark, 2008].

The base variant of the algorithm is only capable of projective parsing. To induce non-projective edges, the algorithm can be enriched by a fourth operation, "swap", which can change the order of the nodes [Nivre, 2009].

In the way we described the algorithm, it only performs unlabelled parsing. However, it is straightforward to extend it to labelled parsing by introducing labelled arc operations: in the labelled variant of MaltParser, instead of a single pair of unlabelled "left-arc" and "right-arc" operations, there is a pair of arc operations for each dependency relation label (e.g. "left-arc-subject", "right-arc-subject", "left-arc-object"...); the operation then creates a labelled dependency edge instead of an unlabelled one. In contrast to MSTParser, parsing and labelling is thus performed jointly.

2.2.2 Parsito/UDPipe

In our later experiments reported in this thesis, we use the Parsito transition-based parser of Straka et al. [2015], which is a neural variant of the MaltParser, following the work of Chen and Manning [2014]. The parser works similarly to MaltParser, but the action selection is made by a neural network classifier, and word embeddings are used to represent input features.

We use Parsito as part of the UDPipe pipeline of Straka et al. [2016], which also contains a tokenizer and the MorphoDiTa tagger of Straková et al. [2014], which we use for POS tagging.

In comparison to MSTperl, Parsito is rather fast and memory-efficient, both in training and in inference, and achieves state-of-the-art results. It also natively supports UD and the CoNLL-U data format, which MSTperl does not. On the other hand, we are losing the easy access to the parser internals, which we had with MSTperl. However, we have found Milan Straka to be very responsive both in providing support for the parser as well as in developing simple introspection tools that we needed in our research.

Switching to Parsito thus gave us access to a high performance parser with little compromise. Moreover, a significant added benefit of using Parsito is the fact that it employs word embeddings on its input, which can be pre-trained on plaintext monolingual data. In our setting, this allows us to pre-train the word-embeddings for the cross-lingual parser on target-language plaintext, which can

[2]Usually using either multinomial logistic regression or support vector machines.

serve as a semi-supervised adaptation of the parser to the target language; we detail this approach in Section 5.3.

Features and configuration

We used Parsito with rather standard settings, except for selectively turning some input features off and on.

We list here the settings of the parser that we worked with in our research, reproduced according to the UDPipe manual:[3]

- `embedding_upostag` (default 20): the dimension of the UPOS embedding
- `embedding_feats` (default 20): the dimension of the Features embedding
- `embedding_xpostag` (default 0): the dimension of the XPOS embedding
- `embedding_form` (default 50): the dimension of the Form embedding
- `embedding_form_file`: word embeddings in word2vec textual format

If the dimension of an input feature is set to 0, such feature is not used by the parser.

The parser is quite oblivious to the meaning of the individual input features; they are mostly all treated in the same way. For most purposes, they can thus be simply seen as named references to the columns in the CoNLL-U files. This makes it possible to conduct various experiments only by modifying the contents of the respective columns in the datasets and setting appropriate values for the options listed above, without the need to modify the internals of the parser.

Although we will review the parser settings in the experiment descriptions, we can already summarize the settings which we frequently use:

- For fully supervised monolingual parsing, we use the default settings.
- For cross-lingual parsing, we usually do not use the morphological features (`feats`), as these do not seem to typically port well across languages.
- For delexicalized cross-lingual parsing (Chapter 3), we also do not use the word forms (`form`); the parser then only relies on UPOS tags (`upostag`) and on the order of the tokens.
- For lexicalized cross-lingual parsing (Chapter 5), we pre-train word embeddings on the target language texts with word2vec [Mikolov et al., 2013] and provide them to the parser (`form_file`).
- For experiments with subselection of fine-grained morphological features (Section 3.2.1), we sometimes needed to provide two different features independently to the parser, which we achieved by constructing the input to contain one of them in the morphological features field (`feats`) and the other one in the language-specific tag field (`xpostag`).

[3] `http://ufal.mff.cuni.cz/udpipe/users-manual#model_training_parser`

2.3 Parser evaluation

In this section, we explain and thoroughly discuss the two main parser evaluation measures which we use throughout this work – Unlabelled Attachment Score (UAS) and Labelled Attachment Score (LAS). We also discuss specifics of parser evaluation on the UD datasets, and in the cross-lingual setting.

2.3.1 UAS and LAS

Dependency parsers are typically evaluated using two measures: Unlabelled Attachment Score (UAS), and Labelled Attachment Score (LAS). Both of these measures are simply token-level accuracies, taking into account all tokens in the test data, and giving each token an equal weight in the evaluation.

UAS [Eisner, 1996] only takes the tree structure into account, ignoring the dependency relation labels. Each token is considered correctly parsed if it is assigned the correct head; otherwise, it is considered to be parsed incorrectly. UAS is simply the proportion of correctly parsed tokens – e.g. if there are 10 tokens in the test data, and the parser attaches 8 of them to correct parents, its UAS is 80%.

LAS assesses both the assigned head as well as the assigned dependency relation label. While theoretically, the label expresses the relation between the head node and the dependent node, in practice, it is generally treated as belonging to the dependency node; thus, each node is assigned exactly one head and one dependency relation label. LAS then considers a node correctly parsed if it is assigned both the correct head and the correct dependency relation label; all incorrectly parsed nodes are treated equally, i.e. it does not matter whether only the head is incorrect, only the label is incorrect, or both are incorrect. As could be expected, the LAS of a parser output is the proportion of correctly parsed nodes under these criteria. LAS is typically used as the standard evaluation measure for dependency parsing, including e.g. parsing shared tasks [Buchholz and Marsi, 2006, Zeman et al., 2017].

In cross-lingual parsing, UAS has often been used as the standard evaluation measure, especially in the past; this is still true for unsupervised parsing, as it is not clear how the dependency relation labels should be devised in an unsupervised way. McDonald et al. [2013] are most probably the first who use LAS for evaluation of cross-lingual parsing, noting that with the release of GSD, where dependency relation labels are consistently annotated in the same way across all languages, reliable evaluation with LAS is finally possible. Tiedemann et al. [2014] and other researchers then follow this approach, using LAS as the primary evaluation measure.

From previous work, it seems that UAS was used instead of LAS mostly in the cases where the evaluated parser actually only produced unlabelled parse trees, thus making evaluation with LAS impossible. However, in the usual definition of the parsing task, the parser is expected to produce labelled dependency trees, which should not be hard to accomplish with the current state of available treebank resources and parsers. Therefore, we see no reason to use UAS instead of LAS for cross-lingual parser evaluation, and evaluate our parsers using LAS whenever possible in this thesis. In older experiments performed with the unlabelled

MSTperl parser, we only report UAS, as we did not perform the label assignment at that time. However, all the final best-performing setups were implemented or reimplemented using the labelled UDPipe parser, and we report LAS for all of them.

2.3.2 UD specifics

UD have an (at least) two-layer structure of dependency relation labels: a base label and potentially a language-specific subtype, as in `nsubj:expl` used for expletive subjects in French.[4] In some cases, especially in fully supervised monolingual parsing, it may be reasonable to use the full labels for evaluation. However, in the cross-lingual setting, it seems to make very little sense to use language-specific labels, as these will mostly be specific to either only the source or only the target language. Therefore, we omit the language-specific dependency relation label parts from our evaluations. (Actually, we go one step further, and eliminate the language-specific subtypes already in the treebanks.)

Furthermore, UD also allow multi-word tokens, multi-token words and elided tokens. However, we do not deal with these in our work, removing any such special tokens from our data.[5]

2.3.3 Other measures

In some works, sentence-level accuracies are also measured, i.e. the proportion of sentences that are parsed completely correctly; this measure is often denoted EM (exact match). While this may potentially be interesting for some high-accuracy parsers, in our setting even the token-level accuracies are considerably low, and we therefore do not even attempt to measure the sentence-level accuracies.

There are other possible flavours to dependency parser evaluation, the more common ones including:

- ignoring punctuation nodes,
- ignoring sentences longer than N tokens,
- ignoring the treeness requirement, i.e. accepting general single-headed oriented graphs,
- ignoring the edge direction, i.e. treating both an edge from parent to child and from child to parent as correct (undirected attachment score).[6]

While all of these can be regarded as sensible from some points of view and in some settings, we feel that in general they somewhat obfuscate the reported results, making them harder to understand and compare. We thus prefer not to use the alternative evaluation measures in our work.

[4] `http://universaldependencies.org/ext-feat-index.html`

[5] Simply put, we simplify the treebanks by removing any tokens which do have an integer ID as the value of the first field in the CoNLL-U file.

[6] This measure is usual in unsupervised dependency parsing [Klein and Manning, 2004].

2.3.4 Evaluation on under-resourced languages

In cross-lingual parser transfer research, this work not being and exception, parser transfer is typically not actually applied to truly under-resourced languages for which there are no treebanks, since there is no easy way of evaluating such experiments. We follow the usual way of using target languages for which there is a treebank available and thus the experiments can be easily evaluated, but we do not use the target treebank for training, thus simulating the under-resourcedness of the target language.

Still, as some of the target languages we use have only small treebanks available, they can already be considered somewhat under-resourced. Furthermore, the newest releases of UD even contain real under-resourced languages, for which only tiny test treebanks are available.

One also needs to always be careful about which POS tags to use in the target language. For research purposes, it may be interesting to apply the parser to target data labelled with manually-assigned gold tags, or with tags predicted by a tagger trained on the gold tags. However, in practice, we can hardly assume to have access to such POS tags for real under-resourced target languages. Therefore, we take care to ultimately evaluate our approaches using realistically obtained taggers; we specifically address cross-lingual tagger induction in Chapter 6.

The other option is to use no POS tags at all. While this is technically possible, as the parser can be trained using only word forms as input features, we found that in practice this leads to much worse results with the parsers which we use, and therefore do not follow that approach in this work.

3. Delexicalized Parser Transfer

In this chapter, we introduce our base approach to cross-lingual parsing, the single-source delexicalized parser transfer.

We explain the general idea of delexicalized parsing in Section 3.1. In Section 3.2, we particularly focus on the cross-lingual transfer of delexicalized parsers. And finally, in Section 3.3, we review our case study of the influence of treebank annotation style on delexicalized parser transfer.

3.1 Delexicalized parsing

A typical syntactic parser is *lexicalized*, i.e. it uses the individual word forms as input features. Usually, it also takes in POS tags as a useful abstraction over the individual words, which helps it generalize over rare words and rare contexts. However, the lexical features bind the parser tightly to the vocabulary appearing in the training data. This may already pose problems in monolingual parsing, e.g. if the training data is very small. However, it becomes a fundamental obstacle in cross-lingual parsing, where we intend to apply the parser to a different language, which, unless handled somehow, is bound to render the learned lexical features mostly or completely useless.[1]

One possible way out is to remove the lexical features from the parser, using only the POS tags (and potentially other morphological features, such as case and gender), thus obtaining a *delexicalized* parser. Parsing without lexical features is an approach taken already by the first grammar-based parsers, and still appearing in simpler parsing methods. However, in those cases, the starting point already is a parser that does not take lexical features, i.e. there was no explicit delexicalization taking place. In many cases, the lexical features were not used simply because the parser was unable to handle them efficiently, e.g. because manual creation of lexicalized grammar rules is an incredibly laborious task. However, we only talk about delexicalized parsing when one takes an already lexicalized parser, and then explicitly removes the lexical features from it. This method was, to the best of our knowledge, introduced by Zeman and Resnik [2008] for cross-lingual delexicalized parser transfer (see Section 3.2).

In the simplest case of delexicalized parsing, only coarse-grained POS tags, such as UPOS, are used as the input features. The morphological lemmas obviously need to be removed, since these are lexical features. With fine-grained morphological features, such as case, number, gender, or tense, the situation is less clear – they can all be kept, they can all be removed, or some of them may be kept and others removed, based on what seems to be most appropriate for the particular setting in which the delexicalized parser is applied. By default, we remove all of these features; we discuss that in more detail in Section 3.2.1.

Although the delexicalization of the parsing is usually described as removing the lexical information directly from the treebanks, in practice, we can simply set up the parser not to use the lexical features. E.g. for the UDPipe parser, which we use in most experiments, it is sufficient to set the dimension of the

[1] This problem becomes even more pronounced if the languages vary in the used alphabet.

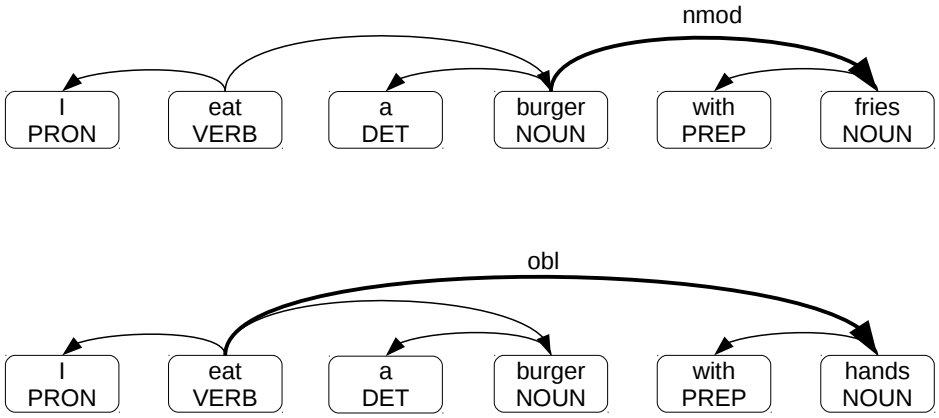

Figure 3.1: An example of a pair of sentences where lexical information is needed to parse them correctly – both share the same POS tags, but the syntactic structures differ.

word form embeddings to zero (by default, we do the same for the dimension of the embedding of the morphological features).[2]

The delexicalization is inevitably a lossy procedure. For some sentences, their syntactic structure can be easily determined even without the lexical information, just based on the POS tags. In other cases, stripping the lexical information introduces ambiguity, as the same sequence of POS tags can have multiple syntactic analyses.

See Figure 3.1 for an example of such a pair of sentences: "I eat a burger with fries" and "I eat a burger with hands". With lexical information present, none of the sentences is ambiguous, as fries would be an impractical tool to use for eating a burger, while hands would be a very peculiar condiment to serve with a burger. Yet, these two sentences happen to share the same sequence of UPOS tags:

PRON VERB DET NOUN PREP NOUN

Thus, the input to the syntactic parser becomes ambiguous once the lexical information is stripped; this is actually a common and well-known case of syntactic ambiguity, where the governing node for an adpositional phrase ("with hands") cannot be reliably determined (PP attachment ambiguity).

In such cases, a delexicalized parser will presumably either choose one of the possible analyses rather randomly, e.g. based on some irrelevant context features, or lean towards the one which is more frequent in the data. In any case, such ambiguities inevitably decrease the accuracy of delexicalized parsing, both in the monolingual and in the cross-lingual setting.

Table 3.1 compares the performances of lexicalized and delexicalized supervised monolingual dependency parsers on the UD 1.4 dataset. While the average difference is 8.8 LAS points, a lot of variance can be observed. Very small drops (or even an increase) in accuracy can be observed for languages with very small training treebanks, where the lexical features are too sparse and the generalizations offered by using POS tags are very useful. However, in general, the differences in the accuracies depend much more on the particular language than on the treebank size, as can be seen e.g. in the small drop for the large Indonesian

[2] `--parser='embedding_form=0;embedding_feats=0;'`

Target	Lexicalized	Delexicalized	Δ	TB (kT)
da	77.97	69.74	-8.23	89
el	77.86	66.53	-11.33	47
hu	78.16	65.51	-12.65	33
id	72.46	66.56	-5.90	98
ja	62.83	59.08	-3.75	80
kk	61.30	60.09	-1.21	5
lv	67.77	49.53	-18.24	13
pl	86.63	75.19	-11.44	69
sk	81.86	68.05	-13.81	81
ta	69.44	62.63	-6.81	6
tr	72.09	58.34	-13.75	41
uk	41.49	49.38	7.89	1
vi	61.73	46.74	-14.99	32
AVG	70.12	61.34	-8.79	46

Table 3.1: Comparison of LAS of lexicalized versus delexicalized supervised monolingual parsers on UD 1.4 target treebanks. Training treebank size in thousands of tokens is also listed.

treebank, while the largest drop is observed for the very small Latvian treebank.

On one hand, the scores of the delexicalized parsers are often somewhat competitive, suggesting that delexicalization might be justified if it enables us to use a cross-lingual parser transfer approach which we would otherwise be unable to use, as we will show in Section 3.2. On the other hand, the observed drops are large enough to motivate approaches that enable us to use lexicalized parsers even in the cross-lingual setting, which we investigate in Chapter 5.

3.2 Delexicalized parser transfer

In this thesis, we take the single-source delexicalized parser transfer as our base approach to cross-lingual parsing, upon which we build our methods. The method was introduced by Zeman and Resnik [2008], who trained a delexicalized parser on a Danish treebank and evaluated it on a Swedish one.[3] They note that while the lexicons of these two languages will most probably differ significantly even if they are very close, they may share both many morphological as well as syntactic properties, which motivates their approach.

As they focus on a pair of very close languages, they try to utilize as rich morphological annotation as possible, based on the Interset tag conversions (Section 1.2.2). In our work, however, we also focus on more distant language pairs, which is why we only use the coarse POS tags in our base approach, only selectively reintroducing some of the morphological features where appropriate (see Section 3.2.1).

While the intended use-case is the syntactic analysis of an under-resourced target language using a resource-rich source language, the authors take the usual

[3]The authors actually used phrase-structure parsing, even though they operate on dependency-based CoNLL treebanks, which they first converted into the PennTB-style phrase-structure trees. However, the key principle of their method is easily applicable both to dependency-based and constituency-based parsing.

Target		Source		Source-target transfer	
language	LAS	language	LAS	languages	LAS
da	69.74	sv	65.03	sv→da	59.01
el	66.53	bg	76.98	bg→el	53.09
hu	65.51	sv	65.03	sv→hu	46.14
id	66.56	pt	70.21	pt→id	45.56
ja	59.08	fi	54.62	fi→ja	34.74
kk	60.09	hi	72.63	hi→kk	29.67
lv	49.53	fi	54.62	fi→lv	41.07
pl	75.19	cs	67.58	cs→pl	66.34
sk	68.05	cs	67.58	cs→sk	63.71
ta	62.63	hi	72.63	hi→ta	37.69
tr	58.34	fi	54.62	fi→tr	31.21
uk	49.38	ru	70.38	ru→uk	55.19
vi	46.74	cs	67.58	cs→vi	29.69
AVG	61.34		66.11		45.62

Table 3.2: LAS of source and target supervised monolingual delexicalized parsers and of cross-lingual delexicalized source-target delexicalized parser transfer, i.e. target data parsed by the source parser. Gold POS tags are used here for both source and target.

approach of simulating this situation by evaluating on a treebank for a resource-rich language, as we already explained in Section 2.3.4 – while there is a Swedish treebank available, its syntactic annotation is only used by the authors to be able to evaluate their method.

However, the same cannot be said for the target morphological annotation, as this is used as the *input* for the parser, and must thus be available even for the assumedly under-resourced language. While we use the gold morphological annotation from the target treebank in our experiments, we note that the assumption of its availability is unrealistic, and therefore explore methods for cross-lingual POS tag induction in Chapter 6. Nevertheless, at this point, we simply note that such methods exist and perform quite well even in restricted settings. Therefore, while the absolute accuracies reported for cross-lingual parsing using supervised target POS tags are unrealistically high, the methods themselves are applicable even with cross-lingually induced POS tags, and their relative comparisons remain similar under such settings, as will be shown later. Thus, we feel confident applying and evaluating our methods on harmonized supervised target language POS tags for research purposes.

In Table 3.2, we compare the LAS of supervised monolingual delexicalized parsers, trained on source and target treebanks, and a delexicalized parser transfer, i.e. the source parser applied to the target data and only evaluated on the target treebank. The question of how to select an appropriate source language to use for each target language is an important one, and we devote a significant part of Chapter 4 to that issue.

As could be expected, the cross-lingual transfer of the parser typically leads to a huge drop in parsing accuracy. For very similar language pairs, such as Swedish-Danish or Czech-Slovak, the drop is typically not very dramatic. On the other hand, for dissimilar language pairs, the drop can be devastating, as in

Hindi-Kazakh or Finnish-Japanese; the scores are most probably low enough to render the resulting parser completely useless. This shows the need to carefully select the source language (or even combine more of them), as will be discussed in Chapter 4. However, we can admit upfront that for a target language very dissimilar to any of the available source languages, we have not been able to achieve reasonable parsing accuracies by any means.

Interestingly, for one target language in our dataset – Ukrainian – the transferred parser actually performs *better* than the supervised monolingual target parser. This is because of a combination of three factors: the Ukrainian training data being very small (only 1200 tokens), the accuracy of the source (Russian) parser being very high, and the two languages being very close. This already shows the power of cross-lingual parser transfer – Ukrainian is clearly an under-resourced language here, and even the base cross-lingual approach already delivers more accurate parsing than the monolingual parser.

3.2.1 Using fine-grained morphological features

In most of our experiments, we only use the coarse-grained UPOS tags, removing all of the more fine-grained morphological features. This is mainly motivated by the fact that we found this approach to generally lead to better results than keeping all of the features. It should be noted here that both MSTperl and UDPipe do not perform any clever processing of the features by default, taking the whole string specifying the feature values as an atomic unit. This presumably leads to considerable data sparseness, as well as insufficient generalization over the feature values by the parsers.

In [Rosa et al., 2017], we investigated the possibility of subselecting only some of the features to keep, according to what seems to be shared among the source and target languages. These experiments were conducted on the VarDial dataset, with the source and target languages being very close and morphologically rich. We achieved moderate improvements (around +1.5% LAS) by using only the *case* feature and discarding all other features. This was partially motivated by a similar approach of Collins et al. [1999], who found the morphological case to be a very informative feature for monolingual parsing of morphologically rich languages (Czech in particular).

However, we have not been able to substantially improve upon that result, neither by trying to add other individual features, alone or in conjunction, nor by trying to empirically measure the level of cross-lingual sharedness of the features on parallel data.[4]

Note that even though we used supervised source and target taggers in these particular experiments, which may lead to problems due to differences in morphological annotation across the languages, we also investigated and eventually used cross-lingual tagging in that work, still not observing any larger gains in parsing accuracy.

Moreover, as the notion of a morphological case is by no means shared across all languages, we have then abandoned this approach altogether in further experiments, reverting to not using any other morphological annotation than UPOS.

[4]The latter experiments were conducted by David Mareček.

This presumably makes our approach more sensible for source and target languages with a lower degree of grammatical similarity.

Interestingly, our observations are in stark contrast to those of Agić et al. [2014], who reported large and consistent gains in cross-lingual parsing accuracy thanks to using rich morphological annotation. Although the authors used a different setup and different languages, which may explain some of the differences in their and our findings, we believe that the issue of usefulness of fine-grained morphological features for cross-lingual parsing clearly deserves further exploration in future.

3.3 Case study of annotation style learnability

UD have become the *de facto* standard for treebank annotation in the last years, featuring, among other, the attachment of function words (adpositions, determiners, etc.) as leaves. However, it has been argued that Prague style trees are easier to obtain by using statistical parsers. Among other differences, adpositions (i.e. prepositions and postpositions) provide important cues to the parser for adpositional group attachment, which is one of the most notorious parsing problems. This information may become harder for the parser to access when the adpositions are annotated as leaves.

The UD annotation scheme has been intentionally designed with the idea that linguistic adequacy or usefulness of the annotation is more important than suitability for a parser. This eventually led to the fact that the selected ways of annotating some structures may be difficult for a parser to produce. The approach suggested by de Marneffe et al. [2014] was to use a different annotation style for parsing, with Prague style adposition annotation, among other, and to automatically convert the dependency trees to full Stanford style only after parsing. The original paper even suggested that an official parser-friendly version of USD, easy to convert to and from standard USD, may be defined in future, although, to the best of our knowledge, this has not materialized. While any automatic conversions are usually noisy[5] to some extent,[6] they have been frequently used in parsing, as we have reviewed in Chapter 1.

The issue of dependency representation learnability has been studied by several authors, who generally came to similar conclusions [Schwartz et al., 2012, Søgaard, 2013, Ivanova et al., 2013]. Notably, all of the authors have found the

[5]In this thesis, when we say that something is *noisy*, what we mean is that it is somewhat inaccurate, imprecise or biased, it does not always give the correct result, it cannot be fully relied on, thus introducing some additional erroneousness (*noise*) into the processing pipeline. We thus use the words *noise*, *noisy* in a broad sense, regardless of where the "noise" really comes from, and to what extent it is random or systematic. We will thus venture to denote a set of rules as *noisy*, even if this may sound as a contradiction, since the behaviour of rules is, by definition, fully predictable: as we are operating in the immensely variable realm of natural languages, even simple rules applied to our data can lead to very unexpected outcomes, which we may perceive as noise.

[6]A 100% accurate and reversible conversion is theoretically possible if both of the representations capture exactly the same information. However, practice has shown that this is nearly never the case, as even representations of simple and regular phenomena typically get murky once they interact with coordinations, nested coordinations, ellipses, multiword expressions, etc.

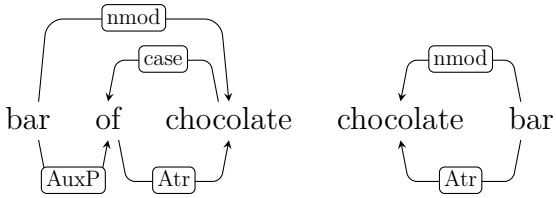

Figure 3.2: Stanford style (above) and Prague style (below) analysis of the phrases "bar of chocolate" and "chocolate bar". Note that in Stanford style, these phrases have a more similar structure, both featuring an *nmod* edge directly from "bar" to "chocolate". This shows the principle of constructions with a similar meaning also having a similar dependency structure.

Prague-style adposition-as-head easier to learn for the parser than the Stanford-style adposition-as-leaf. Naturally, the experiments were performed with standard monolingual supervised parsers.

In multilingual and cross-lingual parsing scenarios, the higher cross-lingual similarity of USD/UD dependency trees may be of benefit. From all of the differences between Prague and Stanford, the adposition attachment seems to be the most interesting, as adpositions are usually very frequent and diverse in languages, as well as very important in parsing.

In this section, adapted from [Rosa, 2015c], we investigate this problem empirically, by evaluating the influence of adposition annotation style in cross-lingual multi-source delexicalized parser transfer. Specifically, we use the HamleDT 2.0 collection and the MSTperl parser to evaluate the potential benefit of employing Stanford style adposition attachment instead of the Prague style in parsing.

3.3.1 Prague versus Stanford

One of the prominent features of Stanford style dependencies is their approach to function words. The general rule is that all function words, such as adpositions or conjunctions, are attached as leaf nodes. This is a result of a standpoint which favours direct dependency relations between lexical nodes, not mediated by function words. This also makes dependency structures more similar cross-lingually, as it is very common that the same function is expressed by an adposition in one language, but by other means, such as morphology or word order, in another language – or even within the same language, as shown in Figure 3.2. On the other hand, the Prague style dependencies annotate adpositions as heads of adpositional groups.[7]

3.3.2 Automatic conversions

We use the HamleDT 2.0 collection of 30 dependency treebanks, which had been semi-automatically harmonized to Prague dependencies and then Stanfordized

[7]The lexical nodes are only directly connected in Prague tectogrammatical (deep-syntax) dependency trees, where function words are removed and their functions are captured via node attributes. It is worth noting that in general, there is little difference between representing information by means of node attributes or leaf nodes; thus, Stanford trees and Prague tectogrammatical trees are actually very similar in structure. The similarity becomes even greater with the now-emerging Enhanced UD [Schuster and Manning, 2016].

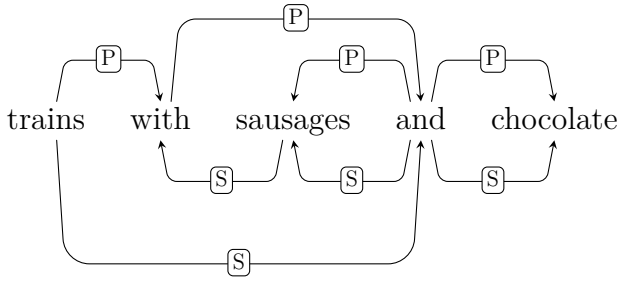

Figure 3.3: Original Prague style adposition analysis (above), and Stanford style adposition analysis as produced by the conversion (below). Note that the coordination stays in the Prague style. Edge labels are not shown as we do not use them in this work.

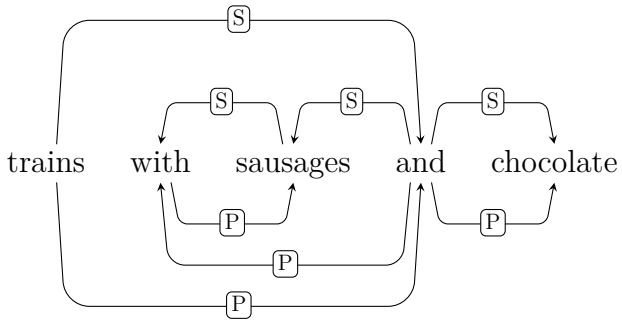

Figure 3.4: Stanford style adposition analysis (above), and Prague style adposition analysis as produced by the conversion (below). Together with Figure 3.3, this shows a case where our conversion is imperfect, as we are unable to obtain the original structure after the conversion roundtrip.

into USD. The treebanks contained in the dataset have been presented in Table 1.1.

In these experiments, we use the Prague style version of HamleDT as the basis; instead of using the full Stanford version, we only focus on one of its prominent features – adposition attachment. Thus, we alternate between Prague adposition attachment as head (denoted "P"), and Stanford adposition attachment as leaf node (denoted "S"), using simple conversion scripts.

- The conversion from P to S takes each adposition and attaches it as a dependent of its left-most non-adpositional child, together with all of its other non-adpositional children. Thus, the adposition becomes a leaf node, unless it has adpositional dependent nodes (typically this signifies a compound adposition). Coordinating conjunctions are passed through (recursively) – if the left-most non-adpositional child is a coordinating conjunction, then its dependent leftmost non-adpositional conjunct is used instead as the new head of the adposition (see Figure 3.3).

- In the conversion from S to P, each adposition with a non-adpositional head is attached as a dependent of its head's head, and its original head is attached as its dependent (see Figure 3.4).

The roundtrip accuracy of the conversion (UAS after converting from P to S and back) is around 98% in total, and around 94% for adposition nodes alone.

Setup	Lex	Delex	Transfer
Prague	80.54	74.12	56.68
Stanford full	76.47	69.53	48.91
Prague non-punct	80.23	74.00	56.08
Stanford full non-punct	76.84	70.66	50.15

Table 3.3: Prague versus full Stanford annotation style, UAS averaged over 30 target languages.
The *Lex*icalized and *Delex*icalized parsers are monolingual.
The *Transfer* parser is a combination of 29 sources parsers applied to the remaining target language.

3.3.3 Experiment setup

For the multi-source cross-lingual parser transfer, we use an unweighted parse tree combination approach in the style of Sagae and Lavie [2006], which we will describe in more detail in Section 4.3.1. The general setup is as follows. One of the 30 treebanks is taken as the target treebank, and the remaining 29 treebanks become source treebanks. Then, delexicalized parsers are trained on the source treebanks, resulting in 29 trained parser models. Next, each of the parsers is applied to each sentence in the test section of the target treebank. And finally, the obtained parse trees for each sentence are combined together using the Chu-Liu-Edmonds MST algorithm, and the resulting dependency tree is evaluated against the target treebank annotation.

Note that there are three places where a conversion from one annotation style to another may take place – conversion of the source treebank before training a parser, conversion of the parser output before the parse tree combination, and conversion of the parse tree combination output. We will denote the setups using the pattern"X/Y/Z", where "X" denotes the annotation style used for parser training and parsing, "Y" is the style into which the parser outputs are converted before being combined, and "Z" is the style into which the result of the combination is converted. Furthermore, "P,S/Y/Z" will refer to parsing both with "P" style parsers and "S" style parsers, thus resulting in 58 trees for each sentence to combine, rather than 29. In many setups, there is no conversion after the combination, or there is even no combination performed (in monolingual setups); therefore, we will often omit the last part of the pattern, using only "X/Y".

3.3.4 Full Universal Stanford Dependencies

As a preliminary experiment, we compared the Prague version with its fully Stanfordized version. The results are shown in Table 3.3. It can be seen that the Stanford version performs much worse than the Prague one – its results are lower by around 5 UAS points.

Closer inspection showed that many of the errors are actually due to sentence-final punctuation attachment. In Stanford style, sentence-final punctuation is to be attached as a dependent node of the root node of the sentence (typically the main predicate). However, this is difficult for the first-order parser, as it has no knowledge of the root node when scoring the potential edges, and thus the punctuation gets often attached to some other verb. In Prague style, the

Setup	Lexicalized	Delexicalized
P/P	**80.54**	**74.12**
S/P	78.44	72.65
S/S	79.77	73.91
P/S	80.23	73.94

Table 3.4: Average UAS of supervised monolingual parsers, both lexicalized and delexicalized.

sentence-final punctuation is attached to the technical root node, which is marked by special values of the node features, and thus the assignment is very easy to make. While this is an important point to keep in mind when parsing into full Stanford style, it is of little relevance to the goal of this work – punctuation attachment is rarely important in NLP applications, and is not very likely to substantially contribute to cross-lingual dependency structure similarity either. For this reason, we also include UAS measured only on non-punctuation nodes. Still, adposition attachment, which we are mostly interested in, accounts for only a part of the score difference.

3.3.5 Prague versus Stanford adpositions

Further on, we only use the Prague style annotation of the treebanks, with adpositions annotated either in Prague style (P) or Stanford style (S).

Supervised parsers

We first evaluate supervised monolingual lexicalized and delexicalized parsers, alternating between the P and S annotation styles of adpositions. The results in Table 3.4 show that in the lexicalized setting, the UAS of the P style parser is +0.77 above the S style parser. Actually, to obtain S style parse trees, it is better to parse the text using a parser trained on a P style treebank, and then convert the output parse trees (this yields a +0.46 higher UAS than parsing directly using an S style parser). Here, the adpositions clearly provide important information to the parser, and their annotation as heads benefits the results.

In the delexicalized setting, the P style parser scores higher than the S style one only by a small margin (+0.21 UAS). Moreover, parsing directly using the S style is now comparable to parsing using P style and then converting to S style. This suggests that the most important piece of information for correctly attaching an adposition is its lemma (or word form), and delexicalizing a parser thus reduces the advantage of P style annotation for correct adposition attachment.

29-to-1 delexicalized parser transfer

We now move on to the main focus of our work, evaluating the effect of adposition annotation style in cross-lingual transfer of 29 delexicalized source parsers to a target language using parse tree combination.

Table 3.5 shows that using either P or S for everything leads to comparable results, with the S style now achieving a slightly better score (+0.20 UAS on average). The results tend to get worse when additional conversions are performed; we thus omit such setups from further evaluation. Interestingly, slight

P style output		S style output	
Setup	UAS	Setup	UAS
P/P/P	56.68	S/S/S	56.88
P/S/P	55.43	S/P/S	56.31
S/S/P	55.51	P/P/S	56.48
S/P/P	55.84	P/S/S	56.80
P,S/P/P	**56.81**	P,S/S/S	**57.07**
P,S/S/P	55.71	P,S/P/S	56.67

Table 3.5: Average UAS of various setups of delexicalized parser transfer, always using 1 language as target and the remaining 29 languages as source. The setup details the annotation style (Prague or Stanford) used for training/combining/evaluating the parsers.

improvements can be obtained by applying both P parsers and S parsers and combining them after conversion of the resulting trees to the S style, achieving a total average increase of +0.39 UAS absolute over the P style baseline.

Table 3.6 shows detailed results of the better-performing transfer setups for all target languages, together with the results of the supervised monolingual methods. We can see that employing the S style annotation in the cross-lingual transfer leads to comparable or slightly better results. This indicates a potential benefit of the S style annotation in a cross-lingual setting, presumably due to the increased similarity of the dependency structures across languages.

Smaller source treebank subsets

For a deeper insight and further confirmation of our findings, we also performed a set of experiments with smaller 3-member subsets of the treebank collection. We selected several treebank groups, based on the ratio of adposition tokens to all tokens. We also only chose large enough treebanks (more than 100,000 tokens). The subsets are listed in Table 3.7; we also used a larger "All9" set of all the 9 selected treebanks. Only these were then used for training; the remaining 21 languages were used for testing as target languages.

The summary results are to be found in Table 3.8. For these datasets, the advantage of employing the S style in combination with P style becomes clearly visible, frequently leading to better results than when using only the P style. Moreover, converting the parse trees to S style before combining them is also often better than converting them to P style. The improvements are large especially for the High, Low and Mix datasets. This suggests that the role of Stanford style is stronger with small and highly diverse datasets, where its benefit of making the dependency trees more similar becomes rather important.[8] For the High dataset, the best result surpasses the Prague-only baseline by as much as +2.24 UAS absolute on average, yielding a better result for 19 of the 21 target languages.

[8]Of course, this is only a speculation, as there are many other properties of the source treebank subsets which we were unable to factor out that may influence the results – e.g. the High and Low subsets contain genealogically highly varied languages, but we were unable to find such a varied subset among the languages with a medium frequency of adpositions.

Tgt	Delexicalized supervised			Delexicalized transfer			
lang	P/P	S/S	P/S	P/P	P,S/P	S/S	P,S/S
ar	**69.61**	69.29	69.50	44.61	**44.99**	43.16	44.13
bg	**83.87**	82.76	83.32	**73.17**	72.72	72.24	72.65
bn	77.59	**78.82**	77.59	59.98	60.34	**60.47**	60.22
ca	**79.71**	79.03	79.33	**66.45**	66.38	65.61	66.07
cs	**70.99**	70.69	70.69	64.06	**64.14**	63.62	63.93
da	**81.13**	80.31	80.67	**63.74**	63.53	62.82	63.09
de	**77.52**	76.92	77.47	52.58	55.17	**55.95**	55.32
el	**75.40**	75.15	74.73	67.05	67.69	67.63	**67.78**
en	**76.57**	76.19	76.03	46.13	**48.23**	47.65	47.09
es	**79.75**	78.52	79.25	**69.73**	69.61	68.85	69.17
et	80.96	**82.85**	80.75	71.34	72.07	74.06	**74.48**
eu	68.34	**68.41**	68.34	46.12	45.92	**46.15**	46.07
fa	70.44	**71.72**	70.78	54.69	54.77	56.41	**56.69**
fi	63.10	62.51	**63.13**	**51.48**	51.17	50.60	51.08
grc	48.92	**49.10**	48.80	46.24	46.38	46.48	**46.50**
hi	**80.55**	80.52	80.52	30.12	29.64	33.23	**33.64**
hu	**72.54**	71.79	72.34	59.68	59.89	60.50	**60.81**
it	**77.49**	76.57	76.92	64.52	**65.13**	64.44	64.50
ja	81.72	84.03	**84.35**	44.23	42.64	44.02	**44.88**
la	44.08	**44.12**	43.81	41.14	41.28	41.34	**41.47**
nl	**74.02**	73.70	73.57	62.47	62.04	**63.81**	63.80
pt	**80.14**	78.68	79.77	71.35	**71.60**	71.14	71.26
ro	85.19	**85.34**	84.85	59.66	**59.85**	58.52	58.67
ru	**73.08**	72.70	72.90	**63.82**	63.65	62.43	63.13
sk	**71.38**	70.88	70.93	63.66	**63.73**	63.36	63.62
sl	72.91	**72.93**	72.69	**54.40**	53.68	53.80	53.68
sv	**78.84**	77.97	78.18	62.08	62.18	**62.22**	61.60
ta	**68.17**	67.92	67.62	38.76	**39.01**	37.66	38.91
te	**85.59**	84.09	85.59	66.83	66.16	**67.00**	66.50
tr	**73.99**	73.72	73.72	40.39	40.82	**41.28**	41.26
Avg	**74.12**	73.91	73.94	56.68	56.81	56.88	**57.07**

Table 3.6: UAS of supervised delexicalized monolingual parsers and delexicalized transfer parsers.

Subset	ADP freq.	Language
High	15%	Spanish
	19%	Hindi
	19%	Japanese
Med	9%	Czech
	8%	English
	9%	Swedish
Low	0%	Basque
	4%	Ancient Greek
	1%	Hungarian
Mix	15%	Spanish
	9%	Swedish
	1%	Hungarian

Table 3.7: Subsets of source treebanks, selected according to their frequency of adposition tokens.

Setup	High	Med	Low	Mix	All9
P/P	40.53	52.00	44.53	41.03	54.98
P,S/P	41.29	52.57	45.00	41.75	55.37
S/S	41.36	51.64	43.69	41.95	54.85
P,S/S	**42.77**	**52.67**	**46.41**	**42.66**	**55.42**

Table 3.8: UAS of delexicalized parser transfer, averaged over 21 target languages, with the specified subset treebanks as sources.

3.3.6 Summary

In this section, we investigated the usefulness of Stanford adposition attachment style as an alternative to the Prague style in dependency parsing, using a large set of 30 treebanks for evaluation. We especially focused on an evaluation in multi-source cross-lingual delexicalized parser transfer, as one of the targets behind the design of USD and UD is to be more cross-lingually consistent than other annotation styles.

We managed to confirm that for supervised parsing, Prague annotation style is favorable over Stanford style, as has been already stated in literature. However, in the parser transfer setting, Stanford style adposition attachment tends to perform better than the Prague style, presumably thanks to its abstraction from the high interlingual variance in adposition usage.

Best results are achieved by combining outputs of parsers trained on treebanks of both Prague and Stanford adposition attachment style, reaching an average improvement of +0.39 UAS absolute over the Prague style baseline. Our results are further confirmed by experiments using smaller and more diverse subsets of training treebanks, where the advantage of combining Prague and Stanford adposition annotation style becomes even more pronounced, reaching average improvements of up to +2.24 UAS absolute over the Prague style baseline.

However, we have not further focused on that path in our research. In the further chapters, we thus only use Stanford-style annotations for all experiments, employing either the Stanfordized HamleDT 2.0 treebanks, or the UD 1.4 treebanks.

We would also like to note that we only used the first-order non-projective MSTperl parser in all experiments reported in this section; therefore, our conclusions may only be valid for that parser.

4. Using Multiple Sources

Multiple potential source treebanks for resource-rich languages are usually available, and it is non-trivial to select the best one for a given target language. While mostly ignored at first, the problem became rather clear with more and more treebanks becoming available, and researchers in cross-lingual parsing have made various attempts at solving it (see Section 4.1), including both methods of selecting one best source to use, as well as combining multiple sources.

The key part of this chapter, as well as of the whole thesis, is our attempt at solving this problem, using a designated language similarity measure, and a refurbished method for parser combination.

In Section 4.2, we introduce KL_{cpos^3}, our language similarity measure based on Kullback-Leibler divergence of probability distributions of coarse POS tag trigrams, estimated from POS-tagged corpora for the source and target languages. The measure has been designed and tuned specifically for multilingual delexicalized parser transfer, to be used both to select the most similar source language for a given target language, as well as to assign weights to multiple source languages in a multi-source combination. We developed two alternative approaches for performing such a combination: parse tree combination, described in Section 4.3.1, and parser model interpolation, introduced in Section 4.3.2.

In Section 4.4, we first evaluate the measure in the setting for which it was originally designed and tuned, namely delexicalized parser transfer using gold target POS tags. We show that in a single-source setting, KL_{cpos^3} often succeeds in selecting the best available source treebank for a given target language, or, in many other cases, selects a different but competitive one. In the multi-source parse tree combination approach, KL_{cpos^3} is often able to appropriately weight the available source treebanks, so that their weighted combination outperforms an unweighted one in many cases as well as on average.

Interestingly, KL_{cpos^3} has also been shown to perform well in various modifications of the original setting, for which it was not originally designed or tuned. It stays accurate when computed on cross-lingually induced POS instead of gold ones, as shown in Section 4.4.2 and independently confirmed by Agić [2017], who also shows it to outperform other language similarity measures in that setting. It is also successful in lexicalized parsing instead of delexicalized (Chapter 5), and even when applied to cross-lingual POS tagging instead of parsing (Chapter 6).

Thus, eventually, our ultimate best-performing setup successfully uses KL_{cpos^3} at several places. We therefore consider KL_{cpos^3} to be the key component of our approach, and the most important invention presented in this thesis.

Some parts of this chapter are adapted from [Rosa and Žabokrtský, 2015a] and [Rosa and Žabokrtský, 2015b].

4.1 The problem, and previous approaches to it

In cross-lingual parsing, we need to transfer the knowledge of a source language to a target language. While it has been always understood that the source language should be similar to the target language, this is easier said than done. Moreover, with the current explosion of the pool of easily available treebanks within the UD

project, with 102 treebanks for 60 languages currently being available, the choice is huge and thus even harder to make.

Various approaches have been used in the past, both to select the one best source treebank, as well as to use multiple (or all) of the available treebanks combined.

4.1.1 Ignoring the problem

Many works on cross-lingual parsing do not explicitly address this problem, either pre-selecting a fixed source language as in [Zeman and Resnik, 2008], or trying each of the available source languages and then choosing the best performing one. However, as already noted by McDonald et al. [2011], in the intended real-world scenario, one does not have access to information about the accuracy of the cross-lingual parsing on the target language, and thus cannot use it to select the source language.

Oracle treebank

The common approach, especially in earlier works and with a smaller number of available source treebanks, was to simply try one source after another, and report all of the scores, usually emphasizing the highest achieved score as *the* result. Such an approach does indeed truthfully assess the potential of the cross-lingual parser transfer, showing the highest accuracy *achievable* by a single-source transfer.

However, one should always bear in mind that the evaluation of the method on annotated target language treebanks is only a proxy evaluation, as in the intended scenario, the methods are applied to under-resourced languages, where such an evaluation is impossible. Consequently, one cannot use the evaluation on a target treebank as a method of choosing the source treebank to use, as assuming the availability of such information is overly optimistic.

Therefore, we refer to a source treebank chosen in that way as the *oracle* source treebank, and consider the transfer from the oracle source treebank to the target language as an upper bound approach.

4.1.2 Treebank concatenation

The first work to explicitly acknowledge this problem is that of McDonald et al. [2011], who applied delexicalized cross-lingual parser transfer in a setting with multiple source treebanks available, finding that a treebank for a language that is typologically close to the target language is typically a good choice for the source treebank, but noting that the problem of selecting the best source treebank without access to a target language treebank for evaluation is non-trivial.

As a work-around, the authors propose a simple resource combination method, concatenating all of the available source treebanks into one large multilingual treebank, and training one multilingual delexicalized parser on it, thus effectively eliminating the problem of having multiple source treebanks. This leads to better results than the average over individual single-source parsers, but worse than using only the oracle single-source parser.

The method is a rather simple one, and has several obvious drawbacks. Most importantly, it does not address the language similarity at all, using the same source combination for all target languages, which may be appropriate for some of them but less appropriate for other. It also does not deal with the fact that the source treebanks typically vary in size, which leads to the larger treebanks having more influence on the results. Moreover, it is also quite impractical, as training the parser on the large treebank concatenation can be computationally very demanding.

It is worth noting that we are unaware of any source-selection or source-weighting method that would explicitly take source treebank sizes into account, even though it is clear that these have an important influence on the results, as demonstrated by Tiedemann and Agić [2016]. To the best of our knowledge, all existing methods either explicitly ignore the treebank sizes (either by giving the same power to all sources, or even by cropping the larger source treebanks), or behave similarly to the base treebank concatenation method by implicitly giving more power to larger treebanks without any further normalization or a discussion of the appropriateness of such an approach. We believe this shortcoming of the current approaches to constitute a promising path for future research.

4.1.3 Using the World Atlas of Language Structures

We have introduced the WALS typological database in Section 1.5.2.

If one focuses only on one or a few languages, it is reasonable to look these up in the database manually to investigate their properties. In some scenarios, this may be sufficient to choose the appropriate approach for handling such a language. However, since our field of study tries to focus on many languages at once, automatic processing is a much more palatable option.

The work of Naseem et al. [2012] is the first known to us to explicitly use WALS features in cross-lingual parsing, followed by Søgaard and Wulff [2012] and Täckström et al. [2013b]. These authors focused on shared genealogical classification of the languages and on shared values of important word-order features to estimate source-target language similarity.

An interesting approach appearing in the cited works is selective enabling or disabling of the sensitivity of the parser to word order, based on matches or mismatches in the relevant WALS feature values. In particular, the authors note that for a target language which is rather dissimilar to any of the source languages, delexicalized transfer achieves better results if word order is completely or selectively ignored, in the sense that the parser only looks at the absolute distances of words but not their order. This is motivated by the observation that languages, at least when being observed only through POS, become more similar if we disregard word order. However, our preliminary experiments in that direction did not show much potential there.

Another possible way of employing the word-order features in WALS is by using pre- or post-reordering, thus forcibly making the source language word order more similar to the target one (or the other way round), as done by Aufrant et al. [2016]; such approaches are well known in MT [Xia and McCord, 2004, Collins et al., 2005, Lerner and Petrov, 2013, Hardmeier et al., 2014].

Recently, Agić [2017] used WALS in a similar but simpler way, estimating

the similarity of a pair of languages by computing the Hamming loss on all of the available WALS features. We reimplement and investigate this approach in our work in Section 4.4.2. The approach bears a striking similarity to the work of Duong et al. [2015], who seem to have used an identical approach, but only limiting themselves to WALS features capturing word order, and only using the estimated similarity for source selection, not source weighting.[1]

An interesting and partially related method was devised by Berg-Kirkpatrick and Klein [2010], who used information on language phylogeny to improve unsupervised parsing [Klein and Manning, 2004] by introducing a phylogenetic prior to couple languages based on their phylogenetic similarity.

4.1.4 Looking at part-of-speech tags

Søgaard [2011] trains a POS language model on a target POS-tagged corpus, and uses it to filter the source treebank, keeping only sentences that look like target language sentences to the language model. This decreases the size of the training data, but can still lead to improvements in the parsing accuracies if the threshold is set appropriately.

To the best of our knowledge, our approach of measuring language similarity using the KL divergence [Kullback and Leibler, 1951] of distributions of POS trigrams, which we first presented in [Rosa and Žabokrtský, 2015b], is novel. Interestingly, Duong et al. [2015] seem to have independently developed a very similar method, comparing distributions of POS n-grams of lengths 1 to 6 with the Jensen-Shannon (JS) divergence [Lee, 2001], which is a symmetrization of KL divergence. We compared the performances of KL divergence, JS divergence, and cosine similarity already in our original work, finding the KL divergence to perform best; we review these experiments in Section 4.2.5.

Our work also bears some similarity to Plank and Van Noord [2011], who apply KL divergence in a slightly different (and monolingual) setting. Although the authors mention individual POS tags (not POS n-grams) as one of the potential features whose distributions could be compared with KL divergence, they eventually settle for only relying on plain texts, noting that the reliance on POS tags would introduce additional noise. We review this work in Section 4.1.5.

4.1.5 Looking at words and characters

Interestingly, using POS tags for estimating language similarity seems to predate employment of words and characters for the same purpose in cross-lingual parsing. This is probably due to the fact that most early approaches assumed the availability of POS tags in both the source and the target, thus making them a readily available and easy-to-use feature. As has been already explained in Section 3.1, harmonized POS tags have been repeatedly shown to be a great multilingual abstraction over the texts in various languages, while working directly

[1] Duong et al. [2015] use cosine similarity, while Agić [2017] uses Hamming loss. As they apply it to binary vectors, we would expect the results to be identical, although this depends on the exact implementation they use, especially regarding empty feature values (which are plentiful in WALS). Unfortunately, none of the authors describe their approach in sufficient detail.

with word forms is trickier, and presumably can be expected to perform well only for lexically similar languages.

Estimating domain similarity

A problem somewhat similar to measuring language similarity is measuring domain similarity; the task of cross-lingual parser transfer could thus be regarded as a specific case of domain adaptation [McClosky et al., 2006, 2010]

Several similarity measures have been evaluated by Van Asch and Daelemans [2010] for monolingual domain adaptation of POS tagging, and Plank and Van Noord [2011] extended this work to other tasks, including monolingual cross-domain parser transfer. One of the investigated measures was the KL divergence measured on probability distributions of relevant features estimated from the datasets. Van Asch and Daelemans [2010] only compute the similarity measures on relative frequencies of words, while Plank and Van Noord [2011] extend this by alternatively using character tetragrams. They also investigate other measures, including JS divergence, Manhattan distance, and cosine distance.

Plank and Van Noord [2011] also suggest using POS tags, but dismiss them as introducing additional noise. While this is probably reasonable in a monolingual setting, the situation is reversed in the cross-lingual setting, where we cannot rely much on finding identical words or character n-grams across different languages, making the POS tags a more appealing abstraction, even if somewhat noisy.

Using a language identification tool

Agić [2017] takes a similar approach, but rather than developing the comparison method from scratch, they utilize an already existing tool, although originally designed for a slightly different purpose – the `langid.py` tool of Lui and Baldwin [2012].[2]

The `langid.py` tool is a state-of-the-art solution for language identification, i.e. the task of determining in which language a given piece of text is written. The tool is based on detecting character n-grams (from unigrams to tetragrams) characteristic for each language. Rather than simply comparing the n-gram distributions, it takes a multi-step approach in estimating the informativeness of each n-gram for language detection, as the authors note that some n-grams seem to characterize the topic or domain of the text rather than its language. Eventually, the decision about the most likely language of the input piece of text is performed by a Naive Bayes classifier.[3] The `langid.py` tool comes pre-trained to detect approximately 100 languages, but can be trained on monolingual plaintext data to detect other languages as well. There is also a possible setting of the tool to restrict the detection only to a given set of languages.

Agić [2017] uses the `langid.py` tool in two ways. First, they note a problem not previously addressed in cross-lingual parsing – the identification of the target language. The authors argue that it has been always implicitly or explicitly assumed that one knows which language the target text is written in, although

[2] `https://github.com/saffsd/langid.py`

[3] The authors incorporate a prior distribution over languages into their approach, proportional to the sizes of texts in the individual languages provided to the tool in training. Thus, the tool is intentionally biased towards resource-rich languages.

this may not be always the case. Here, they employ `langid.py` in a standard way. However, one may first need to train additional detection models for other languages which are suspected to be likely to appear on the input. While we acknowledge the sensibility of this approach, we do not follow this path in our work, simply assuming that we know the identity of the target language, whenever necessary.

The second and more interesting employment of the `langid.py` tool is for finding a source language most similar to the target language. The method employed by Agić [2017] is to limit the `langid.py` identifier only to the available source languages (excluding the target language), and to apply it in this setting to the target text. The belief is that, prevented from correctly identifying the actual target language, the tool will instead identify the source language which is most similar to the target language, thus accomplishing the source selection task. While the authors note that this heavily relies on orthographic similarity of the languages, which only has a limited power in this scenario,[4] the approach is shown to achieve some success. Nevertheless, in their experiments, `langid.py` performs worse both than KL_{cpos^3} and WALS. Therefore, we omit it in our experiments.

4.1.6 Combining multiple sources

Since multiple sources are available, a viable solution might also be not to select and use just one of them, but to combine multiple or all of them to exploit them best. The combination can be performed at various stages of the processing.

Combine treebanks

The base approach is the already mentioned treebank concatenation method of McDonald et al. [2011].

Søgaard and Wulff [2012] improved the method by enriching it with weighting. They used a POS n-gram model trained on a target POS-tagged corpus to weight source sentences by their similarity to the target language (using perplexity per word). They then incorporate the sentence weights into the training of the linear model of the easy-first parser of Goldberg and Elhadad [2010] by substituting the original structured perceptron training [Collins, 2002] with the weighted perceptron approach of Cavallanti et al. [2010].

Combine parser outputs

The parse tree combination method was introduced by Sagae and Lavie [2006] for a supervised monolingual setting, applying several independent parsers to the input sentence and combining the resulting parse trees using a MST algorithm. A simpler variant of the method was previously employed by Zeman and Žabokrtský [2005], who used a greedy approach instead of the MST algorithm, leading to suboptimal but competitive results.

The combination is performed following the approach of McDonald et al. [2005a], who formulated the problem of finding the parse tree of a sentence as a problem of finding the MST of a weighted directed graph of potential parse tree

[4] This is particularly obvious for close languages that use a different writing script, such as Hindi and Urdu.

edges. In the tree combination method, the weight of each edge is defined as the number of parsers which include that edge in their output. To find the MST, one can use either the algorithm of Eisner [1996], as done by McDonald et al. [2005a] for projective parsing, or the Chu-Liu-Edmonds algorithm [Chu and Liu, 1965, Edmonds, 1967], which was used by McDonald et al. [2005b] for non-projective parsing.

Conveniently, the tree combination method can be easily ported from a monolingual to a multilingual setting, where the individual parsers, now delexicalized, are trained over different languages. To the best of our knowledge, our work in [Rosa and Žabokrtský, 2015b] was the first to do so; we detail our approach in Section 4.3.1.

A nice feature of this tree combination approach is the straightforward possibility of assigning weights to the individual parsers, as done by Surdeanu and Manning [2010] in a monolingual setting. They let each parser contribute with a weight based on its performance (UAS), thus giving a more powerful vote to parsers that seem to be better on average (the weights of identical votes are summed). This approach was used and further refined e.g. by Green and Žabokrtský [2012].

In the cross-lingual transfer, we are more interested in the similarity of the source and target languages, as we would like to give more power to parsers trained on closer languages. Thus, we also introduce a weighted variant of the method in the cross-lingual setting, using our KL_{cpos^3} language similarity estimation to weight the source parsers; this was also already part of [Rosa and Žabokrtský, 2015b].

This approach was later adapted by Agić [2017] in a somewhat different setting with a sightly modified variant of the KL_{cpos^3} measure, finding it to outperform other similarity measures that he proposed; a linear interpolation of the measures was observed to perform even better.

We note that the monolingual accuracy of the parser may also be of interest even in the cross-lingual setting, as e.g. parsers trained on small treebanks can be expected to perform poorly both monolingually and cross-lingually, even if applied to very similar languages. However, it is not clear how to combine these two signals. We have not experimented with incorporating any estimate of the parser accuracy, nor are we aware of any other work doing so; this could thus constitute an interesting subproblem for future research.

Combine parser internals

In the parse tree combination approach, there is a lot of information potentially lost, as each of the source parsers considers many possible parse trees for the sentence, but only outputs one, without any indication of how confident it is about this particular parse tree, or what alternatives there are. This leads to the idea of leveraging the internals of the parser.

Specifically, it seems intuitively reasonable to try to use the scores which the parser assigns to its decisions – such as the edge scores in the MSTParser – and to try to postpone the parse tree induction step till after the multi-source combination, i.e. using the complete weighted graph of all possible edges for the combination.

We explored this idea in our work on MSTParser model interpolation – which, due to the nature of the first-order MSTParser model, is equivalent to interpolating the predicted edge scores. We presented our research in [Rosa and Žabokrtský, 2015a] and review it in Section 4.3.2. However, we have not found this approach to perform better than the simpler tree combination method, and thus abandoned it. As we later realized, this may have been due to the edge scores taken directly from the MSTParser model not being good indicators of the parser confidence. Mejer and Crammer [2012], in a work previously unknown to us, present a range of well-performing methods of per-edge confidence estimation in the MSTParser, which, we believe, might be the key component missing in our implementation to achieve an interesting performance of the interpolation method.

We are not aware of any prior work on interpolating dependency parser models. However, there is work on interpolating trained phrase-structure parsers, both in a monolingual setting for domain adaptation by McClosky et al. [2010], as well as the interpolation of multilingual probabilistic context-free grammars of Cohen et al. [2011].

Interestingly, there is a *later* work of Agić et al. [2016] revisiting our idea of trying to utilize the edge scores predicted by the parser for all of the possible edges, this time in a lexicalized cross-lingual projection of parser outputs over multi-parallel data. As the authors used a different parser – the TurboParser of Martins et al. [2010] – it is not completely clear whether the observations of Mejer and Crammer [2012], which they do not refer to, may also apply to them. However, similarly to us, they do not observe any gains from using the parser scores.

4.2 KL_{cpos^3} language similarity measure

In this section, we present KL_{cpos^3}, our language similarity measure based on KL divergence [Kullback and Leibler, 1951] of POS tag trigram distributions in tagged corpora.

It has been designed for multi-source cross-lingual delexicalized parsing, both for source treebank selection in single-source parser transfer, which has been described in Section 3.2, and for source treebank weighting in multi-source transfer, which will be described in Section 4.3.1.

We present and motivate the formula for computing KL_{cpos^3} in Section 4.2.1, and explain how we use it for source selection in Section 4.2.2 and for source weighting in Section 4.2.3. Section 4.2.4 discusses the POS tags to use for computing the measure, and Section 4.2.5 details the process of tuning the exact definition of the measure. We include intermediary evaluations on the HamleDT 2.0 development treebanks subset throughout this section; evaluations on other datasets will be presented in Section 4.4.

4.2.1 The formula

The measure is based on comparing estimated probability distributions of POS sequences that appear in the source and target languages. This is motivated by the fact that POS tags constitute a key feature for delexicalized parsing.

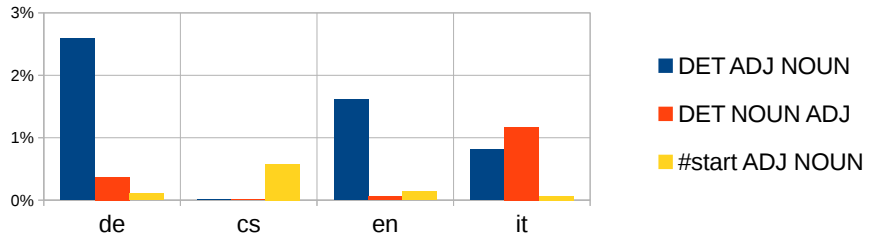

Figure 4.1: Example of estimated probability distributions of several selected POS trigrams in four languages.

The probability distributions are estimated as relative frequencies of POS tag trigrams in the treebank training sections:

$$\hat{P}(cpos_{i-1}, cpos_i, cpos_{i+1}) = \frac{\text{count}(cpos_{i-1}, cpos_i, cpos_{i+1})}{\sum_{\forall cpos_{a,b,c}} \text{count}(cpos_a, cpos_b, cpos_c)}; \tag{4.1}$$

we use a special value for $cpos_{i-1}$ or $cpos_{i+1}$ if $cpos_i$ appears at sentence beginning or end. It is important that the POS tags used are rather coarse; we discuss that in more detail in Section 4.2.4.

See Figure 4.1 for an example of the estimated probability distributions of three POS tag trigrams in four languages. It can be seen that, at least as far as these particular tag sequences are concerned, the English and German languages are quite close to each other, while both Italian and Czech are quite distant from any of the languages.

The particular tag sequences used in this example correspond to the way noun phrases are formed in these languages. Italian has a slight preference of the adjectives to follow the nouns they modify, although they also often precede them; however, in all the other languages, the adjective-noun POS bigram is much more frequent than the noun-adjective one. Furthermore, except for Czech, all of the languages typically start a noun phrase with a determiner, so it is common for a determiner to precede an adjective, and rare for the adjective to immediately start a sentence; in Czech, where determiners are rare, this is the other way round.

Intuitively, when an automatic parser is applied to analyze the structure of a noun phrase in a given target language, one should use a parser trained on a source language that structures noun phrases similarly, so that it can produce the correct analysis. While we assume not to have syntactically annotated data for the target language, Figure 4.1 shows that already morphologically annotated data can suggest a lot about the syntax of the language. This is the motivation behind estimating language similarity from probability distributions of POS tag n-grams.

Furthermore, if we were to analyze English noun phrases by either an Italian or a Czech parser, we expect Italian to be a better choice, since the *DET-ADJ-NOUN* sequence is well known to it, while this is not true for the Czech parser. Moreover, even though the Italian parser also expects to see the inversely ordered *DET-NOUN-ADJ* sequences on the input, which are rare in English, this might not matter much, since this simply means that the ability of the parser to analyze such sequences will remain unexploited when applied to English. This would

however pose a problem in the other direction, using an English parser to analyze Italian noun phrases, since the *DET-NOUN-ADJ* sequence would presumably confuse the parser greatly, as it is not used to encountering it, and therefore presumably unable to handle it correctly. This motivates our use of KL divergence in a particular direction.

We first represent here the general formula for KL divergence from Q to P, $D_{\mathrm{KL}}(P||Q)$, where P and Q are two discrete probability distributions, P being the true or expected distribution, and Q being an approximation or model of P used instead of P:

$$D_{\mathrm{KL}}(P||Q) = \sum_{\forall x} P(x) \cdot \log \frac{P(x)}{Q(x)}\,, \tag{4.2}$$

with the value of the addend defined as 0 if $P(x) = 0$. The value of KL divergence is a non-negative number; the more divergent (dissimilar) the distributions, the higher its value.

In our setting, we estimate the distance of a source language to a target language as the KL divergence of the POS trigram probability distributions, $D_{\mathrm{KL}}(\hat{P}_{tgt}||\hat{P}_{src})$:

$$KL_{cpos^3}(tgt, src) = \sum_{\forall cpos^3 \in tgt} \hat{P}_{tgt}(cpos^3) \cdot \log \frac{\hat{P}_{tgt}(cpos^3)}{\hat{P}_{src}(cpos^3)}\,, \tag{4.3}$$

where $cpos^3$ is a POS tag trigram.

For the KL divergence to be well-defined, we must ensure that the estimated probability of each target trigram is non-zero in source. For this, we employ a simple "add 1" smoothing approach in the source trigrams probability estimation (4.1): for each target trigram unseen in the source data, we set its source count to 1. We have found this smoothing approach to perform well, and did not investigate other options. Recently, Agić [2017] suggested to instead use the approach of Brants [2000], estimating the trigram probabilities as a linear interpolation of trigram, bigram and unigram relative frequencies, as a softer and more principled way of performing the smoothing. We acknowledge this suggestion as potentially promising, also allowing us to easily incorporate higher-order n-grams, and intend to investigate it more thoroughly in future research.

The KL divergence is non-symmetric; $D_{\mathrm{KL}}(P||Q)$ expresses the amount of information lost when a probability distribution Q is used to approximate the true distribution P. Thus, in our setting, we use $D_{\mathrm{KL}}(\hat{P}_{tgt}||\hat{P}_{src})$, as this intuitively corresponds to trying to minimize the error caused by using a source parser as an approximation of a target parser (we are approximating the target language by the source language).

Or, in even simpler words, it does not matter to us much how frequent a POS trigram is in the source language if it does not appear in the target language, so we simply ignore it in the computation (its summand is zero). What we do need, is for POS trigrams frequent in the target language to appear in the source language frequently enough. The less frequent they are in the source language (i.e. the less trained the source parser is to analyze such structures), the smaller the denominator, and thus the larger the summand, increasing the resulting estimate of the distance of the languages.

Target language		Oracle source		KL_{cpos^3} source			Δ
ar	Arabic	ro	43.1	1.7	sk	41.2	-1.9
bg	Bulgarian	sk	66.8	0.4	**sk**	**66.8**	0.0
ca	Catalan	es	72.4	0.1	**es**	**72.6**	0.0
el	Greek	sk	61.4	0.7	cs	60.7	-0.7
es	Spanish	ca	72.7	0.0	**ca**	**72.7**	0.0
et	Estonian	hu	71.8	0.9	da	64.9	-6.9
fa	Persian	ar	35.6	1.1	cs	34.7	-0.9
fi	Finnish	et	44.2	1.1	**et**	**44.2**	0.0
hi	Hindi	ta	56.3	1.1	fa	20.8	-35.5
hu	Hungarian	et	52.0	0.7	cs	46.0	-6.0
it	Italian	ca	59.8	0.3	pt	54.9	-4.9
ja	Japanese	tr	49.2	2.2	ta	44.9	-4.3
	Average		**57.1**	0.9		52.0	-5.1
	Std. Dev.		12.5			16.1	

Table 4.1: Performance of KL_{cpos^3} in source selection for single-source delexicalized cross-lingual parser transfer, measured using UAS on HamleDT 2.0 development target treebanks.

Empirical arguments for choosing the similarity measure and the exact formula of KL_{cpos^3}, based on evaluations on development data, can be found in Section 4.2.5.

4.2.2 KL_{cpos^3} for source selection

In the single-source parser transfer, the delexicalized parser is trained on a single source treebank, and applied to the target corpus. The problem thus reduces to selecting a source treebank that will lead to a high performance on the target language.

In this case, we compute the KL_{cpos^3} distance of the target corpus to each of the source treebanks and choose the closest source treebank to use for the transfer.

We evaluate the performance of KL_{cpos^3} on the development treebanks in Table 4.1 (both development and test treebanks were used as sources here, in a leave-one-out fashion). For each target language, the table first lists the oracle source language, and the UAS achieved by applying the delexicalized parser trained on the oracle source treebank to the target language treebank; the oracle language is the source language which maximizes this UAS, and is thus the language we would ideally like KL_{cpos^3} to select. Next, the language actually selected by KL_{cpos^3} is listed, together with the value of KL_{cpos^3}, and the UAS achieved by using the selected source language for the delexicalized parser transfer. The difference of UAS achieved by using the oracle source versus the automatically selected source is also shown.

As the table shows, KL_{cpos^3} manages to identify the oracle source language in one third of the cases (*bg, ca, es, fi*). In many other cases, it chooses a different but competitive treebank, leading only to a tiny drop in UAS (*ar, el, fa*). For other target languages (*et, hu, it, ja*), there is a larger drop, i.e. the selected

language is only somewhat appropriate. However, only in one case (*hi*) there is a massive drop in UAS as compared to using the oracle source.

4.2.3 $KL^{-4}_{cpos^3}$ for source weighting

In multi-source transfer, multiple (or all) available source treebanks are used for parser training, possibly weighted by similarity to the target language. In this case, we need to appropriately weight the contribution of each of the sources.

To convert KL_{cpos^3} from a negative measure of language similarity to a positive source parser weight for the multi-source tree combination method, we need to find a way of inverting it. However, as the results were unsatisfactory with simply using the inverted value ($1/x$), we did some further tuning (see Section 4.2.5), empirically finding the fourth power of the inverted value ($1/x^4$) to work well. Thus, the contribution of each of the sources gets weighted by $KL^{-4}_{cpos^3}(tgt, src)$.

We deal with multi-source transfer methods in Section 4.3, where we explain how $KL^{-4}_{cpos^3}$ is used for the source weighting, and evaluate the improvements that it brings.

A full listing of the estimated source-target $KL^{-4}_{cpos^3}$ language similarities, computed on the UD 1.4 dataset, is enclosed in Attachment C.

4.2.4 The POS tags

Various treebanks and theories use various definitions of POS tags. While traditionally, the basic categories, such as noun or verb, are typically understood as the part-of-speech, in some treebanks, other morphological information can be encoded in the tag – for example, for nouns, there are 2 UPOS tags in UD (NOUN and PROPN, distinguishing proper nouns), 4 in the Penn treebank (NN, NNS, NNP and NNPS, adding the singular/plural distinction), and 186 in PDT (2 numbers, 4 genders, 7 cases and positive/negative are commonly distinguished, additionally allowing "non-specified" values, plus some rare special values and features).

For KL_{cpos^3} to work, the POS tags need to be reasonable. The first requirement is that the POS tags are harmonized – it does not make any sense to compare probability distributions of POS tag labels in two languages where the labels used are different, or have very different meanings.

Second, we intuitively believe that for a good performance of the method, all of the POS tag labels defined by the tagset should appear in all or nearly all of the languages – it is reasonable to e.g. include a label for determiners, which appear in many languages (although not all), but may not be a good idea to differentiate e.g. between singular and plural nouns in the tag, as there are many languages where noun gender is not overtly marked on the nouns (or not annotated by the treebank annotators, which, for our purposes, is the same thing).

And finally, the tagset should not be too large, which would make the data too sparse – KL_{cpos^3} operates on tag trigrams, so even for a set of 10 POS tags, we are working with a distribution of up to 1000 trigrams. This inevitably leads to many of the trigrams being very rare in the data, with the estimated probabilities being quite unreliable.

We decided to emphasize the kind of POS tags the KL_{cpos^3} measure expects already in its name, where "cpos" means "coarse POS", as it was used e.g. in the CoNLL treebanks – the coarse POS tags were stored in the fourth column of the CoNLL files, labelled CPOS. As the UD have become the *de facto* standard for morphological and syntactic annotation since we defined the measure, we could now as well rename the measure to KL_{upos^3}; however, we prefer to keep the name unchanged.

When we were developing the measure, we used the gold POS tags present in the HamleDT 2.0 treebanks, both on the source side and on the target side. However, it has been since shown that the measure is quite robust to the source of the POS tags, including well performance on cross-lingually predicted UPOS tags. See Section 4.4.2 for an evaluation of the influence of the source of the POS tags, and Chapter 6 for inducing target POS tags without access to POS-tagged target data.

4.2.5 Tuning

To avoid overfitting the exact definition of KL_{cpos^3} and $KL^{-4}_{cpos^3}$, we used the HamleDT 2.0 development treebanks to tune the hyperparameters; namely the choice of the measure, POS n-gram length, and the way of transforming the KL divergence, which is a distance measure, into a similarity measure.

POS n-gram length

We tried to vary the n-gram length from bigrams to tetragrams (we did not include unigrams as they do not capture the word order).

The n-gram length has a surprisingly low effect on the result in the multi-source combination – typically less than 1%. However, as Table 4.2 shows, for source selection, trigrams clearly outperformed the tetragrams and bigrams, identifying the oracle language in half of the cases, and subsequently achieving the highest average UAS when applying the selected source delexicalized parser to the target data. Note, however, that for half of the targets, using any of the n-gram lengths led to choosing the same source language.

It seems that bigrams are not predictive enough, while tetragrams are too sparse in the data and thus the probability distributions are estimated too unreliably.

Turning a distance into a similarity

As KL divergence is a distance measure, while we need a similarity measure for weighting the source contributions in multi-source combination methods, we need to find a way to transform it appropriately. We decided to combine inversion of the value with exponentiation, i.e. using a formula in the form $KL^{-N}_{cpos^3}$, where N is a hyperparameter to tune.

Table 4.3 shows the effect of using various exponents. In this tuning step, the weight was applied to the weighted model interpolation (Section 4.3.2) and evaluated on HamleDT 2.0 development treebanks used both as the targets and the sources (in a leave-one-out fashion).

Target	Oracle		KL_{cpos^2}		KL_{cpos^3}		KL_{cpos^4}	
ar	it	36.57	fa	27.19	**it**	**36.57**	et	23.87
bg	el	63.04	**el**	**63.04**	**el**	**63.04**	it	56.32
ca	es	72.37	**es**	72.37	**es**	72.37	**es**	72.37
el	bg	59.49	**bg**	**59.49**	it	53.31	it	53.31
es	ca	72.73	**ca**	72.73	**ca**	72.73	**ca**	72.73
et	hu	71.76	fi	68.93	fi	68.93	fi	68.93
fa	ar	35.64	hi	19.84	hi	19.84	hi	19.84
fi	et	44.22	ar	26.06	**et**	**44.22**	**et**	**44.22**
hi	hu	47.15	fa	20.77	fa	20.77	fa	20.77
hu	et	52.04	el	47.11	el	47.11	el	47.11
it	ca	59.75	es	58.52	es	58.52	es	58.52
ja	hi	49.78	bg	24.97	**hi**	**49.78**	et	41.11
AVG		55.38	4	46.75	**6**	**50.60**	3	48.26

Table 4.2: The effect of POS n-gram length in KL_{cpos^3} for source selection, using HamleDT 2.0 development treebanks as both sources and targets.

Target	$KL^{-1}_{cpos^3}$	$KL^{-3}_{cpos^3}$	$KL^{-4}_{cpos^3}$	$KL^{-5}_{cpos^3}$	Max	$\Delta KL^{-4}_{cpos^3}$
ar	25.64	27.64	28.27	**29.20**	29.20	-0.93
bg	57.47	62.58	63.58	**64.23**	64.23	-0.65
ca	71.94	**72.39**	**72.39**	**72.39**	72.39	0.00
el	58.87	60.24	60.54	**61.00**	61.00	-0.46
es	71.53	**72.03**	**72.03**	**72.03**	72.03	0.00
et	66.74	68.48	**70.84**	70.28	70.84	0.00
fa	**32.78**	31.30	30.53	30.26	32.78	-2.25
fi	44.94	**45.51**	45.13	44.66	45.51	-0.38
hi	**33.87**	31.60	30.10	29.62	33.87	-3.77
hu	**54.18**	53.34	52.65	51.63	54.18	-1.53
it	**58.08**	57.84	57.26	57.00	58.08	-0.82
ja	36.98	40.45	41.47	**42.17**	42.17	-0.70
AVG	51.09	51.95	**52.07**	52.04	53.02	-0.96
Best	4	3	3	6		

Table 4.3: Evaluation of various values for the exponent used to invert KL_{cpos^3}, applied to weighted parser model interpolation, reporting UAS on HamleDT 2.0 development treebanks. Difference between $KL^{-4}_{cpos^3}$ and the maximum is also listed.

In our evaluation, the parsing accuracy generally tends to rise with increasing the exponent for half of the targets, and decrease for the other half, not making it easy to choose an appropriate solution. A high value of the exponent strongly promotes the most similar source language, giving minimal power to the other languages, which is good if there is one *very* similar source language. A low value enables combining information from a larger number of source languages.

Our choice of the final exponent is thus a matter of compromise. While the exponent value of 5 leads to the best accuracy for the largest number of languages, the average accuracy does not differ substantially from using the exponent value of 3 or 4; the good results for some of the targets come at the cost of larger losses for the other targets, which prefer a lower value of the exponent. As we want the weighting to be rather well balanced, avoiding large losses is more important for us than achieving the highest possible individual accuracies. We thus eventually selected the value of 4 for the exponent, as it performed similarly to the value of 3 but achieved a slightly better average accuracy (but 3 could also have been chosen).

Agić [2017] suggests a different approach, applying the softmax transformation with temperature 0.2 to the inverted distances of the languages, thus obtaining a probability distribution over the sources for each target. Unless one intends to combine multiple similarity measures (which Agić does but we do not), the denominator can be left out, leading to the following similarity measure:

$$sim_{Agic}(tgt, src) = \exp\left(\frac{5}{KL_{cpos^3}(tgt, src)}\right) \tag{4.4}$$

The author offers neither a motivation for diverging from our approach, nor any other justification. We thus performed an empirical comparison in Section 5.5.5, finding our original approach to consistently outperform the transformation suggested by Agić on our dataset.

The choice of the measure

Table 4.4 contains evaluation of several language similarity measures considered in the tuning phase, applied to source weighting in delexicalized multi-source tree-combination cross-lingual parser transfer (Section 4.3.1) and evaluated using UAS on the development set.

To compare the source and target POS trigram distributions, we evaluated Kullback-Leibler divergence computed in both directions, Jensen-Shannon divergence [Lee, 2001], which is a symmetrization of KL divergence, and cosine similarity.

Based on the results, $KL^{-4}_{cpos^3}(target, source)$ was selected, as it performed best in all aspects – it achieved the best observed accuracy most frequently (for 6 of the 12 targets), also outperforming the other variants in the average accuracy, and it is also the most stable of the measures, as shown by its low standard deviation.

Target	$KL^{-4}_{cpos^3}(tgt, src)$	$KL^{-4}_{cpos^3}(src, tgt)$	$JS^{-4}_{cpos^3}(tgt, src)$	$cos_{cpos^3}(tgt, src)$
ar	**34.93**	29.72	32.22	30.53
bg	**63.89**	53.61	53.86	60.73
ca	**72.39**	72.37	72.37	67.93
el	**58.13**	57.83	56.45	57.89
es	72.03	**72.73**	**72.73**	68.10
et	68.49	68.83	**71.03**	69.46
fa	**33.51**	27.82	24.72	32.06
fi	43.81	**44.22**	44.20	39.91
hi	**21.40**	20.77	21.23	19.54
hu	48.23	**50.04**	49.21	49.95
it	55.05	60.18	59.38	**61.28**
ja	40.30	**48.47**	38.22	30.22
Best	**6**	4	2	1
AVG	**51.01**	50.55	49.64	48.97
StDev	**16.66**	17.42	17.98	17.69

Table 4.4: Weighted multi-source delexicalized tree-combination transfer using various similarity measures.

4.3 Multi-source combination methods

To the best of our knowledge, the idea of combining multiple source languages for analyzing one target languages was introduced by McDonald et al. [2011]. The authors used a simple treebank concatenation method, combining all available source treebanks into one multilingual treebank, and using it to train one multilingual delexicalized parser. This method does not assign explicit weights to individual source languages; each source language is implicitly weighted by the size of its treebank, regardless of the target language. We take this method as a baseline approach.

Our work on cross-lingual parsing in a multi-source setting rests on two pillars. The first one, the KL_{cpos^3} language similarity measure, was presented in Section 4.2, allowing us to estimate how appropriate each available source language is for processing a given target language, i.e. for training a delexicalized parser on the source language and applying it to the target language.

However, as was already foreshadowed, it is often the case that there are multiple source languages close enough to the target, and one would then like to learn from all such languages; the hope is that treebanks for other similar languages might provide knowledge which is not available in the treebank for the closest language but is necessary to parse the target language.

And there is yet another, more pragmatic reason to take multiple sources into account. Even though it performs very well, the KL_{cpos^3} measure is far from infallible, often failing to designate the optimal source language. In such cases, we would like to have a method of bringing in other promising sources, trying to alleviate the damage done by not choosing the right source. This is a sort of risk management – we accept the risk of achieving slightly suboptimal accuracies for some languages to avoid massively suboptimal performance for other languages; this is what we tuned the $KL^{-4}_{cpos^3}$ measure for.

For this purpose, we present three methods for combining multiple sources that we have experimented with. Section 4.3.1 presents the parse tree combination method of Sagae and Lavie [2006], which we ported to the cross-lingual setting in [Rosa and Žabokrtský, 2015b]; this is our primary method, as it showed best performance in our evaluations, and we will only use this method in further experiments. In Section 4.3.2, we show an alternative method of parser model interpolation (or interpolation of predicted scores of potential parse tree edges), which we developed in [Rosa and Žabokrtský, 2015a]; it often performs comparably to parse tree combination, but is less robust, and we therefore do not use it in further experiments. For comparison, we also tried to reimplement the parse tree projection method of Agić et al. [2016] in Section 4.3.3, but, contrary to the original work, we have found it to perform substantially worse than the parse tree combination.[5]

4.3.1 Parse tree combination

The multi-source cross-lingual delexicalized parse tree combination method is a simple parser ensembling approach. The original method, which had been devised for a monolingual setting, combines various parsers (i.e. different parsing algorithms), all trained on the same treebank. In our extension to the cross-lingual setting, we only use one parser, but trained on various source treebanks.

Furthermore, as the languages of the source treebanks are different from each other as well as from the target language, we need to delexicalize the parsers – we assume that the POS tags used in all of the languages are the same, but we cannot possibly assume that for the word forms. We have already discussed the POS tags in Section 4.2.4, and we will show how to obtain them cross-lingually in Chapter 6. Moreover, in Chapter 5, we will show how to lexicalize the parser in a cross-lingual setting; but for now, we limit ourselves to a delexicalized one.

Unweighted tree combination

In our work, we implement the tree combination method in its base unweighted variant in the following way (see also Figure 4.2):

1. Train a delexicalized parser on each source treebank.
2. Apply each of the parsers to the target sentence, obtaining a set of parse trees.
3. Construct a weighted directed graph as a complete graph over all tokens of the target sentence, where each edge is assigned a score equal to the number of parse trees in which it appears (each parse tree contributes by either 0 or 1 to the edge score).
4. Find the final dependency parse tree as the maximum spanning tree over the graph, using the algorithm of Chu and Liu [1965] and Edmonds [1967].

[5]We think that the differences in performance may be due to some differences in our implementation, but we have not been able to identify those. However, when applied to POS projection instead of parse tree projection (Section 6.1), the method performed well and in line with the results reported in the original paper, suggesting that our reimplementation is mostly correct.

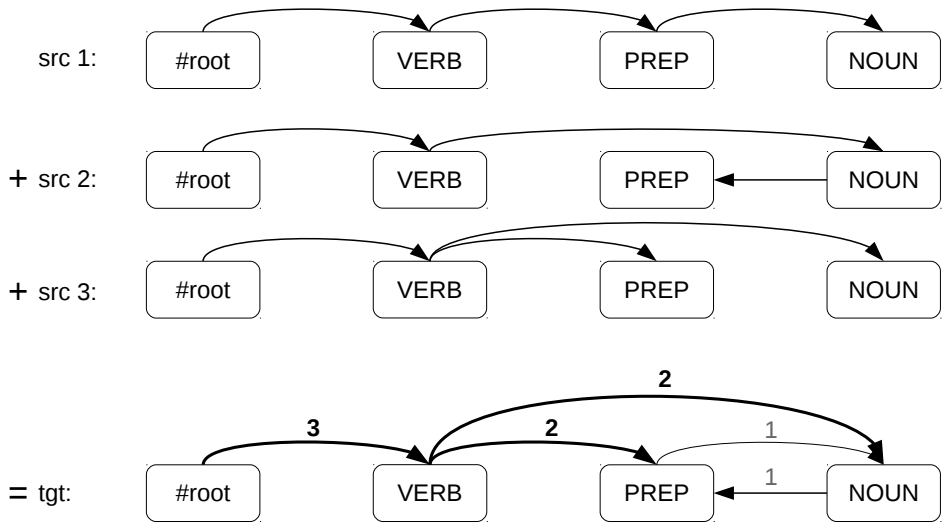

Figure 4.2: Unweighted parse tree combination, combining the parse trees for a delexicalized target sentence (tgt) produced by 3 source parsers (src 1, src 2 and src 3), and selecting the highest scoring dependency tree (MST) as the result (in bold).

We can also formulate the third step using the following formula for the score w_e that each edge e gets assigned:

$$w_e = \sum_{\forall src} I(e \in tree_{src}) \,, \tag{4.5}$$

where the indicator $I(e \in tree_{src})$ is 1 if the edge e appears in the parse tree produced by the parser trained on the source language *src*, and 0 otherwise.

Weighted tree combination

As we have already noted, the method can be further enhanced by adding weighting. In our case, we use the $KL^{-4}_{cpos^3}$ source-target language similarity estimation; i.e., the same weight is applied to all edges in all parse trees produced by a parser for a given source language (see also Figure 4.3).

Thus, in the weighted variant of the method, the third step of the algorithm is modified by each source contributing not with 0 or 1 to the edge score, but with the value of its $KL^{-4}_{cpos^3}$ similarity to the target:

$$w_e = \sum_{\forall src} I(e \in tree_{src}) \cdot KL^{-4}_{cpos^3}(tgt, src) \,. \tag{4.6}$$

It might also be possible to use more fine-grained weighting, e.g. on the level of individual sentences rather than whole languages – we imagine that a variant of KL_{cpos^3} (or other similarity measure) may be applicable to the target sentence, estimating to which source languages the sentence is most similar. However, we have not experimented with that; but we did experiment with adding edge-level weighting, as will be shown in Section 4.3.2.

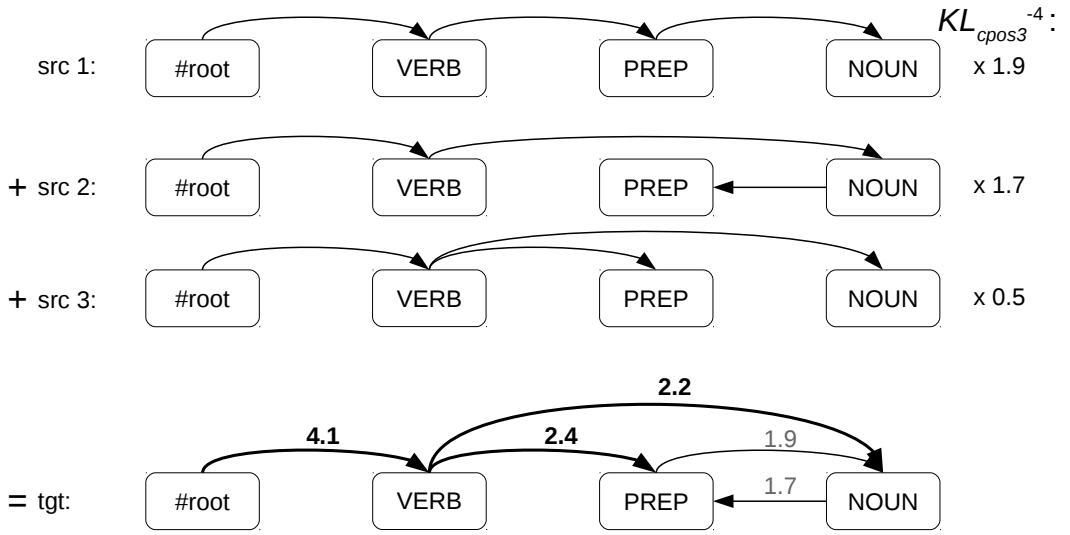

Figure 4.3: Weighted parse tree combination, combining the parse trees for a delexicalized target sentence (tgt) produced by 3 source parsers (src 1, src 2 and src 3), weighted by the $KL_{cpos^3}^{-4}$ similarity of each source language to the target language, and selecting the highest scoring dependency tree (MST) as the result (in bold).

Target language	Treebank concat.	Single-source oracle		Single-source KL_{cpos^3}			Tree combination w=1	Tree combination w=$KL_{cpos^3}^{-4}$
ar	37.0	ro	**43.1**	1.7	sk	41.2	35.3	**41.3**
bg	64.4	sk	66.8	0.4	**sk**	66.8	66.0	**67.4**
ca	56.3	es	**72.4**	0.1	**es**	**72.4**	61.5	**72.4**
el	63.1	sk	61.4	0.7	cs	60.7	62.3	**63.8**
es	59.9	ca	**72.7**	0.0	**ca**	**72.7**	64.3	**72.7**
et	67.5	hu	71.8	0.9	da	64.9	70.5	**72.0**
fa	30.9	ar	**35.6**	1.1	cs	**34.7**	32.5	33.3
fi	41.9	et	44.2	1.1	**et**	44.2	41.7	**47.1**
hi	24.1	ta	**56.3**	1.1	fa	20.8	24.6	**27.2**
hu	55.1	et	52.0	0.7	cs	46.0	**56.5**	51.2
it	52.5	ca	**59.8**	0.3	pt	54.9	59.5	**59.6**
ja	29.2	tr	**49.2**	2.2	ta	**44.9**	28.8	34.1
AVG	48.5		**57.1**	0.9		52.0	50.3	**53.5**
Std. Dev.	15.2		12.5			16.1	16.5	16.7

Table 4.5: Evaluation of the parse tree projection method on HamleDT 2.0 development target treebanks, using UAS. For comparison, we also show the treebank concatenation and KL_{cpos^3}-best single-source transfer baselines, as well as the oracle single-source transfer.

Performance on development set

Table 4.5 shows the performance of the tree combination method, both unweighted as well as weighted using $KL^{-4}_{cpos^3}$, evaluated on the HamleDT 2.0 development target treebanks using UAS.

We can see that the weighted tree combination performs better than the unweighted one in all but one case (*hu*) as well as on average; the average performance is also better than that of the single-source transfer using the source language selected by KL_{cpos^3}.

Interestingly, the multi-source combination even surpasses the single-source oracle transfer in 4 cases (*bg, el, et, fi*), even though KL_{cpos^3} was able to identify the oracle source language only for 2 of these targets. This shows that the ability to weightedly use information from multiple sources can be even more important than that of selecting the single best possible source.

We can also see that the multi-source combination does indeed mostly fulfill its intended role of providing a backoff for cases where the single-source transfer fails to identify the oracle language, as the weighted tree combination improves over the single-source transfer in all but one such case (*ja*), often quite substantially. Of course, it may happen that the tree combination overpowers the single-best source even in cases where it *was* selected correctly, possibly leading to deteriorations. However, the results show that with $KL^{-4}_{cpos^3}$ weighting this is not the case, as in all cases where KL_{cpos^3} was able to identify the oracle language, the weighted tree combination either preserves the achieved performance (*ca, es*), or even further improves over it (*bg, fi*).

On the other hand, the average performance of the oracle single-source transfer is still better than that of the multi-source combination, mainly because of its very low score for some difficult targets (*hi, ja*).

Labels

So far, we have only presented the unlabelled variant of the tree combination method; that is, we have only dealt with inferring the dependency tree structure, but not the dependency relation labels.

We decided to use an approach fundamentally similar to the tree combination method, but without the final decoding through MST, as there are no clear structural constraints on the dependency relation labels.

More specifically, we assign the dependency relation labels by using a simple voting mechanism, where, in the unweighted variant, each edge from a dependent node to its parent is assigned the label predicted by the largest number of source parsers for an edge with that dependent. In the weighted variant, the votes of the source parsers are weighted by $KL^{-4}_{cpos^3}$. We take a practically identical approach in multi-source cross-lingual tagging in Chapter 6.

For simplicity, we include all sources in the voting, even if they predicted a different head for that dependent. While this may sometimes lead to assigning a label not compatible with the edge (e.g. a subject label even if the parent is not a verb), we note that the dependency relation label typically depends much more on the dependent node (and especially on its POS) than on the head node. The dependency relation label predicted by a given source may thus still be correct even if the head predicted by that source is not correct.

We found this approach to work well in practice, and did not investigate other options. However, we believe that further improvements might be possible, and that the following modifications might be worth evaluating in some future work:

- The votes of sources predicting a different head node than the selected one may be downscaled; probably more so if the POS and/or direction of the predicted head differs from that of the selected one.

- Alternatively, one might only consider dependency relation labels suggested by the parsers that predicted the selected head node, with the other sources only participating in the voting if they agree on the dependency relation label with one of those parsers.

- Information from other predicted dependency trees may be harvested, using e.g. the usual dependency relation label for the given pair of dependent and head as an additional vote, based on the POS of the dependent and the head, as well as potentially the direction and/or length of the edge.

- Hard constraints originating from the UD annotation rules or trained on the source treebanks might be enforced, such as prohibiting a head node to have more than one subject dependent.

- And, eventually, one might even train an independent delexicalized dependency relation labeller on the source treebanks and use the aforementioned methods to transfer it cross-lingually to the target language. Note that using an independent second-stage labeller was suggested by McDonald et al. [2005a] for the MSTParser; e.g. our implementation [Rosa and Mareček, 2012], using MIRA training and Viterbi decoding,[6] could be adapted and used.

4.3.2 Parser model interpolation

Here, we present our alternative multi-source combination method – the interpolation of trained parser models.

The method proceeds as follows:

1. Train a delexicalized parser model on each source treebank.
2. Normalize the parser models.
3. Interpolate the parser models.
4. Parse the target text with a delexicalized parser using the interpolated model.

When applied to the first-order MSTParser, this method is fully equivalent to interpolating the individual edge scores predicted by the parsers.

[6] It is actually a tree variant of the Viterbi algorithm, operating on sequences of sibling dependency nodes.

Model normalization

An important preliminary step to model interpolation is to normalize each of the trained models, as the feature weights in models trained over different treebanks are often not on the same scale (we do not perform any regularization during the parser training). We use a simplified version of normalization by standard deviation. First, we compute the uncorrected sample standard deviation of the weights of the features in the model M as

$$s_M = \sqrt{\frac{1}{|M|} \sum_{\forall f \in M} (w_f - \bar{w})^2}\,, \tag{4.7}$$

where $\bar{w}$ is the average feature weight, and $|M|$ is the number of feature weights in model M; only features that were assigned a non-zero weight by the training algorithm are taken into account.

We then divide each feature weight by the standard deviation:[7]

$$\forall f \in M : w_f := \frac{w_f}{s_M}\,. \tag{4.8}$$

The choice of normalization by standard deviation is based on its high and stable performance on our development set, and Occam's razor.[8]

Model interpolation

The interpolated model is a linear combination of the normalized models trained over the source treebanks. The result is a model that can be used in the same way as a standard MSTParser model.

In unweighted model interpolation, the weight of each feature (w_f) is computed as the sum of the weights of that feature in the source models ($w_{f,src}$):

$$\forall f \in F : w_f = \sum_{\forall src} w_{f,src}\,. \tag{4.9}$$

In the weighted variant of model interpolation, we extend (4.9) with multiplication by the $KL^{-4}_{cpos^3}$ weight:

$$\forall f \in F : w_f = \sum_{\forall src} w_{f,src} \cdot KL^{-4}_{cpos^3}(tgt, src)\,. \tag{4.10}$$

[7] We have not found any further gains in performance when subtracting the sample mean from the weight before the division; due to the particular way in which we implemented the MIRA updates in MSTperl, the mean of the feature weights in the trained model is guaranteed to already be equal to 0, except for some minor variation due to rounding errors.

[8] We tried 12 normalization schemes, nearly all of which achieved an improvement of 2.5% to 5% UAS absolute over an interpolation of unnormalized models on average, but often with large differences for individual languages. Another well-performing method was to divide each feature weight by the sum of absolute values of all feature weights in the model; or a similar method, applied during inference individually for each sentence, using only the feature weights that fired for the sentence to compute the divisor.

Equivalence to edge score interpolation

Note that the model interpolation method, both weighted and unweighted, is fully equivalent to interpolation of individual scores of all edges in the complete graph, as predicted by the models (provided that the models are normalized).

Recall the score $w_{e,M}$ assigned to an edge e by the model M:

$$w_{e,M} = \sum_{\forall f \in F} w_{f,M} \cdot f(e) \tag{4.11}$$

where F is the set of features, $w_{f,M}$ are their weights determined by the model, and $f(e)$ is the value of the feature for edge e.

Next, if an interpolated model is used, we can substitute $w_{f,M}$ by its definition from (4.10):

$$w_{e,int} = \sum_{\forall f \in F} \sum_{\forall src} w_{src} \cdot w_{f,src} \cdot f(e) \tag{4.12}$$

where w_{src} is the weight of the source: $KL^{-4}_{cpos^3}(tgt, src)$ in the weighted approach or 1 in the unweighted approach.

However, by simple reordering, we can get:

$$w_{e,int} = \sum_{\forall src} w_{src} \cdot \sum_{\forall f \in F} w_{f,src} \cdot f(e) \tag{4.13}$$

which can be simplified to:

$$w_{e,int} = \sum_{\forall src} w_{src} \cdot w_{e,src} \tag{4.14}$$

where $w_{e,src}$ is the score assigned to the edge e by the *src* model. However, (4.14) does not require modifying the trained models in any way, it simply uses the models to predict the edge scores, and then interpolates these edge scores using the source weights. So, this is an equivalent alternative implementation of the method.

Using this formulation of the method, we can see that it is actually quite similar to the tree combination method, enriched with the source edge weights (see also Figure 4.4) – the tree combination is equivalent to edge score interpolation where all edges selected by the MST algorithm get a score of 1 and all other edges get a score of 0. Once the target weighted graph is constructed, both of the methods continue identically by finding its MST and returning it as the final parse tree.

Comparison to tree combination

Table 4.6 compares the performance of the model interpolation method to the tree combination method on the 12 development language treebanks from the HamleDT 2.0 dataset. All the 30 HamleDT 2.0 treebanks, with gold POS tags, were used to compute the $KL^{-4}_{cpos^3}$ similarity for each pair of languages, and to train delexicalized MSTperl parsers. Then, for each target language, all the other 29 source language parsers were combined.

The performance of the weighted model interpolation is comparable to the weighted tree combination – model interpolation scores -0.5% UAS lower than

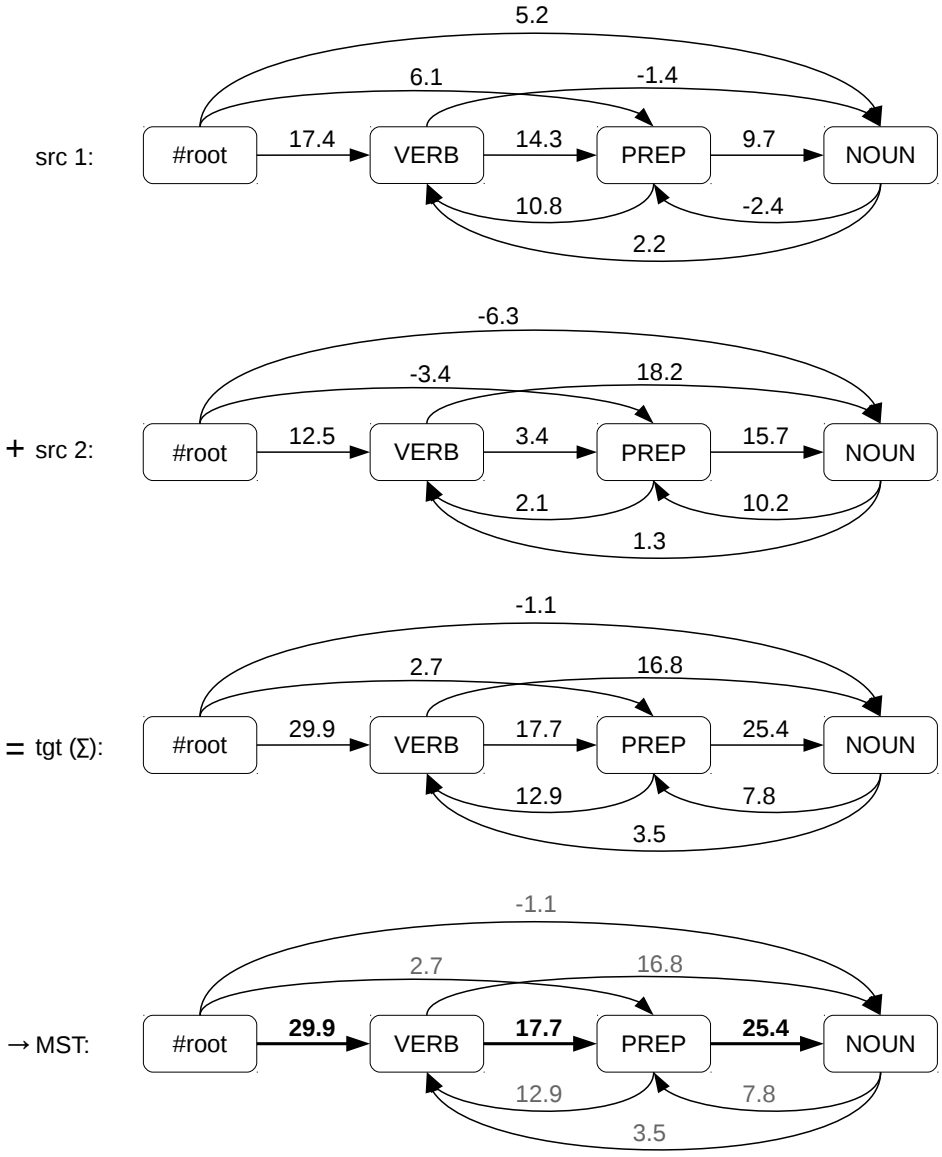

Figure 4.4: Unweighted parser model interpolation shown as unweighted edge scores interpolation, combining the weighted complete graphs of possible dependency edges for a delexicalized target sentence (tgt) produced by 2 source parsers (src 1 and src 2), and selecting the highest scoring dependency tree (MST) as the result.

Target language		Unweighted			Weighted	
		Treebank concat.	Tree comb.	Model interp.	Tree comb.	Model interp.
ar	Arabic	**37.0**	35.3	30.7	**41.3**	34.6
bg	Bulgarian	64.4	**66.0**	60.3	67.4	**68.5**
ca	Catalan	56.3	**61.5**	58.5	72.4	**72.4**
el	Greek	**63.1**	62.3	59.6	63.8	**64.1**
es	Spanish	59.9	**64.3**	60.4	72.7	**72.7**
et	Estonian	67.5	**70.5**	67.4	**72.0**	71.7
fa	Persian	30.9	**32.5**	29.5	**33.3**	28.6
fi	Finnish	**41.9**	41.7	41.5	**47.1**	44.7
hi	Hindi	24.1	24.6	**26.2**	27.2	**32.7**
hu	Hungarian	55.1	56.5	**57.4**	51.2	**53.0**
it	Italian	52.5	**59.5**	56.0	59.6	**60.1**
ja	Japanese	**29.2**	28.8	27.2	**34.1**	33.0
	Average	48.5	**50.3**	47.9	**53.5**	53.0
	Std. dev.	**15.2**	16.5	15.6	**16.7**	17.4

Table 4.6: Comparison of UAS of the tree concatenation baseline with parse tree combination and parser model interpolation, both weighted and unweighted, on the development target treebanks of the HamleDT 2.0 dataset.

the tree combination, but achieves a better result than the tree combination for 7 of the 12 target languages; for most of the target languages, the accuracies of the two methods are very close. This shows that weighted model interpolation may be a good alternative to weighted tree combination.

In the unweighted setting, the situation is quite different, with model interpolation scoring much lower than tree combination (-2.4%), and even slightly lower than treebank concatenation (-0.6%) on average. This suggests that, contrary to our original intuition, edge scores assigned by the source models are not a good proxy for parser confidence, not even when appropriately normalized.[9] This may suggest, among other, that the source-target language similarity is much more important than the exact values of source edge scores for resource combination in delexicalized parser transfer.

Discussion and related work

While the parser model interpolation method is promising, it does not empirically show any clear advantages over the tree combination method, and is much more sensitive to proper weighting of the sources. Also, it relies on having access to the internals of the parsers, which the tree combination method does not. As we initially used MSTperl, our own reimplementation of the MSTParser, we had access to this information. However, when we later switched to a state-of-the-art Parsito/UDPipe parser, we lost access to this information, and abandoned the parser model interpolation method altogether.

It was later brought to our attention that Mejer and Crammer [2012] had

[9]The same tendency was observed across all normalization methods evaluated on the development set.

already clearly demonstrated that the edge scores predicted by the MSTParser are not very good indicators of the parser confidence. Instead, the authors suggest several alternative methods for estimating the confidence of the parser about the correctness of a given edge. The methods seem to perform well, although they require modifications of the parser architecture and lead to an increased computational complexity of the parsing algorithm. Nevertheless, we believe that using their parser confidence estimation methods with our parser interpolation method could lead to interesting results, improving over the more crude tree combination method. Unfortunately, we do not perform such experiments in this work.

Later, Agić et al. [2016] revisited this idea, using a practically identical approach to ours – the scores of target edges are determined by an interpolation (a weighted sum) of the source edge scores predicted by the source parsers, taking into account the complete weighted graphs, and the target dependency tree is then induced via the MST algorithm. Also, the source scores are normalized via the normalization by standard deviation, but applied per sentence, not per model. The only difference is that the authors do not use delexicalized parsers on the target sentence, but rather apply lexicalized parsers on the source sides of a multiparallel corpus (i.e. on source translations of the sentence) and project the edge scores through word alignment. The authors also used the TurboParser [Martins et al., 2010] instead of the MSTParser, so the observations of Mejer and Crammer [2012] might not fully apply to them. Nevertheless, very similarly to us, they did not observe any clear improvements over using the disambiguated parse trees output by the source parsers, rather achieving a slight deterioration on average.

4.3.3 Parse tree projection

For comparison, we also tried to reimplement the method of Agić et al. [2016], based on projection of source parse trees through multi-parallel data. The method is a variant to the classical approach of Hwa et al. [2005], who project source parse trees over word-aligned parallel data, and then employ heuristics to obtain a valid target parse tree. Instead, the authors follow our multi-source tree combination approach, projecting multiple source trees onto the target sentence and combining them via the MST algorithm. Thus, for their method to be applicable, they require multi-parallel data, so that for each target sentence translations in multiple source languages are available, allowing them to use and combine multiple source parsers. However, these are only required in the first step, as once the target side of the multi-parallel data is parsed, a standard parser can be trained on the resulting target treebank and then applied to any target data. The authors also include two potential sources of weighting – edge scores predicted by the source parsers, and alignment scores predicted by the word aligner.

The authors defined the fundamentally same method both for cross-lingual tagging and parsing. For cross-lingual tagging, we found the method to perform rather well, investigated it thoroughly by experimenting with its various hyperparameters (Section 6.1, and applied it with success as one component of our overall approach; however, we found the word aligner scores to actually worsen the results rather than improving them, and therefore use the method unweightedly.

Target lang.	Comb. LAS	Proj., rev. al. LAS	diff.	Proj., int. al. LAS	diff.
da	**47.09**	40.55	-6.54	42.04	-5.05
el	**42.09**	40.04	-2.05	40.22	-1.87
hu	**24.22**	21.59	-2.63	21.36	-2.86
id	**36.11**	32.21	-3.90	32.64	-3.47
ja	6.50	**7.75**	1.25	2.99	-3.51
kk	7.53	7.08	-0.45	**8.28**	0.75
lv	22.86	**23.57**	0.71	22.94	0.08
pl	**46.44**	39.02	-7.42	43.60	-2.84
sk	**48.57**	39.10	-9.47	42.59	-5.98
ta	7.76	**7.92**	0.16	7.76	0.00
tr	13.62	15.56	1.94	**17.07**	3.45
uk	38.17	**38.59**	0.42	31.12	-7.05
vi	**21.47**	19.01	-2.46	15.53	-5.94
AVG	**27.88**	25.54	-2.34	25.24	-2.64

Table 4.7: Comparison of unweighted parse tree combination and parse tree projection, using either reverse alignment or intersection alignment. The difference in LAS obtained by the tree projection versus the tree combination is also shown.

The situation is quite different in cross-lingual parsing, as we observed the method to consistently perform worse than our tree combination method. Thus, we did not take so much care in tuning it for parsing, but rather only evaluated it in its base form. We did not include any of the weighting, as the alignment score weighting performed poorly in tagging, and using source parser scores performed poorly even in the original paper. Besides, we do not have access to edge scores in the parser we use, and, it being a transition based one, there is even no easy way of obtaining them.

In Table 4.7, we compare the performance of the parse tree projection method to the delexicalized tree combination method, using the UD 1.4 dataset and LAS evaluation. We use the unweighted variant of the tree combination, since it is in principle more comparable to the unweighted projection. We use POS tags obtained by the cross-lingual projection method in this experiment.

The results show the parse tree projection to perform worse than the unweighted tree combination, outperforming it only for 5 out of 13 target languages in its reverse-alignment variant, and only slightly – always by less than 2% – while in other cases it typically loses by more than 2%. It also loses on average, also by more than 2%. Moreover, the parse tree projection method is lexicalized in principle, while the tree combination used here is not, as we will only introduce its lexicalization in Chapter 5; this suggests that the tree projection is even somewhat weaker than the results shown here suggest.

We also compared using the reverse alignment, suggested in the original paper, and the intersection alignment, which we found to perform much better in POS tagging. For parsing, using the reverse alignment does perform very similarly to the intersection alignment, leading to a better accuracy for one half of the target languages and to a worse one for the other half, on average reaching LAS higher by 0.3%.

We did not see any massive improvements by softmax-normalizing the edge

scores per each dependent node, which the authors claimed to make a roughly 10% difference in the accuracies. In our experiments, the difference was only around 0.3%. We believe that the softmax normalization becomes much more important when using the edge scores predicted by the source parsers, which is an approach that ultimately did not perform better than only using the resulting source parse trees in the original paper, and we thus did not reinvestigate it here.

4.4 Evaluation

4.4.1 HamleDT 2.0 dataset

We now use the 18 test target language treebanks from the HamleDT 2.0 dataset to evaluate our methods, as opposed to the 12 development language treebanks which we used for tuning. All the 30 HamleDT 2.0 treebanks, with gold POS tags, were used to compute the $KL^{-4}_{cpos^3}$ similarity for each pair of languages, and to train delexicalized MSTperl parsers. Then, for each target language, the other 29 languages were used as potential sources. In the single-source method, only the delexicalized parser trained on the most similar source language treebank is applied to the target data. In the multi-source approach, all of the source delexicalized parsers are applied to the target data. Their outputs are then combined using the tree combination method, either with the contributions of each source weighted by its $KL^{-4}_{cpos^3}$ similarity to the target, or using equal weights for all sources.

Table 4.8 contains the results of applying the delexicalized parser transfer in several setups to the test target treebanks.

Our baseline is the treebank concatenation method of McDonald et al. [2011], i.e. a single delexicalized parser trained on the concatenation of the 29 source treebanks.

As an upper bound,[10] we report the results of the oracle single-source delexicalized transfer: for each target language, the oracle source parser is the one that achieves the highest UAS on the target treebank test section. In this table, we do not include results of a higher upper bound of a supervised delexicalized parser (trained on the target treebank), which has an average UAS of 68.5%. It was not surpassed by our methods for any target language, although it was reached for Telugu, and approached within 5% for Czech and Latin.

The results show that KL_{cpos^3} performs well both in the selection task and in the weighting task, as both the single-source and the weighted multi-source transfer methods outperform the unweighted tree combination on average, as well as the treebank concatenation baseline. In 8 of 18 cases, KL_{cpos^3} is able to correctly identify the oracle source treebank for the single-source approach. In two of these cases, weighted tree combination further improves upon the result of the single-source transfer, i.e., surpasses the oracle. It also always reaches or surpasses the single-best transfer as, in principle, it performs a soft weighted n-best transfer. This proves KL_{cpos^3} to be a successful language similarity measure for delexicalized parser transfer, and the weighted multi-source transfer to be a better performing approach than the single-source transfer.

[10] This is a hard upper-bound for the single-source transfer, but can be (and sometimes is) surpassed by the multi-source transfer.

Target language	Treebank concatenation	Single-source oracle		Single-source KL_{cpos^3}			Tree combination w=1	Tree combination w=$KL^{-4}_{cpos^3}$
bn	61.0	te	**66.7**	0.5	**te**	**66.7**	63.2	**66.7**
cs	60.5	sk	**65.8**	0.3	**sk**	**65.8**	60.4	**65.8**
da	**56.2**	en	**55.4**	0.5	sl	42.1	**54.4**	50.3
de	12.6	en	**56.8**	0.7	**en**	**56.8**	27.6	**56.8**
en	12.3	de	**42.6**	0.8	**de**	**42.6**	21.1	**42.6**
eu	**41.2**	da	**42.1**	0.7	tr	29.1	**40.8**	30.6
grc	43.2	et	42.2	1.0	sl	34.0	**44.7**	42.6
la	38.1	grc	40.3	1.2	cs	35.0	**40.3**	39.7
nl	55.0	da	57.9	0.7	**da**	57.9	56.2	**58.7**
pt	62.8	en	64.2	0.2	es	62.7	**67.2**	62.7
ro	44.2	it	**66.4**	1.6	la	30.8	**51.2**	50.0
ru	55.5	sk	57.7	0.9	la	40.4	**57.8**	57.2
sk	52.2	cs	**61.7**	0.2	sl	58.4	**59.6**	58.4
sl	45.9	sk	**53.9**	0.2	**sk**	**53.9**	47.1	**53.9**
sv	45.4	de	**61.6**	0.6	da	49.8	**52.3**	50.8
ta	27.9	hi	**53.5**	1.1	tr	31.1	28.0	**40.0**
te	67.8	bn	**77.4**	0.4	**bn**	**77.4**	68.7	**77.4**
tr	18.8	ta	40.3	0.7	**ta**	40.3	23.2	**41.1**
AVG	44.5		**55.9**	0.7		48.6	48.0	**52.5**
Std.Dev.	16.9		10.8			14.4	15.0	11.8

Table 4.8: Evaluation using UAS on HamleDT 2.0 test target treebanks.

The weighted tree combination is better than its unweighted variant only for half of the target languages, but it is more stable, as indicated by its lower standard deviation, and achieves an average UAS higher by 4.5% absolute. The unweighted tree combination, as well as treebank concatenation, perform especially poorly for English, German, Tamil, and Turkish, which are rich in determiners, unlike the rest of the treebanks: in the treebanks for these four languages, determiners constitute around 5-10% of all tokens, while most other treebanks contain no determiners at all.[11] Thus, in the unweighted method, determiners are parsed rather randomly[12] – UAS of determiner attachment tends to be lower than 5%, which is several times less than for any other POS. In the weighted methods, this is not the case anymore, as for a determiner-rich target language, determiner-rich source languages are given a high weight.

For target languages for which KL_{cpos^3} of the closest source language was lower or equal to its average value of 0.7, the oracle treebank was identified in 7 cases out of 12 and a different but competitive one in 2 cases; when higher than 0.7, an appropriate treebank was only chosen in 1 case out of 6. When KL_{cpos^3}

[11] In many cases, this is probably related to properties of the treebank annotation or its harmonization rather than properties of the language.

[12] If the source language treebank does not contain determiners, all parser features that contain the POS of the child node inevitably have a zero weight for a determiner. The only non-zero-weight features are thus the parent node POS tag, the distance of the child and the parent, and the conjunction of these two. This very often leads to the determiner being attached to the nearest verb, as verbs are the most fertile nodes and it is not uncommon for them to have both close and distant children.

failed to identify the oracle, weighted tree combination was mostly worse than unweighted tree combination. This shows that for distant languages, KL_{cpos^3} does not perform as good as for close languages.

4.4.2 UD 1.4 dataset

We now proceed to evaluate our methods on the UD 1.4 dataset, using a set of 13 target treebanks and 22 different source treebanks.

As we have already noted, using gold POS tags may be useful in research, but it is not realistic to assume their availability for real under-resourced target languages. Therefore, to show that the methods presented in this chapter can indeed be successfully used in a realistic scenario, we use cross-lingually induced POS tags for the experiments evaluated in this section. For the experiments we report here, we use a POS projection approach very similar to that of Agić et al. [2016], which we already described in Section 4.3.3 for parsing; i.e., the POS tags are projected over word-alignment links from automatically tagged source sides of multiparallel data, and combined with unweighted voting. The method will be described in more detail in Section 6.1.

We also move from the unlabelled MSTParser to the labelled UDPipe parser, and evaluate our approach using LAS instead of UAS; on average, the LAS of the cross-lingual parsing tends to be about 15 points lower than UAS on this dataset.

Source selection

First, we evaluate the source selection based on KL_{cpos^3}. We represent the results of applying KL_{cpos^3} on UD 1.4 treebanks annotated with gold POS, and compare them to using the cross-lingually projected POS.

We also evaluate using the delexicalized POS tagger of Yu et al. [2016], which does not require any resources for the target language other than the text to be tagged. In practice, we did not really apply their tagger to our data, but rather computed KL_{cpos^3} on the tagged data which they released as Deltacorpus [Mareček et al., 2016].

In Section 4.1, we noted that there are other possible methods and information sources related to source-target language similarity. Therefore, we also incorporate WALS Hamming loss (see Section 4.1.3) into the comparison, suggested by Agić [2017].

Table 4.9 compares the possible sources of POS tags to be used for KL_{cpos^3} language similarity computation. Oracle is the upper bound, which always chooses the best performing source language. Gold POS is a theoretical upper bound for the KL_{cpos^3} selection, using the POS tags in source and target treebanks; in practice, we assume that target treebanks are not available. Projected POS compares POS in the automatically tagged source sides of the multiparallel data (using taggers trained on source treebanks), and the projected POS in the target sides. Deltacorpus POS compares POS tags in the Deltacorpus, not using either the source treebanks or the multiparallel data. For each of the POS tag sources, we report the language selected as most similar, and the relative LAS achieved by applying the selected source delexicalized parser to the target data, compared to the oracle LAS: $LAS_{selected}/LAS_{oracle}$. The last line of the table lists average

Target	Oracle source		Gold POS		Proj. POS		Dcorp. POS		WALS	
lang	lang	LAS	lang	% or.	lang	% or.	lang	% or.	lang	% or.
da	sv	47.21	**sv**	100%	**sv**	100%	**sv**	100%	**sv**	100%
el	bg	39.73	en	86%	en	86%	ro	96%	ru	97%
hu	sv	22.11	de	82%	ru	91%	de	82%	fi	97%
id	pt	35.50	he	90%	ro	96%	ar	63%	en	63%
ja	fi	11.75	**fi**	100%	nl	52%	ar	28%	fa	17%
kk	hi	7.08	fi	86%	ru	89%	he	36%	ru	89%
lv	fi	25.00	ru	79%	en	67%	cs	82%	ru	79%
pl	cs	42.98	ru	89%	ru	89%	ru	89%	ru	89%
sk	cs	51.38	**cs**	100%	**cs**	100%	**cs**	100%	**cs**	100%
ta	hi	9.58	**hi**	100%	fi	81%	he	18%	**hi**	100%
tr	fi	20.56	**fi**	100%	**fi**	100%	ar	36%	fa	38%
uk	ru	34.44	**ru**	100%	**ru**	100%	**ru**	100%	**ru**	100%
vi	cs	21.52	sl	88%	ro	90%	ar	72%	en	81%
AVG		28.37	6	92%	4	88%	3	69%	4	81%

Table 4.9: Comparison of using various sources of POS tags for KL_{cpos^3} source selection in cross-lingual delexicalized parser transfer: golden POS tags in training treebanks (not available for target languages in practice), POS tags projected on multiparallel data, and POS tags from the Deltacorpus. Using WALS Hamming loss instead of KL_{cpos^3} is also reported. For each method, the language selected as most similar is listed, along with the oracle rate, i.e. the proportion of the oracle LAS score obtained by using the selected language as the source (% or.).

values for numeric columns; for the language columns, it lists the number of times the oracle language was correctly identified.

As these results were obtained on a different dataset than when developing and evaluating KL_{cpos^3} itself (different tagset, very different dependency annotation style...), we would first like to note that the evaluation once again shows the high quality of the KL_{cpos^3} measure. When computed on gold POS tags, it manages to identify the oracle source language in nearly half of the cases, including some difficult ones (*fi-ja, fi-tr, hi-ta*). When the oracle is not identified, the resulting accuracies are still highly competitive, with the worst result being 79% of oracle for *ru-lv*.

As KL_{cpos^3} was originally developed and evaluated using gold POS tags only, there were some concerns, voiced by Agić [2017] as well as several reviewers of our papers, about porting the measure to a more realistic setting where gold POS tags are not available. Specifically, in the setup evaluated here, the tags are projected unweightedly through small out-of-domain multi-parallel data, with an accuracy of only roughly 70%, which is way below supervised taggers, let alone the gold standard manual annotations. However (maybe somewhat surprisingly), the performance of KL_{cpos^3} on the projected tags is nearly flawless, with an average drop of only 4% absolute. More surprisingly, it even surpasses the gold-POS setup in 4 cases, presumably due to the fact that the projected POS tags eventually get to be used on the parser input, and may therefore be more informative than the theoretically better but practically unreachable gold POS tags. The oracle language is now identified in only 4 cases, while using gold POS tags led to identifying the oracle in 6 cases – for Tamil and Japanese, the method now selects

a (strongly) suboptimal source language, leading to large drops in accuracy.

The performance of KL_{cpos^3} on the Deltacorpus delexicalized tagger POS tags is clearly much lower, although the results are still competitive for about a half of the targets. On one hand, we expected the performance to be even worse, as the tagging accuracy in Deltacorpus is only around 60%. On the other hand, this is not that much lower than the projection tagging, while the accuracy drop is much larger, so we believe that with some more tuning, the delexicalized tagging approach might be brought to a state of usefulness for cross-lingual parsing. Also, while the final POS tags that the method outputs may not be optimal, it may be that the internal representations and/or classifier output distributions contain more useful information which could be leveraged.

In general, the results show that KL_{cpos^3} is a rather robust measure, consistently performing well across various datasets with various qualities of POS tags.

We can also confirm the findings of Agić [2017], who found KL_{cpos^3} to perform better than WALS Hamming loss. While both of these methods identify the oracle language in 4 cases, WALS makes worse mistakes in cases where it does not identify the best language, leading to a lower average score. On the other hand, employing WALS leads to better results than computing KL_{cpos^3} on the Deltacorpus POS tags, while both of these methods can do without parallel data, making them potentially useful in a scenario where parallel data are even more scarce, or non-existent. WALS also has one nice win, returning Finnish as the most similar language to Hungarian, which does make much sense and which the POS-based method was unable to detect.

Source combination

We now proceed to evaluate the multi-source parse tree combination method on cross-lingually projected POS tags.

Table 4.10 shows the LAS achieved by the tree combination, both unweighted and weighted using $KL^{-4}_{cpos^3}$, as well as the single-source approaches using either the oracle source treebank or the source selected with KL_{cpos^3}.

The first thing we notice are the overall low scores, with the average LAS around 30%, which seems to be much lower than what we saw on the HamleDT 2.0 dataset. However, this is mostly an illusion, as the apparent drop is mainly due to us moving from UAS to LAS: the average UAS on the UD 1.4 dataset (not shown in the table) is around 45%, which is not that much lower than the averages on the HamleDT 2.0 dataset, which were around 50%. In other words, it is not that the results suddenly got much worse; they already were this low before, only this was partially hidden by using a different evaluation metric. Thus, the use of cross-lingually predicted POS instead of gold POS only leads to a slight deterioration of the (already quite low) results.

A real change to be noticed is the fact that the differences between the methods diminished – while the distance of the weighted tree combination from the oracle transfer is roughly similar as on the HamleDT 2.0 dataset, it is now very closely followed by the single-source transfer using the automatically selected source treebank, and also the unweighted tree combination performs only slightly worse.

On the other hand, the results are much more stable: on the HamleDT 2.0 dataset, unweighted tree combination actually reached the highest UAS among

Target language	Single-source oracle	Single-source KL_{cpos^3}	Tree combination w=1	Tree combination w=$KL^{-4}_{cpos^3}$
da	**47.44**	**47.44**	47.09	47.17
el	41.94	39.81	**42.09**	41.86
hu	24.45	22.71	24.22	**26.00**
id	36.40	34.64	36.11	**36.58**
ja	**17.27**	8.63	6.50	7.36
kk	**7.83**	6.93	7.53	**7.83**
lv	**30.19**	22.28	22.86	22.97
pl	**47.57**	43.31	46.44	46.70
sk	**49.94**	**49.94**	48.57	49.57
ta	**13.62**	**13.62**	7.76	10.29
tr	**22.82**	**22.82**	13.62	15.35
uk	35.27	35.27	**38.17**	**38.17**
vi	**21.60**	20.75	21.47	**21.61**
AVG	**30.26**	28.32	27.88	**28.57**
St. Dev.	13.82	14.42	16.01	15.61

Table 4.10: Evaluation using LAS on UD 1.4 target treebanks.

the non-oracle methods for half of the target languages (even though being worse on average), while on the UD 1.4 dataset, this only happens in 1 case (*el*); for all other targets, the weighted tree combination reaches a better score (or an identical one for *uk*).

We hypothesize that this stabilization of the results is due to all of the POS tags being induced very similarly (or actually identically), while on the HamleDT 2.0 dataset, we used the gold POS tags, which may still have been influenced by shortcomings in the treebank harmonization.

Otherwise, the overall picture is mostly similar as on the HamleDT 2.0 dataset.

The weighted multi-source tree combination is better both than single-source transfer and unweighted tree combination, both on average as well as in most cases, leading to the best performance for 7 of the 13 targets, while the single-source transfer performs best in 5 cases and the unweighted combination in 2.

What we unfortunately lose is the previously observed property of weighted tree combination not spoiling the single-source transfer in cases where we were able to identify the oracle source with KL_{cpos^3}. On the HamleDT 2.0 dataset, the weighted tree combination was never worse than the single-source transfer. However, on the UD 2.0 dataset, the score differences are lower, and the weighted combination does in fact fail to reach the score of the single-source transfer in 5 cases. While the difference is rather negligible in 2 cases (less than 0.5 absolute for *da* and *sk*), it is unpleasant in the other 3 cases, with a particularly large drop for *tr*. Moreover, 4 of the deteriorations occur with target languages for which KL_{cpos^3} was actually able to identify the oracle source; this may well be the cause itself, as the set of languages we use here is more varied, and so if there is indeed a good and easy-to-identify source language, the other sources may be mostly irrelevant.

On the other hand, while the oracle single-source transfer stays best on average, it is surpassed by the tree combination in 4 cases, which is something that

we observed on the HamleDT 2.0 development set but not on the test set.

To conclude the evaluation, we observed some changes when moving from the unlabelled parsing with gold POS tags on the HamleDT 2.0 dataset to the labelled parsing with cross-lingually projected POS tags on the UD 1.4 dataset. While some of these changes are most probably due to the different setup, such as the higher stability of the scores which we attribute to the higher uniformity of the POS tags, other may simply be because of the different set of source and target languages. Nevertheless, the overall results give the same message, showing both KL_{cpos^3} and the parse tree combination to be good approaches to cross-lingual parsing, and the parse tree combination weighted by $KL^{-4}_{cpos^3}$ source-target similarity to be the best performing setup.

4.5 Summary

In this chapter, we introduced KL_{cpos^3}, our language similarity measure based on the KL divergence of POS trigram distributions in the source and target languages, and showed it to perform well in selecting an appropriate source language for a given target language to use in single-source delexicalized parser transfer.

We then presented several methods for combining multiple sources together to perform multi-source delexicalized parser transfer, with parse tree combination performing best in our experiments. We further enhance the method by weighting the sources by their $KL^{-4}_{cpos^3}$ similarity to the target language.

Finally, we evaluated our approach in a somewhat different setting, using a different set of treebanks in a different annotation style, as well as cross-lingually predicted POS tags instead of supervised ones, and found it to perform well even under such conditions.

5. Cross-lingual Lexicalization

So far, the basis for our cross-lingual parsing has been the delexicalized parser. However, as we already showed in Section 3.1, omitting the lexical information leads to a noticeable drop in parsing accuracy, as the POS tags are often not sufficient by themselves to unambiguously expose the syntactic structure of the sentence. Therefore, in this chapter, we deal with lexicalizing the cross-lingual parsing.

There is in fact a very wide range of directions from which lexicalized cross-lingual parsing can be approached. We try to review them in Section 5.1, including observations from preliminary experiments that we conducted for many of them; however, in the rest of the chapter, we only deal with the approach which we eventually chose, based on applying MT methods to the source language treebanks.

We first present two very basic initial approaches, applicable for pairs of very close source and target languages: source-lexicalized parsing (Section 5.2) and character-level transformations (Section 5.4); we also explored simple string-similarity-based translation in [Rosa, 2017].

We then move on to using full-fledged MT systems in Section 5.5, automatically translating the source treebank into the target language, and then training a rather standard lexicalized parser on it. We explore various setups, and finally settle on the GIZA++ word aligner [Och and Ney, 2003] with intersection alignment and the Moses MT decoder [Koehn et al., 2007] in a monotone word-based setting, i.e. translating each source word to exactly one target word without any reordering.

Such an approach clearly has its limits, as even in close languages, words do not correspond 1:1 (let alone in distant languages) and systematic differences in word order are also common, making the word-based monotone translation sub-optimal in terms of intrinsic translation quality. However, we have found this approach to have two important benefits which seem to outweigh that. First, it makes the annotation transfer extremely simple and thus less noisy. And second, it forces the MT system to produce more literal translations, keeping the structure of the target sentence very similar to the source sentence, which increases the chance of the source annotation to be valid for the translation as well.

5.1 Overview of possible approaches

5.1.1 Projection over parallel data

In the founding work on cross-lingual parsing, Hwa et al. [2005] trained a standard lexicalized parser on a source language treebank, used it to parse the source side of a parallel corpus, and then projected these automatic annotations onto the target side of the corpus, thus obtaining a synthetic target-language treebank, which was then used to train a target-language parser. As the parallel sentences typically do not align 1:1, and because the target annotations need to form valid syntactic trees, complex heuristics have to be employed in the transfer. Later, Agić et al.

[2016] successfully extended this approach to a setting with multiparallel corpora.

We have briefly explored this approach in Section 4.3.3, but found it to perform even worse than the delexicalized parser transfer in our experiments. We think this is because the noise introduced into the setup by using automatically induced source annotations combines with the noisy transfer heuristics, leading to a too low quality of the resulting synthetic target treebank.

However, for cross-lingual POS tagging (Section 6.1), this approach seems to perform quite well – presumably because there are no such strict constraints on the validity of a POS tag sequence as there are for a parse tree, thus avoiding the complex and noisy transfer heuristics. We thus incorporate the projection approach into our final cross-lingual POS tagging setup (Section 6.3).

5.1.2 Machine translation approaches

Machine translation of source treebank

Tiedemann et al. [2014] introduce the approach of using a full-fledged SMT system to translate the word forms in a source language treebank (i.e. annotated source language sentences) into the target language. In this way, they again obtain a synthetic target-language treebank, which can be used to train a standard lexicalized parser.

Depending on the type of the MT system used, the transfer of the source sentence annotations onto the target language sentences produced by the MT system can be quite simple or quite complex. Tiedemann et al. [2014] explore multiple setups, but eventually decide for a complex SMT system, which requires the use of transfer heuristics similar to those of Hwa et al. [2005], and follow this path in their further works [Tiedemann, 2017]. In our work, on the contrary, we follow the other option which they had explored, using a word-based SMT system, which achieves a lower quality of the translation as measured in BLEU [Papineni et al., 2002], but makes it possible to transfer the source annotation by simply copying it over the 1:1 word alignment, thus avoiding the need for noisy transfer heuristics.

The approach of machine translating the source treebanks has the advantage of directly employing the manually annotated resources, which can be expected to be of high quality with only little noise. Of course, the translations provided by an MT system are inherently noisy, containing mistranslations, untranslated words, etc., which is bound to introduce noise in the process. On the other hand, the MT outputs can be expected to structurally correspond better to the source sentences than human translations, as they tend to be more literal, which might actually make the annotation transfer more reliable than if human translations were used.[1] Also, additional monolingual target-language texts are trivial to incorporate into this approach, using them to enrich the training data for the target language model.

[1] Of course, this assumes a certain level of quality of the machine translation; if the available parallel corpora are too small, there is a high risk of the quality of the translation actually being too low to make any assumptions about it.

Machine translation of target sentence

Ramasamy et al. [2014] used the MT system in the other direction, translating each input target-language sentence into the source language, analyzing it with a source language parser, and then transferring the predicted parse tree back onto the original target-language sentence.

In our preliminary experiments, this approach performed worse than using MT in the other direction, and so we do not follow that here. We think that the major shortcoming of that method is that it uses both synthetic parallel data (MT outputs) and automatic parsing, thus containing two steps that are inherently noisy, while the two previously mentioned methods use either real parallel data, or manually annotated parse trees, which presumably reduces the noisiness of the transfer.

Machine translation into a pivot language

In principle, it is of course also possible to apply MT both to the source-language treebank and to the target-language sentence, translating them into a common pivot language; this could either be a real third language, or a synthetic one. This will inevitably lead to an even larger accumulation of noise in the pipeline, and we thus do not expect such an approach to lead to state-of-the-art results. Still, we explore a simple variant of this method in Section 5.4, where we exploit pairs of languages with a high degree of lexical similarity, trying to simplify the word forms used in both of them in the same way to bring them closer together. However, these experiments are rather exploratory.

5.1.3 Using cross-lingual clusters

Täckström et al. [2012] devised an approach which is somewhat similar to the idea of translating both languages into a common pivot language. In their work, they utilize word clustering approaches and parallel data to obtain a set of 256 cross-lingual word clusters, mapping both source and target words into this set of clusters while trying to always map corresponding words into the same cluster. Then they use the word clusters instead of word forms as input features to train a parser on a source language treebank, and apply it to a target language sentences where words are again replaced by their clusters.

The cross-lingual clusters introduce lexical information into the cross-lingual parsing in a rather coarse-grained way; they could also be viewed rather as an alternative fine-grained POS-like tagset. We thus consider this approach to be somewhere between delexicalized and lexicalized parsing.

5.1.4 Using word embeddings

In recent years, word embeddings [Bengio et al., 2003, Mikolov et al., 2013] have become the favourite way of lexicalizing NLP tools, with parsing being no exception. Relating to the work of Täckström et al. [2012], we can probably view word embeddings as a sort of soft multidimensional word clusters.

As we use the UDPipe parser in our experiments, which is a neural parser and uses word embeddings as the representations of the input words (Section 2.2.2),

this opens up several further options for lexicalized cross-lingual parsing for us.

Monolingual word embeddings

In the monolingual setting, the UDPipe Manual[2] recommends pre-training word embeddings on larger plaintext data, if available, as this leads to higher-quality embeddings than when they are only trained on the small treebank (for the purposes of training word embeddings, all treebanks are small).

Obviously, this does not lead to the cross-lingual lexicalization of the parser by itself – one of the other approaches discussed in this section has to be used for that first. However, in [Rosa et al., 2017], we have found that once the lexicalization has been achieved, providing the parser with word embeddings pre-trained on target language data can serve as a further adaptation of the parser to the target language. We discuss and evaluate this approach in Section 5.3.

Multilingual word embeddings

In the multilingual or cross-lingual setting, many authors have explored various ways of inducing bilingual or multilingual word embeddings [Xiao and Guo, 2014, Guo et al., 2015, Zhang and Barzilay, 2015, Gouws and Søgaard, 2015, Duong et al., 2015, Ammar et al., 2016].

However, we have not found any work that would report substantial improvements achieved in this way for the cross-lingual parsing task, especially in comparison to using an MT system; the review work of Søgaard et al. [2015] explicitly concludes that the multilingual word embedding approaches do not seem to bring much for this task. Our preliminary experiments are in agreement with that, and we thus do not follow this path in our approach.

However, there has recently been tremendous progress in performing machine translation with little or no parallel data, based on the observation that the vector spaces of word embeddings tend to have a similar shape in all languages, and thus independently obtained monolingual word embeddings seem to be possible to map onto each other using a linear transformation with local adaptations, based on tiny or even no parallel data [Artetxe et al., 2017, Conneau et al., 2017]. We find such findings to be fascinating and are curious about their potential future applicability to cross-lingual parsing of under-resourced languages.

5.1.5 Translating the parser model internals

It is also possible to translate the lexical features of a trained parser model itself, e.g., if it is a neural parser, by using the bilingual or multilingual word embeddings approaches; however, this approach could also be used with non-neural parsers.

We conducted some preliminary experiments with UDPipe, either translating each word in the vocabulary of the parser into its most probable counterpart, or devising an embedding for each target language word as a linear combination of the embeddings for its source language counterparts. Again, we did not find this approach to perform well. It is not clear to us in what way this could be better than using MT to translate the source treebank, which allows the use of

[2] `https://ufal.mff.cuni.cz/udpipe/users-manual#udpipe_training_parser_embeddings`

standard monolingual target language word embeddings in the process, while the translation of whole sentences (in the treebank) is bound to be more accurate than the translation/transfer of individual words without context (in the word embedding space).

5.1.6 Using subword units

Another approach which is also becoming the *de facto* standard for language processing in neural networks is the splitting of the input words into subword units, either linguistically motivated (morphs/morphemes), as performed e.g. by Morfessor [Creutz and Lagus, 2005], or, more often, linguistically agnostic, based e.g. on convolutional approaches or compression algorithms, such as Byte Pair Encoding (BPE) [Gage, 1994, Sennrich et al., 2016]. Again, these approaches do not achieve the cross-lingual lexicalization by themselves – another method from this section must be applied first.

While we have experimented with both of these approaches – applying either Morfessor or BPE to split words into smaller units – we have found it hard to employ them for either monolingual or cross-lingual parsing; splitting the words into smaller units always led to a decrease of parsing accuracy in our preliminary experiments. Based on manual inspections of the outputs, we now think that the problem is that the concept of the word as an atomic unit is very central to dependency parsing, and splitting words into smaller units seems to add a lot of noise into the processing without bringing any clear improvements to counterweigh that.

In cross-lingual parsing, we hoped that using subwords could prove to be a win-win solution to the question of using accurate M:N translation versus simple 1:1 alignment that allows high-precision structure transfer – with an appropriate splitting of the texts into subwords, one could theoretically achieve a 1:1 alignment for sufficiently close languages with little compromise. However, in our experiments, we found that this is very hard to do correctly, and also makes the processing even noisier. Moreover, most "parallel" sentences in typical parallel corpora are not literal translations of each other, where we could hope for a nearly 1:1 alignment of appropriate subword units, but rather free paraphrases of each other, where no such transformations can make the sentences easy to align 1:1.

5.2 Source-lexicalized parsing

To the best of our knowledge, source-lexicalized parsing was first explicitly introduced as a viable approach for cross-lingual parsing by Zampieri et al. [2017] in the VarDial 2017 Cross-lingual parsing shared task, with the organizers providing both a delexicalized transfer baseline as well as a second, slightly stronger, source-lexicalized parser baseline.

Interestingly, this approach is even simpler than the delexicalized parser. In cross-lingual source-lexicalized parsing, a standard lexicalized parser is trained on the source treebank, and then applied to target language without any adaptation. If there is a non-trivial overlap of the word forms used in the source and target language, such an approach can leverage these and achieve a better parsing accuracy. For even better results, the source parser should be trained only on POS

Setup	cs→sk	da+sv→no	sl→hr
Delexicalized	53.66	57.84	53.93
Source-lexicalized	54.61	59.10	56.85
Difference	+0.95	+1.26	+2.92

Table 5.1: Comparison of delexicalized and source-lexicalized parser transfer for the very close UD 1.4 VarDial languages dev-sets, as reported by Zampieri et al. [2017], using LAS and gold POS tags.

tags and word forms, i.e. excluding other morphological annotation, which seems to be typically insufficiently cross-lingual.

In Table 5.1, we re-present the LAS scores observed by Zampieri et al. [2017] on the UD 1.4 VarDial dataset (Section 1.3.2), showing that for very close languages, it can outperform delexicalized parser transfer by several points in LAS.

As the main assumptions are that the source and target language share a reasonable amount of word forms which have similar meanings in both of the languages, this approach cannot be used but for very close languages, making it weaker than delexicalized parsing in this respect. Still, on very close language pairs, source-lexicalized parsing usually achieves higher accuracies than a delexicalized parser.

On the other hand, source-lexicalized parsing can also be viewed as being stronger than delexicalized parsing, since it does not require target-side POS tags: if these are not available, we can still train a *detagged* parser, which uses only word forms as its features. However, we have found the detagged parser to perform much worse than the delexicalized parser, even if the delexicalized parser is applied to cross-lingually induced POS tags.

In Section 5.4, we build upon the source-lexicalized parser transfer approach by targeting cases where the spelling of corresponding source and target words is often not identical, but still somewhat systematically similar, achieving further improvements. We then compare these methods to the source-lexicalized baseline in Section 5.4.1.

In fact, in our work, we further improve the method by pre-training target-side word-embeddings, as explained in Section 5.3, and use those when training the source parser, hoping that for each target word that is identical to a source word appearing in the training treebank, the parser also learns how to parse similar target words (with similar word embeddings) – which may even be non-existent in the source language, let alone the source treebank.

5.3 Monolingual word-embeddings

As we are using a neural parser (UDPipe), all input features, especially the word forms, have to be converted to vectors (embeddings). By default, UDPipe trains embeddings of each input feature on the training treebank jointly with training the parser. However, this means that word form embeddings are typically trained on very small data, leading to a low quality of the embeddings as well as potentially high out-of-vocabulary rates, which is unfortunate since larger monolingual plaintext data are usually available. Therefore, the UDPipe Manual recommends pre-training word embeddings on larger plaintext data, if available, even in the

	cs→sk		da+sv→no		sl→hr	
Setup	LAS	Δ	LAS	Δ	LAS	Δ
Base	61.33		64.03		59.47	
Pre-train	66.91	+5.6	66.49	+2.5	61.95	+2.5

Table 5.2: Evaluation of the effect of pre-training target word embeddings on large monolingual data, versus the default of training only on the target translation of the source treebank.

supervised monolingual case.[3]

In the cross-lingual setting, this has an additional benefit of using real target language data in the training of the parser, thus letting it, indirectly through the word embeddings, learn something about what sentences in the target language actually look like. Note that the default UDPipe approach is even more problematic in our setting, since we do not have a real target treebank at our disposal, and so target embeddings trained on the pseudo-target treebank can be expected to be of an even lower quality than in the fully supervised monolingual setting. In that sense, pre-training the word embeddings can serve as an additional target language adaptation.

For pre-training the embeddings, we use word2vec [Mikolov et al., 2013], with the parameters suggested in the UDPipe Manual.[4] We use the target side of the parallel data to train the embeddings.

In Table 5.2, we evaluate the effect of pre-training word embeddings on the target side of OpenSubtitles2016 on the VarDial dataset, employed in a MT-lexicalized cross-lingual parser transfer applied on supervised target POS tags. The results clearly show the huge effect of this simple trick, improving LAS by more than 3 points on average. While the available data were quite big in this case, more moderate improvements are to be expected with smaller parallel data, which can more realistically be expected to be available for an under-resourced language. Nevertheless, we do not see how employing the word embeddings pre-training could ever harm the performance, and therefore believe it to be worth performing in all situations.

5.4 Character-level transformations

In this section, we investigate several attempts to slightly improve the source-lexicalized parsing by simple character-level transformations of the source and target word forms.[5] We specifically target a similar case as the source-lexicalized parsing, where corresponding words in the source and target language are very similar, but e.g. may have a slightly different spelling, or the inflection patterns differ slightly, etc.

This approach can be regarded as a simple heuristical translation of both the source and the target language into a simplified common "pivot language", which

[3] `https://ufal.mff.cuni.cz/udpipe/users-manual#udpipe_training_parser_embeddings`

[4] `-cbow 0 -size 50 -window 10 -negative 5 -hs 0 -sample 1e-1 -binary 0 -iter 15 -min-count 2 -threads 12`

[5] This is joint work of Rosa et al. [2017], with substantial contributions made by Daniel Zeman, Zdeněk Žabokrtský, and Rudolf Rosa.

is constructed by keeping the features shared among the languages and discarding those in which they differ.

We have tried this approach for the VarDial language pairs: Czech→Slovak, Slovene→Croatian, and Danish+Swedish→Norwegian (Section 1.3.2). We noted that in many cases, corresponding words in the languages look similar, but are not completely identical, suggesting a space for improvement over the source-lexicalization baseline by the envisioned method.

We noted that some of the differences seem regular, and could be targeted by a rule-based translation system operating on subword level, but assumed this would be a lot of work of unclear value – we only wanted to do a simple quick experiment evaluating the viability of such a heuristical approach, with the intention of ultimately moving on to SMT. Moreover, we wanted to design a set of transformations applicable to multiple languages, while the rule-based translator would inevitably be bound to a fixed language pair.

In particular, we exploit the fact that for similar words that correspond to each other in the VarDial languages, we observed a high interlingual variability in vowels, in diacritical marks, in duplication of letters, and in word-final inflection. This motivates the introduction of the following five simple transformations:

Removing diacritics Replace each letter with diacritical marks by its variant without the diacritical marks, using the Unidecode library[6] by Burke [2001] – e.g. "caffé" → "caffe"

Removing vowels Delete the letters *a, e, i, o, u, y*, with or without diacritical marks – e.g. "caffé" → "cff"

Removing duplicated letters If a letter is duplicated in a word, we remove the duplication – e.g. "caffé" → "café"

Simple stemming Keep only the first N letters from each word – e.g., for stem3, "caffé" → "caf"

Lemmatization Instead of the word form, use the morphological lemma annotated in the treebank.

5.4.1 Evaluation

We evaluated the effects of the aforementioned character transformations on the three language pairs used in the VarDial workshop [Rosa et al., 2017, Zampieri et al., 2017], using the parser transfer from Czech to Slovak, Slovene to Croatian, and Danish+Swedish to Norwegian. The results are shown in Table 5.3. As the baselines, we compare our results to the delexicalized parser (Section 3.2), as well as the source-lexicalized parser (Section 5.2); as the source and target languages are very close, source-lexicalized parsing is consistently better than the delexicalized parsing for all of the language pairs.

We first try to further improve the source-lexicalization by switching to lemmas or simple stems (of length 6) instead of word forms, but the results are generally unfavourable: we observed minor improvements for the Slovene to Croatian parser transfer only, with the simple stemming looking rather promising. The lemmatization performs especially poorly, which may be due to the definitions of

[6] `https://pypi.org/project/Unidecode/`

Setup	cs→sk LAS	cs→sk Δ	da+sv→no LAS	da+sv→no Δ	sl→hr LAS	sl→hr Δ
Base	Δ = improvement versus delexicalized					
Delexicalized	52.4		58.9		53.2	
Source-lexicalized	**56.0**	**+3.6**	**59.3**	**+0.4**	**55.9**	**+2.7**
Simple changes	Δ = improvement versus source-lexicalized					
lemma	51.5	-4.5	56.5	-2.8	56.2	+0.3
stem6	54.8	-1.2	58.8	-0.6	56.8	+0.9
-dia -dup	56.3	+0.3	58.0	-1.4	55.4	-0.5
-vow	**58.2**	**+2.2**	**62.4**	**+3.1**	**57.1**	**+1.3**
-vow -dia -dup	57.7	+1.7	61.9	+2.6	55.4	-0.5
Combinations	Δ = improvement versus -vow					
-vow stem3	58.7	+0.5	62.3	-0.2	56.7	-0.5
-vow stem4	57.4	-0.8	62.5	+0.0	56.7	-0.5
-vow stem5	58.2	+0.0	61.9	-0.5	56.1	-1.0
-vow -dia -dup stem3	58.6	+0.4	61.7	-0.8	57.0	-0.2
-vow -dia -dup stem4	58.8	+0.6	62.0	-0.5	56.0	-1.2

Table 5.3: Evaluation of various simple lexicalization strategies: lemmatization (lemma), simple stemming by clipping to N characters (stem$N$), removing vowels (-vow), removing diacritics (-dia), and removing duplicated letters (-dup).

the lemmas being different in the treebanks;[7] it may also be the effect of using the pre-trained word embeddings (Section 5.3), which we expect to perform a sort of soft lemmatization, among other.

We then proceed with removing vowels, diacritics and duplicated letters. Removing vowels consistently performs very well, adding around 2 LAS points on average. On the other hand, removing diacritics nearly always worsens the performance; removing duplicated letters mostly had only a negligible effect, which is why we evaluate it together with removing diacritics, as we see those transformations as being quite similar in principle. When these approaches are combined, the results are consistently worse than that of only removing the vowels.

Finally, we try to combine removing vowels with the simple stemming, this time experimenting with the prefix size. The words are already considerably shorter due to the removal of the vowels, we therefore try low prefix lengths: 3, 4, and 5. We also try to remove the diacritics and duplicated letters again. However, we do not observe any large and consistent improvements in comparison to the removal of vowels alone. There are some small positive numbers for the Czech to Slovak transfer, but they seem to be rather random fluctuations than real solid improvements, as we have not been able to identify any clear pattern in the results. Furthermore, we can see that the originally promising result of simple stemming for the Slovene to Croatian transfer disappeared when applied on top of the vowel removal, suggesting that after removing all vowels from the suffixes,

[7]For example, we have noticed that the negation prefix "ne-" is stripped in lemmatization in Czech but kept in Slovak. Thus e.g. the word "nezabalená" ("not packed" in feminine gender), which happens to be identical in both Czech and Slovak, is lemmatized as "nezabalený" ("not packed" in masculine gender) in Slovak, but as "zabalený" ("packed" in masculine gender) in Czech. If a parser is trained on Czech lemmas, all lemmas of negated Slovak words will thus be unknown to it.

they already become sufficiently similar, and further removal of the consonantal remains of the suffixes is not beneficial.

Thus, we conclude that from the simple character-level transformations that we tried out, the only one that leads to large and consistent performance improvements across all of the evaluated language pairs is removing vowels.

We acknowledge that these transformations are very simple, and much more sophisticated approaches could be taken: for example, one could employ proper character-based machine translation, as done e.g. by Durrani et al. [2010]. Still, the performance of any character-based methods has a clear (and usually rather low) upper bound, as the character-level similarity even between close languages is rather limited, with many corresponding words that are dissimilar, many similar words that have different meanings, etc. For this reason, we move on to word-based machine translation in the next section. However, we believe that there is some potential value even in the character-based transformations, especially in situations where the source and target languages are extremely close, and the amount of parallel data usable for training a word-based MT system is very limited; in such cases, using character-level transformations, either alone, or in combination with word-based MT, as done e.g. by Durrani et al. [2014], could perform well.

5.5 Machine translation

We now move to proper machine translation, based on models learned from parallel data.

Our final setup is based on using *word-based* SMT in an otherwise rather standard setting[8] – employing GIZA++ word aligner [Och and Ney, 2003] with intersection alignment symmetrization,[9] phrase table extraction with phrase length fixed to 1,[10] Moses decoder [Koehn et al., 2007] in a monotone setting,[11] and the KenLM language model [Heafield, 2011] with trigrams,[12] trained on the target side of the parallel data. However, we explore several alternatives and variations to this setup, both theoretically as well as, in some cases, empirically.

5.5.1 Translation arity

In the previous section, we noted that there are multiple options for the MT system setup, with implications to the annotation transfer process. Here, we will particularly focus on the problem of *translation arity*, i.e. whether each source word is always translated into exactly one target word – we will refer to that approach as word-based translation – or whether a source word or a group of source words can be translated into one or multiple target words (phrase based translation).

In the case of the simpler word-based MT approaches, a 1:1 correspondence between the source and target words can be ensured, and the annotation can then

[8] `http://www.statmt.org/moses/?n=Moses.Baseline`
[9] `-alignment intersect`
[10] `-max-phrase-length 1`
[11] `-distortion-limit 0`
[12] `-o 3`

be trivially copied onto the target sentence without any modifications. However, the word-based approach is bound to produce a lower quality translation, as even for very close languages, some expressions cannot be translated 1:1.

If standard Phrase-Based Machine Translation (PBMT) is used, the quality of the translations is higher, but the source-target word correspondence is rather N:M (phrase to phrase). In that case, a set of heuristics, such as those proposed by Hwa et al. [2005], must be used to map the source annotations onto the target sentence, together with a way of estimating the internal alignment of the phrases. This makes the annotation transfer inevitably somewhat noisy.

Tiedemann et al. [2014] investigated both of these approaches, using a standard PBMT setup, based on GIZA++ and Moses. In the phrase-based translation approach, they use a vanilla Moses setup, and then project the source annotations using heuristics similar to those introduced by Hwa et al. [2005], but utilizing additional alignment information provided by the MT system for a more precise annotation projection. To perform word-based translation, they simply filter the phrase table to only contain one-word phrases, and then proceed in the standard way. Word reordering can still be performed by the MT system, but this does not complicate the annotation transfer much, as Moses outputs the source-target 1:1 word alignment links. Surprisingly, the authors found both of the setups to perform very similarly in their setting, reaching accuracies within 0.1 LAS point on average from each other.

An observation which we believe to be important is that while SMT systems are typically tuned to produce translations best suited to be read by humans, this may not be the right target in our case. What we are actually interested in is a translation that is reasonably accurate, but also enables a reliable transfer of the annotation. Moreover, in Section 5.6, we will actually show that the cross-lingual parsing accuracy does not seem to be too sensitive to the translation quality, as measured by BLEU.

Based on our investigations of the translation outputs produced by the various configurations of the MT system, we observed the word-based translations to be somewhat cumbersome and often containing several garbage function words,[13] but on the other hand much more literal and therefore better preserving the source sentence structure. The phrase-based translations tend to be more fluent and generally better, but also less literal, changing part of speech and language structures to fit what is preferred in the target language, thus making the source morphological and syntactic annotation less transferable onto the output, even if it was not for the noisy M:N transfer heuristics.

Thus, contrary to the case of standard human-targeted translation, but in accord with findings of Tiedemann et al. [2014], we consider word-based MT not to be inferior to phrase-based MT in our setting, but to constitute a viable and competitive alternative, at least in the case of somewhat similar languages. Moreover, employment of phrase-based MT into cross-lingual parsing is rather well studied, while the previous research on employing word-based MT is rather rudimentary, uses sub-optimal approaches and does not exploit even the standard features available in GIZA++ and Moses. Therefore, we decided not to deal with phrase-based MT in our research, but to rather focus on word-based MT, investi-

[13] Typically "translations" of source function words which should not have any counterpart in the target language.

gating the potential of employing state-of-the-art word-based MT approaches in cross-lingual parser transfer.

We would like to note here that both word-based and phrase-based MT can but does not have to be monotonous (a monotonous MT system is one which does not perform word reordering during the translation). We discuss that issue and investigate the effect of translation monotonicity in Section 5.5.3.

5.5.2 Word alignment

For word alignment, by default, we use the standard GIZA++ tool in the usual way, i.e. computing both forward and reverse alignment, and then combining them via symmetrization.

Alignment symmetrization

The word alignment systems offer many flavours of the word alignment, and it is far from clear which one should perform best. GIZA++ itself only performs the word alignment in one direction. In the default *forward* alignment, each target word is aligned to exactly one source word, while each source word can be aligned to 0 or more target words; and vice versa for the *reverse* alignment direction. The logic behind the naming is that if one translates in the direction from the source language into the target language, one needs to produce all of the target words exactly once, so that the output is identical to the target sentence, while it does not matter how the source words are used – some can be translated into multiple target words, and some can be ignored. However, later it was found out that both the forward and the reverse alignments are useful for SMT, while none being of high quality on its own. Therefore, symmetrization heuristics are typically used to combine the two alignments into one of higher quality [Och and Ney, 2003].

The simplest two symmetrizations applied to alignment links are:

union Keep all links that appear either in the forward or in the reverse alignment.

intersection Keep only links that appear in both forward and reverse alignment.

However, as these still do not usually perform very well in PBMT, a set of more sophisticated heuristics was devised, all starting with the intersection alignment as their backbone and iteratively adding some of the most likely alignment links from the union alignment. In particular, the grow-diag-final-and (*gdfa*) symmetrization is usually used in SMT.[14]

Tiedemann [2014] investigated the main symmetrization approaches in applying MT to parser transfer, concluding that the effect of the particular method used is not very strong, and that no general conclusion about which method performs best in this scenario can be drawn from the results, as different methods ranked differently for different language pairs. Specifically, the *gdfa* symmetrization approach was found to perform well on average and often led to the best LAS scores, but its advantage was not crucial.

In our work, we decided to use the *intersection* symmetrization. The main advantage of this method is that it natively produces a 1:1 word alignment,

[14] `http://hermes.fbk.eu/people/bertoldi/teaching/img/Word_Alignment_Symmetrization.pdf`

making it easily applicable to our setting, as word-to-word translation pairs can be directly extracted from it. It is also typically described as strongly preferring alignment precision over recall, which might be preferable in our case: we wish to use cross-lingual lexicalization of the parser only to improve its performance, so we might want to focus only on quite reliable translations. As we demonstrated on the success of source-lexicalized parser transfer (Section 5.2), we do not actually need to cover the whole target lexicon; covering even a small part of it can be very helpful.

The work of Tiedemann [2014] gives us confidence that if we are losing something due to using intersection symmetrization, we should not be losing too much. We did not explicitly replicate his work for parsing, but we do investigate the effect of alignment symmetrization on POS annotation projection over parallel data in Section 6.1.2, where we actually find the intersection symmetrization to substantially outperform all of the other options.

Monolingual Greedy Aligner

For situations where the source and target languages are very close to each other, such as the VarDial dataset, we decided to also try using the heuristic Monolingual Greedy Aligner (MGA)[15] of Rosa et al. [2012a][16] instead of GIZA++. Our main motivation was the fact that GIZA++, as well as many other word aligners, completely ignore cross-lingual string-wise word similarity, while the source-lexicalized parsing experiments (Section 5.2), as well as the character-level transformations (Section 5.4), clearly showed that for very close languages, character-level similarities of source and target words definitely can be exploited.

MGA is intended for aligning sentences written in the same language, and had been specifically tuned to be applied to the Czech language and the PDT annotation; it has been used e.g. on error correction or paraphrasing datasets. However, based on its implementation, it can also be used for aligning a pair of sentences in different but lexically similar languages. It utilizes the word, lemma, and POS tag similarity based on Jaro-Winkler distance [Winkler, 1990],[17] and the similarity of relative positions in the sentences, to devise a score for each potential alignment link as a linear combination of these, weighted by pre-set weights. The iterative alignment process then greedily chooses the currently highest scoring pair of words to align in each step; each word can only be aligned once. The process stops when one of the sides is fully aligned, or when the scores of the remaining potential links fall below a pre-set threshold.

Unlike classical word aligners, MGA does not train or iterate on the input data in any way, but processes each pair of sentences independently, using a set of 5 pre-set hyperparameters (weights of the 4 input features, and the stopping threshold); the values of the hyperparameters had been reportedly manually tuned by its author on a set of ten Czech sentence pairs, but were subsequently found to

[15] `https://github.com/ufal/treex/blob/master/lib/Treex/Tool/Align/MonolingualGreedy.pm`

[16] MGA was developed by Martin Popel.

[17] The POS tag similarity makes much more sense on the Czech positional tagset, with which it was intended to be used. We note that for the UPOS tags, it only makes sense in some cases (e.g. *ADJ* and *ADV*) and not in other cases (e.g. *ADJ* and *ADP*); an UPOS-specific similarity should presumably be defined instead.

perform quite well and thus are typically used with their default values.

Even though we use MGA in a different than the intended setting, applying it to sentences in similar but not identical languages, we did not modify either its hyperparameters or its implementation in any way, but still observed it to perform rather competitively.

We note that MGA needs both the source side and the target side of the parallel data to be morphologically analyzed at least for POS, which is a significant limitation. We actually applied it to the VarDial dataset, where availability of supervised target taggers was assumed. However, in the intended low-resource scenario, cross-lingually induced POS tags (Chapter 6) would need to be used; however, we have not evaluated MGA in such a setting.

5.5.3 Word reordering

In general, the word order patterns differ across languages. Often, the usual ordering of Subject, Object, and Verb (predicate) is used as one of the primary characteristics of a language, leading to division of languages into SVO languages (such as English or Czech), SOV languages (such as Japanese or Latin), etc. However, this is by far not a sufficient characterization; not only do many languages exhibit multiple of these word order patterns, but the languages also differ in the ordering of other sentence components than these three – other commonly investigated are nouns and their modifiers, adpositions (prepositions or postpositions), verbs and adverbs, negation particles, etc. The final ordering of words in a given sentence is typically based on many complex mechanisms, and is difficult to cover by rules. The word order may convey information relating to the role of the sentence constituents, the meaning of the sentence as a whole, emphasis, and other. Thus, correct handling of word order in NLP applications such as MT is both important and hard, and a monotone translation (i.e. one that keeps the source word order) is usually not desirable or even acceptable.

In SMT, various approaches can be taken. In Moses, the default approach is the use of a reordering model, which allows the decoder to pick source phrases to translate out of order (the target translation is always constructed continuously left to right), while paying a reordering cost proportional to the number of source words skipped. If the target language model supports the reordering sufficiently to override the reordering cost, the reordering is performed. This approach may be sufficient for simple local reorderings, but is too weak to perform any sophisticated and/or long-distance word order changes.

Another common approach is to perform pre-reordering, in an attempt to bring the source language word order closer to the target language word order, and then proceed with standard monotone translation. This can be done either in a rule-based way as in [Collins et al., 2005], or machine-learned, as in [Xiong et al., 2006]. There is at least one fundamental problem with this approach: as the source sentence word order presumably conveys some information, this may be lost due to the pre-reordering. However, as SMT often struggles even to transfer the core lexical meaning of the source sentence, the often subtle differences in meaning expressed by the word order would be lost in the translation anyway. A logical, though less frequently used alternative, is post-reordering [Sudoh et al., 2011], i.e. performing the translation first and only reordering the output in a

subsequent processing step; however, this approach cannot usefully cooperate with the target language model, which is a strong component in the decoding.

In previous applications of MT to cross-lingual parsing, the authors usually employed standard phrase-based SMT, in which case the complexity of transferring the parse trees through this M:N translation generally overcasts the issues connected to reordering. Moreover, the authors usually employed the translation systems in their vanilla version, without any parameter adaptations. Therefore, we are unaware of any study focusing on the effect of word reordering in the MT decoder when applied in the cross-lingual parser transfer scenario.

In Table 5.4, we investigate the effect of allowing or disallowing word reordering in the MT system, using UD 1.4 target treebanks, with English fixed as the source language for all of the 32 target languages. We either used monotone translation,[18] where word reordering is forbidden, or employed the default reordering model[19] and transferred the annotation via word alignment links output by Moses.[20] The MT system was trained on the OpenSubtitles2016 dataset. The results were originally reported in [Rosa and Žabokrtský, 2017].

To our surprise, we found that except for 8 target languages quite different from the source language (mostly non-Indo-European), allowing the word reordering consistently leads to a *deterioration* of the cross-lingual parsing accuracy; for all the remaining 24 languages (mostly Indo-European), as well as on average, the monotone translation led to better parsing.

When we inspected the data for a few languages, we spotted a common pattern, similar to our observation for translation arity in Section 5.5.1. What we found was that the monotonicity constraint forces the MT system to produce more literal translations, even if somewhat cumbersome, with the syntactic structure rather similar to the source sentence. If reordering is allowed, the translations are more free, more fluent, but the syntactic structure tends to differ more.

The opposing forces of the language model, which pushes for fluency, and the translation model, which pushes for accuracy, are well known in MT. What we argue here is that for cross-lingual transfer of annotation, the ideal balance seems to be different than what is optimal for human-targeted translation – more emphasis should be put on accuracy than on fluency, which, it seems, can be achieved by enforcing monotone translation.

While content words are typically easy to translate 1:1, it is the function words that often do not really match another word in the target language, as one language often uses a function word where the other language does not (consider e.g. prepositions versus inflection for case, determiners versus morphological agreement, or pronoun dropping). The 1:1 MT system then usually translates such words into some "garbage words", e.g. to a preposition or a determiner that is quite frequent in the target language,

In monotone MT, this leads to these "garbage words" appearing at somewhat unexpected positions, which typically makes the translation somewhat cumbersome and disfluent. As this goes against the language model, it makes the translation model more powerful, leading to more literal translations.

If the MT system is allowed to perform reordering, it rather moves the trans-

[18] `-dl 0`

[19] `-reordering msd-bidirectional-fe`

[20] `-alignment-output-file alignment.out`

Target	Monotone	Reordering	Difference
Low source-target similarity			
hi	10.44	**13.17**	**2.73**
eu	10.20	**11.29**	**1.09**
vi	21.48	19.12	-2.36
fa	14.06	**19.22**	**5.16**
ar	14.03	**16.82**	**2.80**
tr	13.79	13.40	-0.40
uk	35.68	**36.51**	**0.83**
hu	21.31	20.85	-0.46
he	28.48	**30.83**	**2.35**
fi	26.77	25.32	-1.44
AVG	19.63	**20.65**	**1.03**
Medium source-target similarity			
sl	33.53	27.48	-6.05
lv	24.09	13.87	-10.22
et	29.44	28.51	-0.94
pl	37.93	35.23	-2.70
ro	32.29	30.78	-1.51
hr	34.44	**35.33**	**0.89**
el	46.45	46.12	-0.33
id	24.31	**26.79**	**2.48**
ru	30.40	29.71	-0.69
cs	32.56	31.33	-1.23
sk	39.38	37.16	-2.22
nl	41.50	39.29	-2.21
gl	18.92	17.90	-1.02
ca	41.19	14.53	-26.66
bg	45.21	42.67	-2.53
AVG	34.11	30.45	-3.66
High source-target similarity			
sv	47.48	37.00	-10.48
de	47.38	44.00	-3.39
da	50.75	19.61	-31.14
fr	51.83	50.75	-1.08
it	51.35	50.62	-0.73
no	58.56	35.14	-23.42
pt	50.98	48.50	-2.48
AVG	51.19	40.80	-10.39
ALL	33.32	29.65	-3.67

Table 5.4: Effect of enabling word reordering in Moses. Measured in LAS on UD 1.4 treebanks, using English as the source for all target languages.

lations of these non-matching words to a "garbage position", e.g. the end of the sentence or a clause boundary, where the flow of the sentence naturally breaks anyway, and the language model is thus more tolerant at such places. However, this allows the language model to have a stronger influence in other parts of the translation, which may make the translation more fluent but less literal.

5.5.4 Simple translation

Apart from using Moses, we also experimented with a simple lexicon-based translation. This approach was introduced by Agić et al. [2012], and later revisited and explored in more detail by Tiedemann et al. [2014], who refer to it as Lookup translation.

In our implementation, we translate each source treebank into the target language word-by-word, independent of source or target context, based on a word-to-word translation table extracted from the parallel data aligned by an intersection alignment. If there are multiple translation options for a source word, the one most frequently seen in the parallel data is produced; we use Jaro-Winkler similarity of the source and target word forms as a tie breaker. Unknown words are left untranslated.

As a potential improvement to this simplistic setup, we use the UPOS and morphological features labels for source-side disambiguation. We automatically analyze the source side of the parallel data, labelling it with UPOS and morphological features, and store these labels together with the source word in the translation table. Then, to translate a source word which is annotated with a specific UPOS and morphological features in the source treebank, we first try to use a matching entry in the translation table. Two backoff layers are used if the translation table does not contain the source word form with the given annotations, first only taking the word form and the UPOS into account, and the second only looking at the word form. Note that, as we only use these annotations on the source side, i.e. for the language assumed to be resource-rich, we can realistically expect to have these annotations available in practice.

5.5.5 How many sources to combine?

In delexicalized parser transfer, we could easily train and combine the delexicalized parsers for all source languages, since the delexicalized parser was trained only once for each source language, independent on the target. However, with cross-lingual parsing lexicalized through MT, this is not the case anymore, as a separate MT system, and subsequently a lexicalized parser, has to be trained for each source-target pair.

While this does not immediately seem to pose a problem, we quickly found that in our setting, using the UD 1.4 dataset with 22 sources and 13 targets, training of 286 Moses systems is way beyond standard technical limits even of a medium-sized computational cluster, which is available to us. Interestingly, at least in our case, the CPU or RAM capacities turned out not to be the main bottleneck – the disc operations were. For some reason, the Moses training pipeline keeps reading from the disc and writing to the disc almost continuously, which, when done in parallel, sooner or later stalls the whole shared filesystem, and

Target	Top 1	Top 3	Top 4	Top 5	Top 6	Top 7
da	55.89	57.58	57.85	57.68	57.72	**57.90**
el	44.78	48.35	49.10	49.41	49.36	**49.63**
hu	28.40	31.46	**33.49**	32.57	32.55	32.84
id	39.06	39.06	40.90	41.11	42.06	**43.08**
ja	11.65	11.93	12.90	**14.58**	13.06	**14.58**
kk	11.45	14.01	**15.51**	13.86	14.01	13.40
lv	28.16	36.59	38.74	42.23	**42.34**	41.35
pl	54.10	56.79	56.93	**58.46**	57.51	57.79
sk	**65.75**	62.78	63.23	62.21	61.91	61.95
ta	**18.21**	**18.21**	17.34	16.39	16.47	15.52
tr	**24.45**	19.45	23.23	22.47	22.44	21.93
uk	47.30	47.72	**48.55**	**48.55**	46.89	47.30
vi	25.44	26.56	27.18	27.90	**27.93**	27.56
AVG	34.97	36.19	37.30	**37.49**	37.25	37.29

Table 5.5: Comparing the combination of top N source languages with N from 1 to 7, as ordered by their KL_{cpos^3} similarity to the target, using the $KL^{-4}_{cpos^3}$ weighted combination of parse trees produced by MT-lexicalized parsers, evaluated using LAS on UD 1.4 treebanks.

sends most of the training processes crashing.[21] Thus, even with the significant computational power available to us, we were unable to train more than roughly two dozen instances of Moses at the same time, and even then we had to adopt an approach of slightly distributing the starting times of the individual training processes, so that there is a reasonable chance that the training will not reach the periods of the most disc-intense operations at the same time in all of the instances. This all holds for the rather small WTC parallel datasets, where the whole pipeline can run in hours or dozens of hours; we can only imagine that for significantly larger datasets, such the OpenSubtitles data, this endeavour would be even more hopeless. We thus conclude that a researcher cannot be reasonably expected to train several dozen MT systems for a single target language.

Eventually,[22] we managed to obtain trained MT systems for the 7 most KL_{cpos^3}-similar source languages for each target language, as well as the oracle language. We present the results of varying the number of source languages to combine in Table 5.5; we do not include "top 2", as this does not make much sense with the voting scheme that we employ. This again uses the UD 1.4 dataset and the best-performing cross-lingual POS tagging, with a weighted lexicalized parse tree combination on top.

It can be seen that combining the top 5 sources works best on average. However, it is obvious that this is mostly compromise, as for some targets, adding more sources keeps improving the score, for other it keeps degrading the score, and yet for other the score keeps improving up to a point and then starts deteriorating after adding several sources; the breaking point is at the top 5 sources for some of the targets, which further licenses the selection of the "top 5" com-

[21] As well as many processes of our colleagues, to their great dismay. If any of them happens to read this, we sincerely apologize for the disruptions to them.

[22] After a few weeks, and with scheduling most of the training for the weekends to minimize the disruption of work of our colleagues.

Setup	Top 1	Top 3	Top 4	Top 5	Top 6	Top 7	Top 22
unweighted delex.	32.23	32.23	31.70	**32.77**	32.71	32.74	32.47
weighted delex.	32.23	33.17	33.77	34.13	**34.14**	33.80	33.28
unweighted lex.	33.91	35.06	34.38	**35.59**	35.15	35.43	
weighted lex.	33.91	35.47	36.37	**36.66**	36.39	36.35	

Table 5.6: Effect of varying the N for a top N combination for various setups. Only the average LAS on the UD 1.4 target treebanks is shown.

bination as the best performing one, even though the average performance is definitely a more important indicator. Possibly due to the $KL^{-4}_{cpos^3}$ weighting, the score improvements tend to get lower with adding more and more languages (if there are improvements), so we seem not to lose that much by stopping at 5.

Ideally, we would hope for $KL^{-4}_{cpos^3}$ weighting to smooth out this effect completely, only weighting the further sources high enough for them to make a meaningful contribution in cases where something useful can be learned from them, and weighting them down into oblivion if they are simply too dissimilar from the target language.[23] Unfortunately, while we have explicitly tuned $KL^{-4}_{cpos^3}$ to do that and we have seen exactly this behaviour on the development HamleDT 2.0 treebanks, this effect is somewhat watered down in the setting evaluated here, even though it still seems to exist.

Alternatively, we would like to choose a different N to use in the top N combination for each target language, but we simply do not see how to determine such N. It would seem logical to combine fewer sources if one or several are extremely similar to the target language, as they alone can probably provide much of the required information. It would also make sense to combine as many sources as possible if none of them is particularly close to the target language, since different things can probably be learnt from them. However, the results do not seem to support such an approach sufficiently for us to formulate it more clearly and explicitly test it in experiments. We thus simply accept the compromise value of 5 as the best we can do.

Interestingly, the "top 5" approach seems to universally work well on our dataset, even in the unweighted case as well as for the delexicalized parsing, as shown in Table 5.6. We only show the averages here, since there is similar variance for the individual targets as we saw in Table 5.5, from which we did not manage to draw any new meaningful conclusions. Note that the underlying POS tags come from an earlier setup this time, and the results are therefore different from those in Table 5.5.

Taking the top 5 sources interestingly practically always leads to the highest score on our dataset, no matter what the setup is. We are quite certain that this is very probably a property of the particular dataset we are using, and that the "magical number" will most probably be slightly different for other settings, as it is clearly different for individual target languages and only seems best on average. However, we believe that our general observation is valid: it is not necessary or even useful to take all of the available sources, as taking only a few

[23]This is obviously not fully possible, at least because of the fact that $KL^{-4}_{cpos^3}$ does not take source treebank sizes into account, i.e. does not even have all of the necessary information on its input.

Target		$\exp\left(5/KL_{cpos^3}\right)$					$KL^{-4}_{cpos^3}$	
lang.	Top 1	Top 3	Top 4	Top 5	Top 6	Top 7	Top 5	Best
da	55.89	55.89	55.89	55.89	55.89	55.89	57.68	57.90
el	44.78	44.78	44.78	44.78	44.78	44.78	49.41	49.63
hu	28.40	31.44	**33.51**	32.55	32.40	32.75	32.57	33.49
id	39.06	39.06	39.06	39.06	39.06	39.06	41.11	43.08
ja	11.65	11.93	12.88	**14.59**	13.06	**14.59**	14.58	14.58
kk	11.45	14.01	**15.51**	13.86	14.01	13.40	13.86	15.51
lv	28.16	36.02	38.21	41.46	**41.57**	40.99	42.23	42.34
pl	54.10	56.82	57.14	57.28	57.17	**57.41**	58.46	58.46
sk	65.75	65.75	65.75	65.75	65.75	65.75	62.21	63.23
ta	18.21	18.21	18.21	18.05	17.58	16.55	16.39	18.21
tr	24.45	24.45	24.45	23.61	23.43	23.28	22.47	23.23
uk	47.30	47.30	48.96	49.38	49.38	**50.62**	48.55	48.55
vi	25.44	26.54	27.18	**27.90**	27.74	27.49	27.90	27.93
AVG	34.97	36.32	37.04	**37.24**	37.06	37.12	37.49	38.16

Table 5.7: Performance of the softmax-normalization of KL_{cpos^3}, compared to the single-best transfer, and to our $KL^{-4}_{cpos^3}$ transformation on top 5 sources as well as the best-performing combination from Table 5.5, on UD 1.4.

of the most similar ones works best. Besides, training so many MT systems may be too computationally demanding to be carried out in practice, especially in a setting where significant computational power is not available, so we are quite relieved with this finding. We thus limit ourselves to only using the top 5 sources in further experiments on the UD 1.4 dataset.

Comparison to softmax distance normalization

As we mentioned in Section 4.2.5, Agić [2017] suggested a slightly different way of transforming the KL_{cpos^3} distance into a similarity measure, based on softmax normalization with temperature:

$$sim_{Agic}(tgt, src) = \exp\left(\frac{5}{KL_{cpos^3}(tgt, src)}\right) \tag{5.1}$$

As he did not explain why this should be better than our original suggestion of using $KL^{-4}_{cpos^3}$ as the similarity measure, we decided to compare the two approaches empirically.

Table 5.7 shows that the softmax normalization is often very peaked, giving a too low weight to other sources languages than the most similar one. In 4 cases, this effectively eliminates all other sources (*da, el, id, sk*), making the application of the multi-source combination useless; for Slovak, this is actually beneficial, since we have not been able to improve over the single-best transfer anyway, thus avoiding the losses that we encountered. We observe a similar situation for Tamil and Turkish, where we were also unable to surpass the single-best transfer, and Agić's more conservative employment of additional sources thus leads to lower losses against the single-source transfer.

Ukrainian is the only target language where the softmax weighting outperforms both our weighting and the single-source transfer. For all the remaining

System	cs→sk	sl→hr	da+sv→no
Source-lexicalized	53.72	53.35	59.95
Coltekin	64.05	55.20	65.62
Tiedemann	73.14	57.98	68.60
MGA+simple	**78.12**	**60.70**	**70.21**
Supervised	69.14	<u>68.51</u>	<u>78.23</u>

Table 5.8: Results of the VarDial 2017 cross-lingual parsing shared task (LAS). Comparing our MGA+simple submission to the source-lexicalized baseline and monolingual supervised upper-bound, as well as to the submission of Tiedemann [2017] and Çöltekin and Rama [2017].

6 target languages, $KL^{-4}_{cpos^3}$ leads to identical or higher improvements over the single-source than the softmaxed KL_{cpos^3}, both when comparing the "top 5" combination for both and when comparing the best performing combination for both; it also performs better on average. Thus, except for Ukrainian, the softmaxed KL_{cpos^3} was only able to improve over $KL^{-4}_{cpos^3}$ by minimizing the loss towards the single-source, not by actually leading to a better combination of the sources.

While it is probable that a different choice of the temperature hyperparameter value could lead to better performance of the softmax transformation, we note that we simply use the value that Agić [2017] obtained by tuning on English UD 1.3 development data with cross-lingually predicted POS tags, i.e. in a setting very similar to that used in our evaluation, while $KL^{-4}_{cpos^3}$ was tuned on the considerably different HamleDT 2.0 treebanks with gold POS tags (both of the methods were tuned on delexicalized parser transfer). Therefore, we find it fair to conclude that our original transformation of the KL_{cpos^3} distance into the $KL^{-4}_{cpos^3}$ similarity measure outperforms the softmax transformation suggested by Agić [2017].

5.6 Evaluation

In this section, we evaluate the lexicalized cross-lingual parsing in several setups and on several datasets.

5.6.1 VarDial shared task

In January 2017, we participated with an early version of our cross-lingual parsing system in the VarDial 2017 Cross-lingual parsing shared task [Zampieri et al., 2017]. Although we have further improved the system since then, we still find it fruitful to review the results of that shared task, as this was a unique occasion in terms of reliably comparing our approach to other state-of-the-art systems on an identical test set with identical training data; and, incidentally, also using an identical UDPipe parser.

In particular, as our submission to the shared task outperformed both of the other submissions [Tiedemann, 2017, Çöltekin and Rama, 2017] for all target languages, we believe that this shows our methods to be clearly within the state of the art in the field.

In Table 5.8, we represent the results of our participation in the shared task, as well as the results achieved by other participants. We also list the source-lexicalized baseline and the supervised upper-bound. The results are taken over from the task organizers.[24] The task was evaluated using the VarDial subsection of the UD 1.4 treebanks (see Section 1.3.2) with supervised POS tags.

The "MGA+simple" line denotes our winning submission to the shared task [Rosa et al., 2017],[25] based on MGA alignment with simple translation using OpenSubtitles2016 parallel data, with several further improvements. These included, most importantly, target word embeddings pre-training (Section 5.3), slight further source annotation harmonizations (Section 1.2.5),[26] and the incorporation of the *Case* morphological feature into the parser input, as it proved to be cross-lingually useful for these particular language pairs (Section 3.2.1).[27] However, as most of the improvements had been specifically tuned for the particular languages, we removed them in later experiments, since the hand-tuning does not scale well to datasets containing many languages; further on, we only employ the word embeddings pre-training, which is language-independent and seems to be rather universally useful.

We can see that even with the simple machine translation strategy, the system performs well above the baseline, and even surpasses the supervised upper-bound for Slovak. While this setting is not the typical intended one, as none of these languages are truly under-resourced, and also as supervised target tags were used, we still find the results rather solid. Nevertheless, it must be noted that all of these language pairs are extremely close, as can be seen already from the high performance of the source-lexicalized baseline (Section 5.2). Moreover, the Slovak supervised parser seems to have an unexpectedly low accuracy, i.e. we believe that with proper tuning, the monolingual supervised parser might be able to surpass our cross-lingual one.

Additionally, we include the results of the second best system in the competition, submitted by Tiedemann [2017], who used a slightly different approach for each of the language pairs, alternating between the tree projection approach of Agić et al. [2016], which we explored in Section 4.3.3, and phrase-based or syntax-based MT, based on his previous work [Tiedemann, 2014, Tiedemann et al., 2014, Tiedemann, 2015]. It is interesting to note that even though the cross-lingual lexicalization component was the one that the author paid most attention to, investigating several powerful and well-developed approaches and choosing the best-performing one for each target (which brought him improvements of several LAS points), our submission managed to surpass that by a large margin for each of the targets, despite using an extremely simple MT approach. On one hand, it very much seems that our winning of the shared task does not come from using particularly that MT approach, as we have since been able to further improve the performance of our methods by using a stronger SMT system; we

[24] `https://bitbucket.org/hy-crossNLP/vardial2017`

[25] This was joint work by all the authors of the paper.

[26] These additional harmonizations were devised by Daniel Zeman.

[27] Other improvements consisted of varying the way in which the morphological taggers were trained and applied, as supervised target taggers were allowed in the shared task, which opened up a range of possible setups to use; however, as assuming the availability of a supervised tagger is rather unrealistic for a real low-resource target language, we do not find this component of our submission to be of much value.

rather believe that the "further improvements", which we mentioned above, were the key component which differentiated our setup from the others. On the other hand, using such a simple MT approach surprisingly did not substantially hurt our performance either, contrary to our expectations. Thus, the results seem to actually confirm the findings of Tiedemann et al. [2014], who observed that even simple word-based approaches to MT can lead to highly competitive accuracies in cross-lingual parsing.

Furthermore, we believe this to support our decision to prefer the simplicity and accuracy of the annotation transfer to the intrinsic quality of the MT system. While we do not feel that we have sufficient proof to denote the path we chose as the one and only correct way of lexicalizing the cross-lingual parser transfer, we do believe that the results show it to constitute a viable and highly competitive approach at the very least.

The third submission to the cross-lingual parsing shared task was made by Çöltekin and Rama [2017], who participated in multiple of the VarDial shared tasks and thus did not focus that much on the cross-lingual parsing task itself. Interestingly, they use an approach very similar to ours in principle, but without the "further improvements". They use a word-based Moses, but with the `efmaral` aligner [Östling and Tiedemann, 2016] and grow-diag-final-and symmetrization, and do not seem to employ any further tricks or tuning. This further supports our belief that the good performance of our setup was mostly due to the "additional improvements" that we employed, as the submission of Tiedemann [2017] seems to mostly win over that of Çöltekin and Rama [2017] because of careful selection of the machine translation approach to use for each language pair (together with a focus on the approach to tagging).

5.6.2 Extended VarDial language set

We now move to the Extended VarDial dataset with more language pairs, still using the large OpenSubtitles2016 parallel data and supervised target POS tags.

In Table 5.9 and Table 5.10, we evaluate the effect of MT setup, using both intrinsic MT evaluation (BLEU) and extrinsic evaluation in delexicalized parser transfer (LAS). We compare the MGA with simple translation to GIZA++ with monotone word-based Moses, and source-lexicalization as a baseline approach. Note that although source-lexicalization can also be evaluated using BLEU – we simply take the source side of the parallel data and evaluate it against the target side reference translation – the BLEU scores are extremely unreliable under such conditions [Bojar et al., 2010] and we provide them rather for curiosity.

On average, higher absolute BLEU scores are achieved for the more similar languages, but there is large variation in the results; not only language similarity, but also e.g. parallel data sizes are important. Remarkably high scores are achieved for source-lexicalization on VarDial languages, as the shared task organizers focused on pairs of extremely close languages; still, the scores are negligible compared to these achieved by real MT setups.

What we are most interested in is the comparison of the simple MGA+simple translation setup versus the full-fledged GIZA++ and Moses. The main advantage of Moses in that setting is that it uses a language model to explore and evaluate multiple translation options for each word, while the simple translation

Languages	Source-lexicalized	MGA, simple	GIZA++, Moses	Δ abs	Δ rel
Source language very similar to target					
cs→sk	10.1	37.0	37.4	0.3	1%
da+sv→no	7.7	38.3	43.0	4.6	12%
sl→hr	5.3	19.3	20.6	1.3	7%
fr→es	1.7	13.8	18.0	4.2	31%
es→fr	1.7	13.2	17.6	4.4	33%
cs→pl	1.4	9.3	11.0	1.6	18%
Average	4.6	21.8	24.6	2.7	13%
Source language less similar to target					
it→ro	2.1	8.3	12.1	3.8	45%
en→sv	1.3	9.2	13.1	3.9	43%
en→de	2.9	11.0	16.3	5.2	47%
de→sv	1.4	10.4	12.8	2.4	23%
de→en	3.0	15.5	18.9	3.4	22%
fr→en	2.2	16.5	21.4	5.0	30%
Average	2.2	11.8	15.8	3.9	33%

Table 5.9: Evaluation of various setups for machine translation using BLEU computed on the last 10,000 sentences from the parallel data. Comparing the source-lexicalized baseline, MGA + simple translation, and GIZA++ with Moses translation, also reporting the absolute and relative improvement in BLEU score achieved by Moses over MGA

approach only uses the most frequently aligned target word as the translation each time in a context-independent fashion; both of the systems use monotone word-based translation in this experiment, i.e. translating the source words 1:1 into target words without reordering.

While the improvement in BLEU brought by using Moses is substantial for nearly all language pairs, the relative improvement is only 13% on average for the more similar languages, while being 33% for the less similar (although there is again much variation). This shows that for very close languages, even simpler approaches to MT are capable of achieving somewhat competitive translation quality. In particular, the result for Czech→Slovak, which is the most similar language pair in the set, is fascinating, with the difference in BLEU scores being rather negligible; the high similarity of these languages is also manifested in the very high BLEU score of 10.1 for the source-lexicalization baseline.

Let us now move to the extrinsic evaluation (Table 5.10). Suddenly, the situation is somewhat different. It is still true that the source-lexicalized approach is substantially weaker than the real MT approaches (although it does achieve competitive accuracy in several cases). However, the GIZA++ and Moses setup is not such a clear winner in the extrinsic evaluation, actually leading to a worse accuracy than MGA and simple translation for 3 of the 12 language pairs, bringing a rather moderate improvement of 1.4 LAS points on average. The improvements are usually larger for the more distant language pairs, but not always.

Moreover, the improvement in BLEU does not seem to predict well the improvement in LAS. The Pearson coefficient of the correlation between the absolute LAS difference and the relative BLEU difference of the two MT setups is 0.21; if

Languages	Source-lexicalized	MGA, simple	Δ vs SrcLex	GIZA++, Moses	Δ vs MGA
Source language very similar to target					
cs→sk	62.25	68.86	6.61	70.50	1.64
da+sv→no	62.31	67.76	5.45	68.16	0.40
sl→hr	60.02	63.76	3.74	61.93	-1.83
fr→es	61.41	67.52	6.11	69.61	2.09
es→fr	69.69	72.82	3.13	75.21	2.39
cs→pl	62.20	66.52	4.32	69.29	2.77
Average	62.98	67.87	4.89	69.12	1.24
Source language less similar to target					
it→ro	21.94	33.60	11.66	37.06	3.46
en→sv	47.61	52.52	4.91	51.84	-0.68
en→de	37.44	41.46	4.02	43.46	2.00
de→sv	34.94	41.96	7.02	44.77	2.81
de→en	46.19	46.93	0.74	49.99	3.06
fr→en	39.92	53.47	13.55	52.08	-1.39
Average	38.01	44.99	6.98	46.53	1.54

Table 5.10: Evaluation of various setups for machine translation using LAS. Comparing the source-lexicalized baseline, MGA + simple translation, and GIZA++ with Moses translation, also reporting the absolute improvements in LAS.

we use the absolute BLEU difference, it is -0.03. Thus, the results merely suggest that using a higher quality MT system, as measured in BLEU, often leads to a better lexicalization of the cross-lingual parsing, but the connection between the performances is not very strong. There is a clear improvement by introducing a full MT setup instead of the source-lexicalization, but the influence of the particular setup of this MT system seems to be much weaker.

Nevertheless, the setup using Moses does clearly perform better on average, and we thus select it as our final MT system for cross-lingual parser lexicalization.

The results presented here give additional support to our decision to focus only on word-based translation. While we have not explicitly measured the effect of moving from word-based to phrase-based MT, it seems rather clear that, especially for very close languages, no large improvements are to be expected, and even deteriorations may be common.

For more distant languages, the situation may be somewhat different, and PBMT could be expected to bring larger improvements there. Nevertheless, as the absolute LAS scores clearly show, cross-lingual parser transfer between only moderately similar languages yields much lower accuracies, with more than half of the relations being predicted incorrectly by the parser. Moreover, the setting for this evaluation is still a very optimistic one, assuming the availability of supervised target POS tags, rather large parallel corpora for training the MT systems, and still using rather close language pairs. The performance of the cross-lingual lexicalization methods applied to similar languages is thus probably of more interest to us, as for distant languages, the accuracies will most probably be too low for any practical use of the results.

Target lang.	Single-source transfer: Oracle delex	Oracle lex	Automatic delex	Automatic lex	Multi-source tree combination: Unweighted delex	Unweighted lex	Weighted delex	Weighted lex
da	50.44	55.89	50.44	55.89	51.28	**58.31**	50.75	57.68
el	48.34	**52.31**	43.52	44.78	46.05	**50.72**	45.34	49.41
hu	28.93	30.33	25.96	28.40	30.64	**33.11**	30.62	32.57
id	37.48	40.01	35.32	39.06	37.97	**41.67**	37.32	41.11
ja	**19.89**	18.71	9.33	11.65	15.43	**16.04**	14.21	14.58
kk	11.45	**15.96**	11.45	11.45	12.50	**14.91**	12.65	13.86
lv	35.99	40.74	24.31	28.16	37.64	41.68	39.01	**42.23**
pl	55.93	**58.84**	51.79	54.10	54.03	57.59	54.84	**58.46**
sk	59.41	**65.75**	59.41	**65.75**	49.49	53.81	57.44	62.21
ta	17.50	**18.21**	17.50	**18.21**	10.37	10.45	15.44	16.39
tr	**25.23**	24.45	**25.23**	24.45	21.64	22.07	22.86	22.47
uk	44.40	47.30	44.40	47.30	44.40	46.89	44.81	**48.55**
vi	22.47	25.44	21.83	25.44	25.54	**28.10**	25.32	27.90
AVG	35.19	**38.00**	32.34	34.97	33.61	36.57	34.66	**37.49**
St.Dev.	15.61	17.02	16.25	17.52	15.33	16.92	15.66	17.39

Table 5.11: Evaluation of cross-lingual lexicalization on UD 1.4 treebanks subset with LAS, using WTC data, GIZA++ alignment, word based monotone Moses, cross-lingually induced POS tags, and sources selected using $KL^{-4}_{cpos^3}$. The top 5 sources are used in the tree combination. Best score and best non-oracle score are marked in bold, and reaching the oracle score in the single-source transfer is marked by underlining.

5.6.3 UD 1.4 language set

In our final evaluation, we move to a more realistic setting, using the larger and more varied UD 1.4 dataset, employing the best cross-lingual tagging setup from Chapter 6 to obtain target POS tags, training the word-based monotone Moses MT system on the smaller out-of-domain WTC multiparallel data, and using $KL^{-4}_{cpos^3}$ to select and weight the top 5 source languages to use for processing each target language (or top 1 in the single-source transfer).

Table 5.11 shows the parsing accuracies in LAS for both delexicalized and lexicalized cross-lingual parser transfer in single-source and multi-source transfer, using either unweighted or weighted parse tree combination. For the single-source transfer, the highest obtainable results (i.e. the oracle) are also presented; however, as we did not train the Moses system for all existing language pairs, the lexicalized oracle may not always be the true oracle – we used the source language of the delexicalized oracle, unless the result for one of the "top 5" languages was higher, in which case we selected this one as the oracle source.

Lexicalization

The first thing to notice from the results is that lexicalized parsing outperforms the delexicalized one for all methods and all target languages except for Turkish (and, in the case of the oracle, for Japanese), with the average improvement on all of the setups being around 3 LAS points. While this may not seem to be a lot, we would like to note that the difference between monolingual supervised lexicalized

and delexicalized parsers for these languages is 8.8 LAS points on average, with the average lexicalized LAS score being 70.1% (Table 3.1). With the cross-lingual parsing LAS being only about half of the supervised, we probably cannot hope to achieve a much larger improvement than 4.5 points on average. Thus, while there still seems to be room for improvement, we seem to already cover most of that gap with our approach.

Interestingly, we can see that even though the word-based monotone MT could be expected to be particularly unsuitable for distant language pairs, we cannot make such conclusion from the results. While the absolute improvements in LAS brought by the lexicalization are clearly larger for Indo-European languages than for the non-Indo-European ones (which are all quite solitary in our dataset), the absolute LAS scores themselves are also higher. In relative terms, the lexicalization of the tree combination setup, on average, improves the LAS scores by 8.6% for the Indo-European target languages, and by 6.2% for the non-Indo-European ones, which does not seem to be a crucial difference. Moreover, the *median* relative improvements are even closer to each other: 8.3% for Indo-European and 7.8% for non-Indo-European targets. We thus conclude that the monotone word-based MT seems to be reasonably suitable even for very distant languages, which we find rather surprising.

Source selection

KL_{cpos^3} is able to identify the oracle for half of the target languages; note that the oracle source language may not be the same for the lexicalized and for the delexicalized parsing, as it happened for Kazakh and Vietnamese.[28] However, it is nearly always surpassed by the top 5 multi-source tree combination, typically both by the weighted one as well as the unweighted one. On the other hand, in the three cases where the single-source method performs best (*sk, ta, tr* – each time the oracle source is successfully identified), the tree combination methods suffer quite unpleasant losses, especially the unweighted one.

Moreover, while the oracle single-source transfer still achieves the highest accuracy on average, it is no more such a strong "upper-bound", being outperformed by the weighted tree combination for half of the target languages and only gaining 0.5 LAS points over it on average. For most target languages, the accuracies achieved by these two methods are within 2 or 3 LAS points from each other.

Source weighting

As for the performance of $KL^{-4}_{cpos^3}$ weighting itself, we can again see a pattern similar to what we observed for the delexicalized transfer on the HamleDT 2.0 dataset in Table 4.8 – while the weighted tree combination is better on average, even though by only 0.9 LAS point, it is actually surpassed by the unweighted combination for half of the target languages. Furthermore, in cases where $KL^{-4}_{cpos^3}$ works well for the multi-source weighting, it is also typically able to identify the oracle treebank, and thus the single-source method is highly competitive or even better than the weighted tree combination. Thus, although being best on average,

[28] This is mostly because the accuracies are so low that the notion of an "oracle" source language is quite weak here – basically, many of the available languages are more or less equally bad sources for these targets.

it actually only achieves the best score in 3 cases. From this perspective, the unweighted combination actually may seem as a better option.

This could also be interpreted as the KL_{cpos^3} measure itself still performing well in the given setting, appropriately selecting one or multiple source treebanks to use for the cross-lingual parsing (note that, in this way, even the unweighted tree combination makes use of it), but its $KL^{-4}_{cpos^3}$ variant not performing as nicely as we saw it in Chapter 4. This may well be the case – the KL_{cpos^3} measure itself was theoretically motivated in a sensible way, and its only hyperparameter that was determined empirically based on the performance on development treebanks was the length of the POS n-grams to use, where trigrams seemed to clearly perform better than both bigrams and tetragrams. On the other hand, we did not find any good theoretical background for determining the best way of converting it from a mere ranking measure to a weight useful for interpolating multiple sources, and thus based the exact formula of $KL^{-4}_{cpos^3}$ practically exclusively on its performance on the development data, still noting that the exponentiation to the minus fourth power is more of a compromise choice rather than a clearly best approach (see Section 4.2.5). Therefore, it is probably no surprise that it seems to perform best on average but not for many of the individual target languages, as it was actually chosen in the development particularly based on its good and stable average performance.

Nevertheless, even though the weighted tree combination is usually not the best performing approach, it typically gets very close to it, losing on average only 1 LAS point towards the winner, and always less than 3.6 points. In this way, the $KL^{-4}_{cpos^3}$ weighting does perform its job well, evening out the results and thus preventing any huge losses (but also the wins). With the unweighted variant of the tree combination method, as well as with the single-source method, it is much more of a hit-or-miss, losing up to 12 LAS points towards the winner for the former, and up to 14 points for the latter. Thus, even though we acknowledge that it does not usually achieve the best result, we still conclude that the weighted lexicalized tree combination seems to be the best of the methods which we evaluated.

Predicting the LAS

The absolute LASes are rather low on average, but there is much variance. By far, the strongest influencer of the achieved accuracy for a target language is the presence of sufficiently similar source languages in the dataset (see also Attachment C). In particular, for Indo-European target languages, the average LAS of the weighted tree combination is 53%, while for non-Indo-European ones, it is only 24%; this is quite striking. The languages are even separable into the two groups by the LAS being above or below 42% (or 40.5% for the oracle single-source transfer). Thus, independent of what particular setup one uses, the key to success in cross-lingual parsing is to find annotated resources for one or more source languages sufficiently similar to the target language.

KL_{cpos^3} can be used as a rather good predictor of the final LAS. The Pearson coefficient of the correlation between the lexicalized weighted tree combination LAS and the lowest KL_{cpos^3} distance for the target language (i.e. the distance of the target language and the source language we select for the single-source

Target	l.w.t.c.	KL_{cpos^3}	$1/4KL_{cpos^3}$	Δ
da	57.68	0.39	64.46	6.78
el	49.41	0.55	45.70	-3.71
hu	32.57	0.73	34.39	1.82
id	41.11	0.61	40.75	-0.36
ja	14.58	1.14	21.94	7.36
kk	13.86	0.94	26.56	12.70
lv	42.23	0.71	35.22	-7.00
pl	58.46	0.55	45.62	-12.84
sk	62.21	0.50	49.51	-12.70
ta	16.39	0.80	31.42	15.03
tr	22.47	0.70	35.89	13.42
uk	48.55	0.59	42.09	-6.46
vi	27.90	0.92	27.18	-0.73
AVG	37.49	0.70	38.52	1.02

Table 5.12: Estimation of LAS of the lexicalized weighted tree combination as $1/4k$, where k is the lowest source-target KL_{cpos^3} on the dataset for the given target. The LAS are given in %.

transfer) is -0.87; for the lexicalized single-source transfer, it is -0.89, and -0.85 for the oracle.

As shown in Table 5.12, the formula of $1/4KL_{cpos^3}$ can be used to very roughly guess the LAS that the lexicalized weighted multi-source tree combination will achieve. While the estimation often misses the real value by more than 10 LAS points, the average absolute error is only 7.8, which is rather respectable, given that this estimation ignores other important factors, such as the similarities of other source languages, or the size of the training datasets.

On the other hand, on this dataset, a similar result can be achieved simply by estimating the LAS to be 53% for Indo-European languages and 24% for non-Indo-European languages (i.e. the average values for these groups), with a correlation of 0.86 and an average absolute error of 7.4.

5.6.4 Comparison to unsupervised parsing

Finally, we also compare our best performing setup to unsupervised dependency parsing of Mareček [2016b]. This is a state-of-the-art approach, based on the Dependency Model with Valence of Klein and Manning [2004] and employing a range of improvements implemented as external prior probabilities, most importantly exploiting the reducibility principle [Mareček and Žabokrtský, 2012] and performing a minimal supervision by adding a set of handcrafted language-independent grammatical rules based on the UD guidelines.

As our setup, we use the lexicalized weighted parse tree combination based on the 5 closest source languages. Since the unsupervised parser is delexicalized, we also compare its performance to the delexicalized variant of our setup.

Both of the parsers are evaluated on exactly the same dataset, i.e. the UD 1.4 development treebanks tagged with cross-lingually induced POS tags (Chapter 6). However, as the unsupervised parser only produces unlabelled parse trees, we evaluate the results using UAS instead of LAS; i.e. our setups correspond to the

Target lang.	Cross-lingual lex	Cross-lingual delex	Unsupervised delex	Unsupervised Δ delex
da	**68.16**	62.15	48.09	-14.06
el	**64.75**	60.80	59.36	-1.44
hu	**51.14**	50.64	41.08	-9.56
id	**57.28**	53.91	42.21	-11.70
ja	32.15	31.60	**33.81**	2.21
kk	26.36	24.70	**32.38**	7.68
lv	**56.43**	52.36	32.31	-20.05
pl	**73.11**	71.05	61.89	-9.17
sk	**76.24**	71.11	64.81	-6.30
ta	**31.91**	**31.91**	25.50	-6.42
tr	**33.11**	31.76	28.82	-2.94
uk	67.63	**70.12**	55.19	-14.93
vi	**47.30**	43.51	41.03	-2.48
AVG	**52.74**	50.43	43.57	-6.86

Table 5.13: Comparison of our best performing cross-lingual setup to the unsupervised parser of Mareček [2016b], applied to cross-lingually induced POS tags and evaluated with UAS. The difference of scores between the unsupervised parser and our delexicalized parser is also reported.

last two columns of Table 5.11 except for the evaluation measure used. The results for the unsupervised parser were obtained directly from David Mareček through personal communication – this experiment was performed with his help just for the sake of this comparison.

The results in Table 5.13 show that cross-lingual parsing clearly outperforms unsupervised parsing, especially in the case of target languages for which sufficiently similar source languages are available in our dataset (e.g. *da, lv, pl, uk*). On average, the delexicalized cross-lingual parsing performs better by nearly 7 UAS points, and more than 9 UAS points in the lexicalized setting.

On the other hand, in the case of more solitary languages, the unsupervised parser often achieves quite competitive scores (*el, tr, vi*), and even surpasses our setup by respectable margins in two cases (*ja, kk*). This seems to show that for very solitary languages, the cross-lingual transfer does not have sufficient ground to handle the target language well, as it mostly builds upon knowledge learned from distant source languages. The unsupervised parser takes a different approach, actively exploring the target language, identifying repeated patterns, and trying to devise structures that fit the language well; the ultimate optimization criterion for the unsupervised parser is to devise parse trees in such a way that the same substructures are frequently repeated in the corpus. The results seem to show that this is a promising path, especially for solitary languages, and that cross-lingual parsing does indeed have something to learn from unsupervised parsing in this respect. We believe that this opens interesting opportunities for future research.

5.7 Summary

In this chapter, we examined the options of introducing lexical features into the parser, because, as we demonstrated in Section 3.1, POS tags alone are generally not sufficient to achieve high-accuracy parsing.

Similarly to Tiedemann et al. [2014], we found translation of the source treebank by a machine translation system to perform well in that task. However, while Tiedemann [2014] decided to further focus on phrase-based M:N translation and complex dependency tree transfer mechanisms, we followed the other promising path of using lower-quality simpler word-based 1:1 MT, allowing us to employ a direct 1:1 annotation transfer mechanism.

We found that the quality of the MT system seems to have only a moderate influence on the accuracy of cross-lingual parsing, with even very simple approaches already capable of achieving substantial improvements in parsing accuracy, which then rises only slowly when more sophisticated MT is employed. We thus consider it sensible not to venture into the complex M:N translation, which would slightly improve the translation quality but would make the annotation transfer much more complex and noisy.

Surprisingly, we even observed a positive effect of using word-based monotone translation. This approach seems to force the MT system to resort to very literal translations, in which the morphological and syntactic structure of the source sentence tends to be better preserved than in higher-quality but less structurally similar translations, making the annotation transfer even more reliable.

We then re-evaluated the source selection and source combination methods from Chapter 4, which were originally developed for and applied to delexicalized parsing, finding them to perform rather well even in the cross-lingually lexicalized setting. Our conclusion is thus mostly similar to the previous chapter, finding the parse tree combination method weighted by $KL^{-4}_{cpos^3}$ source-target language similarity to perform best. However, this time, we select the combination of only the 5 most similar sources as the ultimate setup, which we observed to achieve respectable accuracies while being more practical than a combination of all sources.

6. Cross-lingual Tagging

POS tagging is a very important prerequisite for syntactic parsing, both monolingual and cross-lingual. In NLP, POS tagging is a standard task on its own, with many applications, and is useful even without subsequent parsing. However, as our focus in this thesis is on parsing, we rather treat POS tagging only as a necessary pre-processing step in the parsing process. Nevertheless, as it is a crucial step, we focus on it in more detail in this chapter.

The need of POS tags is clear in the case of delexicalized parsing, since they are practically the only input feature for the parsers. While we originally used supervised target POS tags for cross-lingual delexicalized parser transfer, it is clear that this is too optimistic given the intended use case, and an exploration of cross-lingual POS tagging methods is therefore inevitable.

For lexicalized parsing, the need of POS tags is less obvious, as the parser could, in theory, internally perform all the processing that the tagger does, without the need to explicitly use the POS tag labels in the process. While this is true, and there is indeed successful research on joint tagging and parsing [Bohnet and Nivre, 2012] or parsing without tagging [Fairon et al., 2005], this has never become the mainstream approach to parsing, presumably because it results in much more complex systems without clear performance gains. Research has shown that, while a joint approach can always be expected to achieve better accuracies, in the case of tagging and parsing, the decoupling of these two steps does not reduce the accuracies too much, while being more practical. One may e.g. experiment with varying the tagging or parsing approach independently, without having to modify and retrain the whole system. As both tagging and parsing are rather hard problems on their own, asking for clever and complex solutions, the researchers benefit greatly from the possibility to only focus on one or the other, making both development and evaluation much simpler and clearer. Best results are then obtained by combining independently developed best taggers and best parsers, using the POS tags as a useful and practical common interface.

The benefit of this decoupling is even more clear in our case, since the POS tags themselves are a very good language-independent abstraction over the individual word forms, as shown by the delexicalized parsing approaches, allowing us to focus on several tasks independently – the delexicalized parser transfer, the cross-lingual lexicalization of the parsers, and the cross-lingual POS tagging. This decoupling is also justified by our evaluations, as improving the approach used for one of them practically always improves the overall performance of the whole setup, regardless of the particular approaches used for the other problems.

Nevertheless, as POS tagging is not our main focus, we do not present any particularly advanced or novel methods in this chapter. Instead, we build on pre-existing approaches of other authors, which we have also already seen applied to cross-lingual parsing in this thesis, and introduce some rather minor modifications and improvements into them. Specifically, we use POS projection over multiparallel data in Section 6.1, which is a simpler version of the tree projection method from Section 4.3.3, and machine translation of source treebanks in Section 6.2, which is mostly identical to the parser lexicalization via MT from Section 5.5.

6.1 Projection over (multi)parallel data

To the best of our knowledge, the first work on cross-lingual tagger induction is that of Yarowsky et al. [2001], who introduce the approach of projecting POS tags over parallel corpora. The main principle of their method of devising a tagger for an under-resourced target language, which we follow even in our work, is as follows:

1. train a POS tagger for a source language,
2. use it to tag the source side of a parallel corpus,
3. word-align the parallel corpus,
4. transfer the source POS tags over the word alignment links onto the target words,
5. train a target tagger on the, now tagged, target side of the parallel corpus.

The setting for the method fits perfectly our use case, as it utilizes just the resources that we assume to have, i.e. a source language treebank and a source-target sentence-aligned parallel corpus.

The method has been revisited and further improved by many authors, some of them devising remarkably sophisticated solutions. For example, Das and Petrov [2011] introduced a graph-based projection approach, constructing bilingual graphs with nodes corresponding to word types and word trigram types to define constraints for an unsupervised tagging model. In a somewhat related approach, Täckström et al. [2013a] leveraged both bilingual texts and the Wiktionary lexicon to induce both token-level and type-level constraints, which were then used to provide a partial signal in training a partially observed conditional random field model. A wide range of other works exist in this field.

In our work, we mostly follow the approach of Agić et al. [2015, 2016], who observed the utility of exploiting multiparallel corpora, such as Bible (through EBC) or WTC texts. The crucial advantage of such resources is the fact that for each target sentence, there are aligned source sentences in multiple languages. These constitute a much more robust source of information, as, for each target word, there are multiple POS tag options based on the various aligned tagged source words. The authors find a simple majority voting mechanism to perform remarkably well, and gain further improvements by incorporating the alignment scores to obtain a weighted voting setup. This is practically the same method as we described in Section 4.3.3 for parsing, but it is simpler, as there are no complex constraints on the validity of the tag sequence as there are on the parse trees ("treeness"); i.e., the MST algorithm is not used here. The authors also show that respectable tagging accuracies can be obtained even with rather small parallel corpora based on religious texts, which, however, can be realistically expected to be available for many under-resourced target languages. This makes their work even more relevant for us.

6.1.1 Our implementation

Our implementation of the cross-lingual POS projection on multiparallel data is not exactly same as that of Agić et al. [2016], we therefore explicitly detail our approach here. We assume that there is one target language and a set of source languages (we handle each target language independently).

1. We use the WTC, sentence-aligned with Hunalign[1] [Varga et al., 2007] and word-aligned with FastAlign[2] [Dyer et al., 2013], using the intersection symmetrization.

2. We train the UDPipe[3] [Straka et al., 2016] tagger on the UD 1.4 source treebanks, setting it to only predict UPOS.[4]

3. Using the trained taggers, we label the source sides of the multiparallel data with POS tags.

4. For each target word, we gather the POS tags of source words aligned to it, and then use majority voting to choose a POS tag for the target word.

5. If the target word is not aligned to anything, we use the POS tag most frequently assigned to the target word in other of its occurrences in the data; we thus need to perform the POS projection in two passes.

6. If the tag for a word still cannot be determined, we label it as "NOUN", as this seems to be the most frequent POS for open class words ("PROPN" might also be a good label).

7. We train the target UDPipe tagger on the, now tagged, target side of the multiparallel corpus, using the same settings as above.

We have also investigated several variations to the approach, but eventually settled on the method as described above. Nevertheless, we describe those experiments in the following sections. There are also other potential improvements which we did not evaluate, such as taking the target context into account, or tagging all target languages simultaneously as done by Agić et al. [2015].

6.1.2 Effect of alignment symmetrization

As we noted in Section 5.5.2, there is a range of possible choices for word alignment symmetrization. Agić et al. [2016] reported that reverse alignment works best for annotation projection, although this is never clearly explained – the paper claims that Agić et al. [2015] "observe a major advantage in using reverse-mode alignment for POS projection (4-5 accuracy points absolute)". However, no such observation is mentioned in the cited paper; instead, the paper explicitly mentions that "many-to-many word alignments are allowed", and does not even contain the word "reverse". Therefore, we decided to investigate this problem ourselves

[1] `https://github.com/danielvarga/hunalign`

[2] `https://github.com/clab/fast_align`

[3] `http://ufal.mff.cuni.cz/udpipe`

[4] `use_xpostag=0; provide_xpostag=0; use_feats=0; provide_feats=0; use_lemma=0; provide_lemma=0`

Target	fwd	rev	gdfa	gd	gdf	union	intersect
da	78.13	82.10	80.56	80.41	79.95	78.60	**84.02**
el	62.31	72.26	65.79	64.98	63.80	63.29	**73.06**
hu	64.23	64.13	65.80	66.41	62.41	64.46	**70.57**
id	77.22	77.84	77.68	77.81	77.49	77.01	**79.66**
ja	29.92	47.05	49.76	50.03	**50.77**	49.74	48.72
kk	49.25	50.45	52.71	**54.97**	53.16	51.51	53.31
lv	72.83	71.43	73.52	73.87	73.13	72.20	**75.16**
pl	74.72	77.45	75.20	76.00	75.19	75.00	**79.93**
sk	76.06	79.61	76.73	77.23	77.01	76.09	**80.98**
ta	41.73	40.14	42.68	42.28	41.81	41.09	**43.94**
tr	60.20	58.63	59.47	59.27	59.00	59.29	**62.36**
uk	69.71	69.71	68.05	68.05	68.05	66.80	**74.69**
vi	57.41	56.75	58.41	56.74	57.92	57.79	**59.69**
AVG	62.59	65.20	65.10	65.23	64.59	64.07	**68.16**

Table 6.1: Effect of alignment symmetrization on cross-lingual tagging accuracy. The aligner outputs forward (*fwd*) and reverse (*rev*) alignment links, which can be symmetrized by using the simpler *union* and *intersect* approaches, as well as the more advanced grow-diag (*gd*), grow-diag-final (*gdf*) and grow-diag-final-and (*gdfa*) symmetrizations.

in more detail, comparing the forward and reverse alignment, as well as all of the symmetrization heuristics available in the FastAlign aligner.

In Table 6.1, we compare the POS projection accuracies achieved when different alignment symmetrizations are used. After performing the POS projection through the word alignment links, a UDPipe tagger was trained on the resulting dataset, and then evaluated on the gold target treebank (dev section).

First, the results do partially confirm the claim of Agić et al. [2016], as with the reverse alignment, the average achieved accuracies are +2.5 % absolute higher than with the forward alignment. However, *all* of the symmetrization heuristics overpower the forward alignment. While most of them fare comparably to the reverse alignment, the intersection symmetrization clearly stands out, adding another +3 % absolute over the reverse alignment on average, and scoring best of all of the alignment flavours for 11 out of the 13 target languages. Based on this finding, we decided to use the intersection alignment as our default choice in all experiments.

6.1.3 Weighted projection

Agić et al. [2016] suggest to use the alignment score returned by the word aligner to improve the quality of the POS projection, although they do not describe the process in sufficient detail. It seems that in their case, the word aligner provides a score for each individual alignment link, while in our case, when FastAlign is set to output alignment score,[5] it only provides one for each aligned sentence pair (a large negative number). Nevertheless, we tried to use what we had, taking the $exp()$ of the score to push it into the $(0, 1)$ interval, and combining together the resulting scores from the forward and reverse alignment via multiplication. We

[5] `-s`

Weight	1	align score	$KL^{-4}_{cpos^3}$	KL_{cpos^3}	1 or $KL^{-4}_{cpos^3}$
da	84.02	84.09	**84.46**	**0.388**	**84.46**
el	73.06	72.55	**73.29**	**0.547**	**73.29**
hu	**70.57**	65.61	68.40	0.727	**70.57**
id	**79.66**	78.04	78.28	**0.614**	78.28
ja	**48.72**	31.82	43.78	1.139	**48.72**
kk	53.31	46.39	**53.61**	0.941	53.31
lv	**75.16**	70.22	73.21	0.710	**75.16**
pl	79.93	76.58	**80.80**	**0.548**	**80.80**
sk	80.98	81.82	**83.13**	**0.505**	**83.13**
ta	**43.94**	37.13	43.07	0.796	**43.94**
tr	**62.36**	56.73	60.89	**0.697**	60.89
uk	74.69	70.54	**75.10**	**0.594**	**75.10**
vi	**59.69**	58.39	59.61	0.920	**59.69**
AVG	**68.16**	63.84	67.51	0.702	**68.26**

Table 6.2: Effect of adding weighting into the POS projection setup. The base setup is an unweighted one, using each source language with a weight of 1. Then there are two setups which weight the source contributions, either by alignment score, or by $KL^{-4}_{cpos^3}$ source-target similarity. Finally, there is a setup conditioned on the lowest KL_{cpos^3} value of any source, using $KL^{-4}_{cpos^3}$ weighting if the KL_{cpos^3} value is < 0.7, and the base setup otherwise (i.e. each source weighted by 1).

then used this score to perform weighted voting in the POS projection, with all POS labels suggested by a given source weighted by this score.

We also tried to incorporate source-target language similarity estimates into the projection, using the $KL^{-4}_{cpos^3}$ measure, described in Section 4.2. In this experiment, instead of weighting the source votes by the alignment scores, we weight them by the source-target $KL^{-4}_{cpos^3}$ similarity, computed on the POS tags obtained by the unweighted projection. – so this time, the same source weight is used for all sentences.

Table 6.2 compares the accuracy of cross-lingual tagging with and without weighting. Surprisingly, adding the alignment scores makes the results much worse. Because Agić et al. [2016] reported an improvement, we suspect that they used a different way of extracting the alignment scores from the word aligner.

With adding the $KL^{-4}_{cpos^3}$ weighting, the situation is more subtle, as the results improve for one half of the target languages but deteriorate for the other half, with the overall effect being only slightly negative on average. A more careful look shows that the results tend to improve in cases where the achievable accuracies are high enough, i.e. in cases where there are some source languages sufficiently similar to the target languages, and it thus makes sense to prefer them in the projection. On the other hand, if all of the source languages are rather distant from the target language, it does not make much sense to differentiate among them, and simply using all of them in an unweighted scheme is the most reliable approach. In other words, it seems that if we do not have enough confidence to suppress some of the sources, we should just treat them all equally. Note that this is very similar to how $KL^{-4}_{cpos^3}$ usually behaves in the parse tree combination approach (Section 4.4 and Section 5.6).

We decided to perform one more experiment, conditioning the setup to use

Sources	All		Top 3		Top 5	
Weight	1	$KL^{-4}_{cpos^3}$	1	$KL^{-4}_{cpos^3}$	1	$KL^{-4}_{cpos^3}$
da	84.02	84.46	83.32	84.19	84.55	**85.28**
el	73.06	**73.29**	73.24	72.84	73.17	72.78
hu	70.57	68.40	**71.99**	68.10	63.73	64.67
id	**79.66**	78.28	74.80	75.14	77.32	76.14
ja	48.72	43.78	37.87	43.12	**52.08**	46.87
kk	53.31	53.61	50.00	51.96	**56.48**	54.67
lv	**75.16**	73.21	74.51	72.75	74.15	73.93
pl	79.93	**80.80**	79.38	79.93	78.45	79.86
sk	80.98	**83.13**	80.60	82.36	80.64	82.72
ta	43.94	43.07	**45.61**	45.29	42.91	43.63
tr	**62.36**	60.89	59.33	59.52	60.56	60.26
uk	74.69	75.10	73.86	**78.84**	77.18	75.93
vi	59.69	59.61	59.31	59.58	60.56	**60.63**
AVG	**68.16**	67.51	66.45	67.20	67.83	67.49

Table 6.3: Comparison of source subselection versus using all sources in POS tag projection.

on the KL_{cpos^3} value for the most similar source. Specifically, if the best KL_{cpos^3} is lower than its average value of 0.7, we use the weighted projection, and if it is higher, we use the unweighted projection.

While this turns out to be the best performing setup, choosing the better of the two weighting schemes for 10 of the 13 languages, the average improvement over the unweighted baseline is quite negligible (+0.1 % absolute). Therefore, based on Occam's razor, we do not feel confident adding such a potentially overtuned hyperparameter,[6] and rather select the unweighted projection as the approach to use.

6.1.4 Subselecting the sources

In Section 5.5.5, we found that for the lexicalized parse tree combination, combining only the top 5 most similar source languages worked quite well. We therefore also evaluated the effect of only using the top 3 or top 5 sources for POS projection, using both the weighted and the unweighted projection.[7]

As the results in Table 6.3 show, on average, none of the subselection setups outperforms the base of using all sources. Similarly to what we observed in Section 5.5.5, for individual target languages there is much variance in terms of which setup performs best; most of the setups achieve the best result in the table for 2 or 3 targets. Still, similarly to what we had observed before, higher accuracies are evidently obtainable by selecting the right setup for each target

[6]The results clearly show that with a bit more tuning, a threshold value of e.g. 0.6 would lead to even better results, selecting the better of the two approaches in 12 out of the 13 cases. However, we consider it to be highly risky to tune such a hyperparameter on only 13 data points, especially when the potential gains are rather moderate.

[7]For each of the setups, training a tagger on the target side of the parallel data is necessary, which makes the experiments rather demanding computationally, as training the tagger takes dozens of hours for some of the target languages. This is why we only evaluated a handful of setups.

language, but we fail to identify any clear rule or pattern to guide such selection.

On the other hand, it generally seems that using more sources leads to better results, as the "top 5" setup mostly outperforms the "top 3" setup, and the average accuracy approaches that of the base setup. We think this may be because the intersection alignment typically has "gaps", i.e. contains numerous unaligned words on both the source side and the target side. Using more sources lowers the probability of a particular target word not being aligned to any source word and thus not having any support for choosing the right POS tag for it. In that way, even more distant sources can improve the projection by providing at least some information in cases where it would be missing otherwise. It is thus possible that there *might* be an optimum number of sources to use, maybe 7 or 10 or 15... However, as these experiments are rather computationally demanding, we did not follow this question further.

Moreover, in the projection approach, the number of sources used does not influence the computational complexity of the approach much, since most of the computational power is spent on training the final tagger on the projected POS tags, independently of the number of sources used. We are thus content to use all of the available sources for the projection.

6.2 Machine-translating the training data

The other approach to cross-lingual tagging that we explore is machine translation of the source treebanks, which is practically the same method as the one we used for cross-lingual parsing in Section 5.5, only applied to POS tagging.

We first introduce the base approach in Section 6.2.1. We then deal with the specifics of applying it in the multi-source setting in Section 6.2.2, introducing the KL_{cpos^3} measure into the approach. Finally, we further improve it by using a simple self-training approach in Section 6.2.3

6.2.1 Base approach

The base approach can be summarized as follows:

1. train an MT system on source-target parallel data,
2. translate the word forms in a source treebank, keeping the POS annotation intact,
3. train a target POS tagger on the resulting synthetic target treebank.

Using MT to translate source training treebanks into the target language for cross-lingual POS tagger induction was introduced by Tiedemann [2014], applying the same approach to tagging and parsing. His setup was very similar to what we use in our work, featuring the GIZA++ word alignment and word-based Moses decoder with the KenLM language model.

In our base approach, we differ rather slightly from his approach. The original author seems to have used the usual grow-diag-final-and alignment symmetrization, as it performed quite well in his evaluation in comparison to other symmetrizations, although the evaluation was only performed for cross-lingual

parsing, not tagging. We use the intersection symmetrization instead, as we saw it perform better than other symmetrizations for cross-lingual POS projection in Section 6.1.2; however, we have not evaluated the effect of the symmetrization specifically in the MT setup.

We use monotone translation, as we observed it to perform better than translation with reordering in Section 5.5.3, although this evaluation was also performed only for cross-lingual parsing, not tagging. We assume that the original author used word reordering, as this is part of the usual setup of Moses, thus adding an additional step of reordering the POS annotation according to the alignment links output by Moses.

We train the MT system on the smaller but more realistic WTC instead of the larger Europarl, and therefore only use 3-grams instead of 5-grams in the language model.

The tagger trained on the translated treebank is UDPipe in our case, while Tiedemann [2014] used the HunPos tagger of Halácsy et al. [2007]. However, we do not expect this difference to have a dramatic influence on the results.

6.2.2 Multi-source setting

Similarly to cross-lingual parsing, it is realistic to assume that multiple treebanks are available to use as the sources for translation into the target language and training of the target parser. We thus revisit the ideas from Chapter 4 of source selection and source combination, based on the $KL^{-4}_{cpos^3}$ language similarity measure, and apply them to cross-lingual tagging.

Moreover, as we already explained in Section 5.5.5, using *all* of the available source languages is not viable, as training so many MT systems is too computationally demanding. Therefore, selecting only one or a few sources to use for each given target is an inevitable first step in the processing pipeline.

We would also like to emphasize that the $KL^{-4}_{cpos^3}$ measure was explicitly designed and tuned for use in cross-lingual *parsing*. It is thus very interesting to observe its performance for the considerably different task of cross-lingual POS tagging.

Chicken-egg problem

In the cross-lingual POS tagging, we first need to deal with a chicken-and-egg problem. All of the multi-source methods that we wish to apply here use the $KL^{-4}_{cpos^3}$ source-target language similarity measure in one way or another; but to compute it, we need POS tagged data in both the source and the target language. While for cross-lingual parsing, we assumed that we somehow have the target POS tags, we obviously cannot assume that for cross-lingual tagging.

Therefore, even though we investigate the treebank translation method as an *alternative* to the POS projection on multi-parallel data (Section 6.1), we actually do need to first use the projection even in this setup, to obtain initial target POS tagging which we need to compute the $KL^{-4}_{cpos^3}$ measure. This is all due to the fact that it is computationally much less demanding to only compute the word alignment with FastAlign for all source-target language pairs than it is to build instances of a full-fledged Moses SMT system.

Target	1st	2nd	3rd	4th	5th	max	1st-max
da	86.56	**87.99**	82.25	74.97	65.04	87.99	-1.43
el	68.62	68.57	67.11	**69.75**	67.97	69.75	-1.13
hu	56.81	**62.96**	58.29	59.86	56.31	62.96	-6.15
id	61.30	**73.22**	66.75	68.27	67.94	73.22	-11.92
ja	31.57	36.26	24.39	33.55	**36.80**	36.80	-5.23
kk	**54.07**	48.95	51.51	47.29	50.00	54.07	0.00
lv	61.54	64.31	70.30	69.04	**73.08**	73.08	-11.54
pl	**81.17**	75.07	71.54	70.55	70.26	81.17	0.00
sk	**86.59**	78.02	75.69	75.14	75.00	86.59	0.00
ta	40.46	36.03	**47.35**	30.32	41.41	47.35	-6.89
tr	**57.74**	49.84	52.55	57.09	47.91	57.74	0.00
uk	**83.40**	68.46	70.12	65.15	72.61	83.40	0.00
vi	49.02	48.01	50.96	50.32	**52.03**	52.03	-3.01
AVG	**62.99**	61.36	60.68	59.33	59.72	66.63	-3.64
Best	5	3	1	1	3		

Table 6.4: Evaluation of the cross-lingual POS tagging accuracy, training the target tagger on a translation of the treebank for the Nth most similar source language, as measured with KL_{cpos^3}.

Fortunately, we can omit the last step of the projection approach, i.e. training a tagger on the target side of the projectedly-tagged parallel data, as we can (and do) simply use these tagged data to estimate the $KL^{-4}_{cpos^3}$ similarity. This makes the whole process significantly more efficient, as it typically only takes a few dozens of *seconds* to word-align the WTC with FastAlign and to project the POS tags onto the target language, while it then often takes a few dozens of *hours* to train a tagger on the projected tags.

Single-source

First, we examine using KL_{cpos^3} to select the closest source language to use for a single-source transfer, i.e. training the target tagger on the translation of only one source treebank.

As we already explained, we did not train the MT system for all possible source-target combinations. We thus have no knowledge of the optimal source to use in the single-source setting (i.e. the oracle). Instead, Table 6.4 lists the tagging accuracies obtained when using one of the top 5 sources according to the predictions of KL_{cpos^3}. We also list the maximum accuracy achieved for each target, as an approximation of the oracle, and the difference between the maximum and the source selected by KL_{cpos^3}.

Although there is considerable variance in the results, we can see that KL_{cpos^3} actually performs rather well in this task, selecting the best of the evaluated source languages in 5 cases. Also the average accuracy is highest when using the sources predicted as the first most similar, with a loss of less than 4 % absolute towards the maximum. The accuracies also often tend to decrease with moving towards the sources predicted to be less similar to the target.

The results suggest that KL_{cpos^3} can indeed be useful even in cross-lingual tagging. We thus proceed by applying it to both subselection and weighting of

Target	top 1	top 3	top 4	top 5	top 6	top 7
da	86.56	88.53	87.87	**89.30**	88.74	88.14
el	68.62	73.22	75.29	75.59	76.06	**76.42**
hu	56.81	64.78	66.87	66.03	70.82	**70.95**
id	61.30	74.85	75.02	77.01	**79.09**	78.47
ja	31.57	32.46	34.24	**42.04**	39.91	37.68
kk	54.07	56.63	54.22	56.33	**59.34**	57.83
lv	61.54	70.41	75.22	78.85	78.38	**79.15**
pl	81.17	82.55	82.85	**84.29**	83.49	83.19
sk	**86.59**	85.84	84.99	84.25	84.25	84.53
ta	40.46	44.10	45.37	49.09	47.35	**49.64**
tr	57.74	56.63	60.13	61.50	62.79	**62.91**
uk	**83.40**	78.84	78.42	77.59	78.84	78.84
vi	49.02	53.90	54.57	56.10	55.69	**56.46**
AVG	62.99	66.36	67.31	69.07	**69.60**	69.55
Best	2	0	0	3	2	**6**

Table 6.5: Evaluation of the unweighted multi-source cross-lingual POS tagging accuracy, combining the top N sources as ordered by KL_{cpos^3}.

the source languages.

Unweighted multi-source combination

We now move on to an unweighted combination of multiple sources. This is again a voting-based method, very similar to that used in the multi-source POS projection (Section 6.1) or the dependency relation label assignment in parse tree combination (Section 4.3.1).

For each target language, we first select several sources languages to use, based on their KL_{cpos^3} distance from the target. We then translate each of the source treebanks into the target language, and train a target tagger on it. Each target sentence is then tagged by each of the target taggers, and each word is assigned the POS tag predicted for it by the largest number of the taggers.

Table 6.5 shows that combining multiple sources helps a lot, much more than what we saw for parse tree combination. We believe that for parsing, the grammatical similarity of the source and target languages matters a lot, as the same sequences of POS tags can signify very different syntactic structures, and thus not much can usually be learned from distant source languages. In POS tagging, however, it can be expected that e.g. a noun in one language typically corresponds to a noun in most other languages, and even more distant sources can thus contribute usefully. We already saw this effect in POS tag projection, where we were unable to find a subselection of the sources that would clearly improve over a combination of all of the sources.

For the individual target languages, we mostly see an approximate pattern of the accuracies improving with adding more sources, culminating at one point, and then gradually deteriorating with the addition of even more sources. For some targets, the culmination point is already at using 1 source only, for others, it is e.g. at the combination of 5 sources, but the most common situation is that the culmination point is either at 7 sources or beyond (which we cannot distinguish,

Target	top 1	top 3	top 4	top 5	top 6	top 7	unw. top 7
da	86.56	88.72	88.86	**88.88**	88.82	88.86	88.14
el	68.62	73.24	74.20	75.28	75.76	**76.04**	**76.42**
hu	56.81	64.51	66.79	65.01	69.76	**70.32**	**70.95**
id	61.30	61.30	71.08	73.83	76.44	**76.70**	**78.47**
ja	31.57	32.83	35.67	36.79	37.28	**39.01**	37.68
kk	54.07	56.02	56.48	57.98	57.98	**59.64**	57.83
lv	61.54	71.48	74.86	76.98	77.61	**77.94**	**79.15**
pl	81.17	83.46	83.74	**85.04**	84.84	84.96	83.19
sk	86.59	86.35	**86.84**	86.33	86.39	86.49	84.53
ta	40.46	40.46	42.68	45.84	46.00	**47.43**	**49.64**
tr	57.74	59.49	59.96	60.63	61.41	**62.16**	**62.91**
uk	83.40	**83.82**	79.67	81.33	82.16	80.91	78.84
vi	49.02	52.55	54.14	55.24	55.42	**56.10**	**56.46**
AVG	62.99	65.71	67.30	68.40	69.22	**69.73**	69.55
Best	0	1	1	2	0	**9**	7

Table 6.6: Evaluation of the weighted multi-source cross-lingual POS tagging accuracy, combining the top N sources as ordered by KL_{cpos^3}. Also listing the top 7 setup from the unweighted combination.

as we did not evaluate using more than 7 sources).

The average accuracy is actually highest for the combination of the top 6 sources. However, as the top 7 combination surpasses it for the majority of targets while performing only negligibly worse on average, we see the "top 7" combination as the most promising setup.

Weighted multi-source combination

The weighted variant of the method proceeds in the same way, but weights the votes of the sources by their $KL^{-4}_{cpos^3}$ similarity to the target.

Table 6.6 shows that the weighted approach quite successfully avoids the problem of the unweighted one, shifting the culmination point to the top 7 combination for most targets (9 out of 13), and reaching only a tiny loss towards the best performing combination for 3 of the 4 remaining targets (0.02, 0.08 and 0.35); only for Ukrainian, top 7 is considerably lower than top 3, with a loss of nearly 3 percentage points.

This shows that even in cross-lingual tagging, $KL^{-4}_{cpos^3}$ is capable of evening out the accuracies across the varying number of sources used, similarly to what we observed in cross-lingual parsing. As we are adding more sources, they are getting weaker and weaker weights, and so, instead of culminating and then deteriorating as we saw for the unweighted combination, the accuracy rather tends to level out, preserving what we have and avoiding bad losses.

On the other hand, as we have also already seen before, this at the same time seems to prevent possible larger gains, i.e. the evening out works both ways. This is demonstrated by the unweighted top 7 combination outperforming the weighted one for half of the targets (these are mostly the same cases for which top 7 was best among the unweighted combinations), even though it achieves a slightly worse average accuracy.

Target	Top 7	Retrain
da	**88.86**	88.77
el	76.04	**76.75**
hu	70.32	**71.49**
id	76.70	**78.56**
ja	39.01	**43.64**
kk	**59.64**	59.34
lv	77.94	**79.97**
pl	84.96	**86.38**
sk	86.49	**86.95**
ta	47.43	**47.90**
tr	62.16	**63.52**
uk	80.91	**82.57**
vi	56.10	**58.48**
AVG	69.73	**71.10**

Table 6.7: Evaluation of a simple self-training approach, applying the weighted combination of the top 7 sources to WTC target side and retraining the tagger on it.

Nevertheless, although the gains in accuracy are not completely convincing, we believe the weighted combination to be a better method than the unweighted one, since it is more stable and predictable.

6.2.3 Simple self-training

Probably the most important drawback of the MT approach, as compared to the projection approach, is the fact that we train the target taggers on synthetic texts, which may be very different from real texts in the target language. In the projection approach, this is not the case, as the final tagger is trained on the target side of the parallel corpus, i.e. real target language texts (while the main drawback of the projection method is the projection of automatic POS tags, whereas the MT method directly uses gold standard source annotations).

Here, we try to somewhat rectify that shortcoming by introducing real target texts into the process, using a simple self-training approach. Specifically, we apply the weighted multi-source cross-lingual POS tagging to the target side of the parallel data, and then train a new POS tagger on that data.

As Table 6.7 shows, this approach is clearly successful, leading to an increase of accuracy for 11 of the 13 target languages, on average improving the accuracy by +1.4 percentage points.

The tagger thus benefits from training on larger data, even if these are tagged noisily. We hypothesize that, while it seems unlikely that the final tagger would learn different tags for words that were known already to the underlying taggers, it may arrive at a better grasp of the POS sequences usual in the target language, allowing it to better disambiguate the correct POSes for known but ambiguous words, as well as to bridge words unknown to it.

Also, as our MT setup is a simple one, its outputs contain many untranslated words, and the tagger trained on the translated treebank is thus taught to tag not only the translated target words, but also the source words, which may confuse

it, as well as waste its learning capacity on words that it will never encounter once applied to real target data.

Moreover, we believe that the fact that we use particularly the target side of the parallel data, i.e. the training data for the MT system, may also be of importance, as it consists of words that are bound to be best known to the taggers; we would expect the taggers to work especially well on such data. However, this is hard to measure reliably, as we do not have the gold POS tags for the parallel data.

6.3 Comparison and combination

We now compare the performances of the two individual cross-lingual POS tagging methods that we described, in their settings which we believe to be the best, based on our experiments. For POS projection over multiparallel data, we use the unweighted combination of all sources and the intersection alignment symmetrization. In the treebank machine translation approach, we use the weighted combination of the 7 closest sources with the simple self-training.

The results in the first two columns of Table 6.8 clearly show the superiority of the machine translation approach, outperforming the projection approach for 10 of the 13 targets, and reaching an accuracy higher by 3 percentage points on average. We believe that the strongest advantage of the MT method is the utilization of gold source annotation instead of the automatic ones, while using automatic translations instead of human translations might not be such a weak point as it would seem to be, due to the machine translations being more literal than the human translations, especially given our monotone word-based setting for the MT system.

Interestingly, the results of our evaluation go against the claims made by Tiedemann and Agić [2016] and Agić et al. [2016], who assume that small parallel corpora, such as the WTC, are insufficient for building an MT system sufficiently accurate to be used for the treebank translation approach to cross-lingual analysis, without actually providing any support for such claims. While the rather poor results of the MT approach which we observed in the single-source setting may suggest this, the subsequent combination of multiple sources leads to surprising accuracy gains, eventually surpassing the projection method. It is possible that Tiedemann and Agić [2016] did not take such an option into account, as we have found no mention of a multi-source combination approach to the treebank translation method in their work, even though it is directly based on the methods previously introduced by these authors. We also believe that we benefit here from using the simpler word-based approach to MT, while the authors seem to have assumed using the more complex phrase-based MT. However, as we showed in Section 5.6, the accuracy of the underlying MT system seems to only have a moderate effect on the accuracy of the cross-lingual analysis.

Nevertheless, this does not mean that we can completely abandon the projection method, as we still need it to provide us with an initial tagging to compute the $KL^{-4}_{cpos^3}$ similarity for source subselection and weighting. Moreover, it is interesting to note that the accuracy of the projection method is still rather close to that of the MT method, even though its approach is considerably different.

This naturally leads to the idea of combining the two methods in an ensemble

Target	Projection	MT	Ensemble	Retrain	Supervised
da	84.02	**88.77**	**89.08**	87.75	95.81
el	73.06	**76.75**	**77.02**	**77.65**	97.45
hu	70.57	**71.49**	**73.60**	73.00	94.27
id	**79.66**	78.56	**80.66**	**81.20**	93.33
ja	**48.72**	43.64	44.54	**53.76**	96.97
kk	53.31	**59.34**	**60.84**	**62.05**	81.02
lv	75.16	**79.97**	**81.37**	**81.90**	90.30
pl	79.93	**86.38**	85.64	85.64	94.87
sk	80.98	**86.95**	**87.00**	86.88	92.07
ta	43.94	**47.90**	**51.39**	50.04	84.32
tr	62.36	**63.52**	**65.93**	**66.34**	92.19
uk	74.69	**82.57**	78.42	78.42	70.95
vi	**59.69**	58.48	59.03	**60.91**	88.23
AVG	68.16	71.10	71.89	**72.73**	90.14

Table 6.8: Comparison of the two cross-lingual POS tagging methods – POS projection and treebank machine translation – in their best preforming settings, with the better performing one marked in bold. We also include accuracy of their ensemble combination (marked in bold if it surpasses both of the individual methods), and the self-training approach applied in top of the ensemble method (marked in bold if it surpasses the non-retrained ensemble). Accuracy of a fully supervised monolingual tagger, trained on the target treebank, is also reported.

fashion. We implement this idea by incorporating the tagger trained on the projected POS into the voting; i.e. we now combine 8 taggers, 7 trained on the translated source treebanks, and one trained on the projectedly tagged target side of the multiparallel corpus. As we use the weighted voting, we need to assign a weight to the additional tagger; we use the value of $KL^{-4}_{cpos^3}$ for the closest source. Additionally, we also apply the self-training on top of the ensemble, by using the ensemble tagging to assign POS tags to the target side of the multiparallel corpus and then training the final tagger on this data.

Table 6.8 shows the ensemble to improve over both of the individual methods for 9 of the 13 targets, as well as on average (+0.7). This shows that, while the two methods achieve a similar accuracy, their strengths and weaknesses somewhat differ, and it thus makes sense to combine them.

Unsurprisingly, the ensemble seems to perform particularly well if both of the methods yield comparably good accuracies (*hu, tr*). If one of the methods performs substantially better than the other, the ensemble often fails to improve over it (*ja, pl, uk*), or does so with only a small gain (*da, sk*). Nevertheless, the combination always outperforms at least the weaker of the two underlying methods.

The self-training approach does further improve the average performance by another +0.7 percentage points, but leads to a drop in accuracy for 4 target languages. This is in slight contrast with our observations when the self-training was applied on top of the base treebank translation method, where the average improvement was twice as large, and minor deteriorations only occurred for 2 targets. We believe that the effect of mixing in the projection tagger is somewhat complementary to the self-training approach, as both of these benefit from the

Target lang.	Top 7 unw.	Top 7 w.	Retrain	Ensemble
Danish	59.11	**59.51**	59.37	57.68
Greek	47.69	47.74	48.70	**49.41**
Hungarian	26.96	29.95	30.79	**32.57**
Indonesian	37.16	37.88	38.72	**41.11**
Japanese	11.73	10.05	9.20	**14.58**
Kazakh	11.14	12.35	12.95	**13.86**
Latvian	37.53	38.05	41.10	**42.23**
Polish	55.45	56.00	57.51	**58.46**
Slovak	59.74	62.52	**63.46**	62.21
Tamil	14.96	13.06	14.57	**16.39**
Turkish	19.51	18.40	18.96	**22.47**
Ukrainian	51.87	51.04	**54.36**	48.55
Vietnamese	25.95	26.00	26.95	**27.90**
AVG LAS	35.29	35.58	36.66	**37.49**
AVG POSacc	69.55	69.73	71.10	**72.73**

Table 6.9: Cross-lingual parsing LAS based on the tagging setup used – unweighted combination of the top 7 sources, weighted combination of the top 7 sources, self-training using the weighted top 7 combination, and self-training on top of the ensemble setup. The last line lists the average POS tagging accuracy of the underlying cross-lingual tagging setup.

tagger being exposed to the target side of the parallel corpus. Nevertheless, the self-training on the ensemble outputs still does bring improvements, including a particularly large gain for Japanese (+9.2), and we thus designate it as our final ultimate best-performing approach to cross-lingual POS tagging.

The absolute accuracies of the taggers are far from perfect, as been seen when comparing them to the supervised tagger. Still, they are rather respectable, often surpassing 80%, suggesting that the tagging may be already useful for downstream applications, most notably for cross-lingual parsing. The supervised performance is approached for some targets, especially for Slovak, where the huge treebank for a very similar source language (Czech) brings remarkably good results. For Ukrainian, the supervised tagger is even surpassed, as the Ukrainian UD 1.4 training data are very small (about 1000 tokens). This nicely simulates the under-resourced scenario, showing that our methods can outperform supervised methods if training data is scarce.

6.3.1 Influence on parsing

We now show the influence of improvements in the cross-lingual POS tagging to cross-lingual dependency parsing, evaluated within the best parsing setup from Chapter 5 (weighted parse tree combination of 5 closest source parsers, lexicalized using word-based monotone Moses treebank translation).

Table 6.9 compares the LAS achieved in parsing on top of POS tags provided by four of the tagging setups that we evaluated – unweighted combination of the taggers trained on machine translated treebanks for the 7 closest source languages, the weighted variant thereof, the self-training applied on top of the weighted combination, and the ensemble of the translation and projection approaches with

self-training on top.

We can see that improvements in tagging accuracy generally tend to lead to corresponding improvements in parsing accuracy, both on average as well as for the individual target languages (although there is, as could be expected, some variance in the numbers). The weighted combination is slightly better than the unweighted one on average as well as for 8 of the 13 target languages. Self-training leads to an improvement of all but two targets and adds 1 LAS point on average. And ensembling with self-training leads to a further improvement for 10 of the 13 languages, adding another +0.9 LAS on average (for all the three target languages that experienced a deterioration in LAS, there was also a decrease of tagging accuracy).

On average, the table shows the tagging accuracy rising by 3.2 percentage points from the weakest setup to the strongest one, while the corresponding LAS rises by 2.2 points.

Most importantly, from the results we can see that improvements in tagging accuracy generally tend to lead to improvements in parsing accuracy, justifying the approach of dealing with tagging and parsing independently and then combining the best performing setups to form the final system.

Interestingly, we obtained best results when the parsers were trained on gold POS tags, even though better results are usually achieved by training on POS tags predicted by the tagger that will eventually be used in the inference, as this helps the parsers to know what POS tags they can expect and adapt to that. Moreover, this effect could be expected to be even stronger in the cross-lingual setting, where the inference-stage POS tags are typically considerably different from the gold POS tags. However, in our setup, the machine-translated treebanks which we use for training the parsers are actually quite different from the real target-language texts that both the taggers and the parsers will be eventually applied to, which probably weakens this effect.[8]

6.4 Summary

In this chapter, we investigated two pre-existing approaches to cross-lingual POS tagging.

First, we reimplemented the method of projecting POS tags over multiparallel data, suggested by Agić et al. [2016]. We found the intersection word-alignment to outperform the reverse word-alignment suggested by the original authors. On the other hand, we were unable to improve the performance of the method by incorporating source weighting based on alignment scores or $KL^{-4}_{cpos^3}$ source-target similarity, and we also did not find a subselection of the sources that would perform better than using all of the sources.

Next, we examined the method of training target language taggers on source language treebanks translated with a word-based MT system, which was introduced by Tiedemann [2014]. We showed the effectiveness of KL_{cpos^3} for selecting appropriate source languages to use for a given target language, noting the need to first use the projection method to obtain initial target POS tags for the com-

[8]Training the parsers on gold POS tags led to the parsing accuracies being higher by approximately +1.5 LAS point on average.

putation of KL_{cpos^3}. We then successfully extended the single-source method into a voting-based multi-source combination, and further improved it by the introduction of weighted voting, with source weights computed as the $KL^{-4}_{cpos^3}$ source-target similarities. We obtained further gains by performing simple self-training, tagging the target side of the parallel corpus and retraining a tagger on that, thus allowing it to encounter real target language texts in training.

We then compared performances of the two methods, finding the translation-based method to consistently outperform the projection-based one, although not drastically. On the other hand, the projection is quite light-weight and can be easily performed for all language pairs, while the translation is computationally demanding and needs to only be applied on a subselection of sources for each target. Subsequently, we also tried to combine the two methods together, finding the ensemble to outperform both of the individual methods. Moreover, applying the self-training on top of the ensemble led to a further moderate improvement of tagging accuracy.

Finally, we compared several of the tagging setups within our final approach to cross-lingual parsing, confirming that more accurate POS tagging tends to lead to improvements in the subsequent parsing.

We tried to realistically focus on the intended use case for the cross-lingual methods, using only a small parallel corpus, which, however, is available for a large number of languages, including many under-resourced ones. This also allowed us to investigate a larger number of setups, since the computational resources necessary to perform the experiments are typically proportional to the size of the datasets used. We assume that operating on larger datasets would further improve the resulting accuracies, while it is not clear whether the same methods would still be the best performing ones under such settings. Nevertheless, such datasets are typically not available for under-resourced languages, thus probably making this uncertainty to be of little importance.

Conclusion, or How to parse an under-resourced language

As a conclusion of the thesis, we summarize our findings in the form of a set of instructions for parsing an under-resourced target language for which no annotated data are available.

Get source treebanks

You will need harmonized dependency treebanks for some source languages; ideally for some languages which are close to the target language, but even distant languages can help. The Universal Dependencies treebank collection is currently the best such resource in existence.

Get parallel data

Obtain source-target parallel data for your target language and the source languages for which you have treebanks; multiparallel data are even better. The Watchtower texts seem to be very good for that purpose, and available for many under-resourced languages. Other religious texts are also available in many languages, especially the Bible.

Get monolingual target data

The target side of the parallel data can be used as monolingual target data. For some target languages, other larger monolingual data may be available, such as the Wikipedia texts.

Tokenize the parallel data

If the target language uses word spaces and punctuation, a simple rule-based tokenizer can be used. If not, a specialized tokenizer should be applied. For the source languages, a tokenizer can be trained on the source treebanks.

Align the parallel data

If the parallel data are not sentence-aligned, this has to be performed first; the Hunalign tool can be used for that. For word alignment, there is a range of existing tools, such as FastAlign or GIZA++. The intersection symmetrization should be applied to the produced alignment, as other symmetrizations are too noisy and also more difficult to work with.

Train source part-of-speech taggers

It is recommended to train a tagger that only predicts the coarse POS tags, as more fine-grained morphological labels do not seem to be sufficiently cross-lingual. We were satisfied with using the UDPipe tagger.

POS-tag the parallel data

Use the trained taggers to assign POS tags to the source sides of the parallel data.

Project POS tags over the word alignment

For each target token, determine its POS as the most frequent POS assigned to the source words aligned to it. This will give you initial target POS tags

Measure language similarity

Using the POS-tagged source and target sides of the parallel data, compute the $KL^{-4}_{cpos^3}$ language similarity for each source-target pair.

Train machine translation systems

For several of the source languages which seem to be the most similar to the target language (5 seems like a nice number), train a source→target MT system, such as Moses. Use only word-to-word translation without reordering. Employ a language model, trained on the target data you have.

Translate the source treebanks

Translate the word forms in the source treebanks into the target language. Using a word-based monotone MT system ensures that the annotation can be kept intact.

Train target word embeddings

Apply the word2vec to the target data to obtain target word embeddings. If you use a neural dependency parser in the next step, providing it with the pre-trained embeddings can help adapting it to the target language (but is typically useful even in a monolingual scenario).

Train tagger and parser

Train target taggers and parsers on the translated treebanks. We were satisfied with using UDPipe. Use the gold POS tags for training the parser.

Retag the target data

Retag the target data with the POS taggers trained on the translated treebanks. Determine the the final POS tag for each token by weighted voting, with the weight of the vote of each tagger determined by the $KL^{-4}_{cpos^3}$ similarity of its source language to the target language. Also include the current POS tags, obtained through projection over word alignment, into the voting, with a weight identical to that of the closest source.

Train the final tagger

Train the final tagger on the POS-tagged target data. Training the tagger on real target language data instead of the machine translation outputs makes it better. Also, it is more practical, as it results in a single standard parser that can be directly applied to target language texts, rather than having to perform the multi-source tagger combination each time.

Parse the target data

Tag the target data with the final tagger, and parse it with the dependency parsers trained on the translated treebanks. Score each possible target dependency edge by the sum of the $KL^{-4}_{cpos^3}$ similarities of the sources of all of the parsers that predicted that edge. Find the final parse tree by applying the Maximum Spanning Tree algorithm to the resulting weighted directed graph. For each dependent node, determine the label for the dependency relation to its head node through weighted voting on the labels predicted by all of the parsers, again using $KL^{-4}_{cpos^3}$ weights.

Optional: Train a final parser

A final parser can be trained on the parsed target data. However, in the case of the parser, this seems to actually decrease the accuracy, presumably because this way it gets trained on better target texts but worse POS tags; to achieve the highest accuracy of parsing, this step thus should probably be skipped. On the other hand, having one final parser which can be directly applied to target language sentences instead of a pool of parsers whose outputs need to be combined to get the parse tree may obviously be more handy in practice.

Use the final tagger and parser

Apply the final tagger and parser (or the parser combination) to any tokenized texts in the target language.

Expect low accuracy

The accuracy of the results depends on many factors, especially on the similarity of the available source languages to the target language and on the sizes of the available parallel data and source treebanks. Very roughly, with using the WTC multiparallel data, the accuracy of the POS tagger can be expected to be approximately $(70 \pm 20)\%$, and the labelled attachment accuracy of the parser approximately $(35 \pm 20)\%$. While this may be insufficient for most practical purposes, it is, to the best of our knowledge, about the best you can get.

Bibliography

Itzair Aduriz, María Jesús Aranzabe, Jose Mari Arriola, Aitziber Atutxa, Arantza Díaz de Ilarraza, Aitzpea Garmendia, and Maite Oronoz. Construction of a Basque dependency treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories*, 2003.

Susana Afonso, Eckhard Bick, Renato Haber, and Diana Santos. "Floresta sintá(c)tica": a treebank for Portuguese. In *Proc. of LREC*, pages 1968–1703, 2002.

Željko Agić. Cross-lingual parser selection for low-resource languages. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 1–10, Gothenburg, Sweden, May 2017. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W17-0401`.

Željko Agić, Danijela Merkler, and Daša Berovic. Slovene-Croatian treebank transfer using bilingual lexicon improves Croatian dependency parsing. In *Proceedings of IS-LTC*, pages 5–9, 2012.

Željko Agić, Jörg Tiedemann, Kaja Dobrovoljc, Simon Krek, Danijela Merkler, and Sara Može. Cross-lingual dependency parsing of related languages with rich morphosyntactic tagsets. In *EMNLP 2014 Workshop on Language Technology for Closely Related Languages and Language Variants*, 2014.

Željko Agić, Dirk Hovy, and Anders Søgaard. If all you have is a bit of the Bible: Learning POS taggers for truly low-resource languages. In *The 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2015)*. Hrvatska znanstvena bibliografija i MZOS-Svibor, 2015.

Željko Agić, Anders Johannsen, Barbara Plank, Héctor Alonso Martínez, Natalie Schluter, and Anders Søgaard. Multilingual projection for parsing truly low-resource languages. *Transactions of the Association for Computational Linguistics*, 4:301, 2016.

Alfred V Aho and Jeffrey D Ullman. *The theory of parsing, translation, and compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ, 1972.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A Smith. One parser, many languages. *CoRR, abs/1602.01595*, 2016.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 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)*, volume 1, pages 451–462, 2017.

Nart B. Atalay, Kemal Oflazer, Bilge Say, and Informatics Inst. The annotation process in the Turkish treebank. In *Proc. of the 4th Intern. Workshop on Linguistically Interpreteted Corpora (LINC)*, 2003.

Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 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*, pages 119–130, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL `http://aclweb.org/anthology/C16-1012`.

David Bamman and Gregory Crane. The Ancient Greek and Latin dependency treebanks. In Caroline Sporleder, Antal Bosch, and Kalliopi Zervanou, editors, *Language Technology for Cultural Heritage*, Theory and Applications of Natural Language Processing, pages 79–98. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-20227-8.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3 (Feb):1137–1155, 2003.

Taylor Berg-Kirkpatrick and Dan Klein. Phylogenetic grammar induction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1288–1297, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1858681.1858812`.

Eckhard Bick, Heli Uibo, and Kaili Müürisep. Arborest – a VISL-style treebank derived from an Estonian constraint grammar corpus. In *Proc. of Treebanks and Linguistic Theories*, 2004. URL `http://beta.visl.sdu.dk/pdf/Bick_Uibo_Muurisep_TLT04.pdf`.

Igor Boguslavsky, Svetlana Grigorieva, Nikolai Grigoriev, Leonid Kreidlin, and Nadezhda Frid. Dependency treebank for Russian: Concept, tools, types of information. In *Proc. of the 18th conference on Computational linguistics-Volume 2*, pages 987–991. Association for Computational Linguistics Morristown, NJ, USA, 2000.

Bernd Bohnet and Joakim Nivre. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465. Association for Computational Linguistics, 2012.

Ondřej Bojar, Kamil Kos, and David Mareček. Tackling sparse data issue in machine translation evaluation. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 86–91, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1858842.1858858`.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. TIGER: Linguistic interpretation of a German corpus. *Journal of Language and Computation*, 2(4):597–620, 2004. Special Issue.

Thorsten Brants. TnT: A statistical part-of-speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, ANLC '00, pages 224–231, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. doi: 10.3115/974147.974178. URL `https://doi.org/10.3115/974147.974178`.

Sabine Buchholz and Erwin Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 149–164, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Sean M. Burke. Unidecode! *Sys Admin*, 10(12):54–60, December 2001. ISSN 1061-2688. URL `http://interglacial.com/tpj/22/`.

Xavier Carreras. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, volume 7, pages 957–961, 2007.

Giovanni Cavallanti, Nicolo Cesa-Bianchi, and Claudio Gentile. Linear algorithms for online multitask classification. *The Journal of Machine Learning Research*, 11:2901–2934, 2010.

Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, September 1956. ISSN 0096-1000. doi: 10.1109/TIT.1956.1056813.

Christos Christodouloupoulos and Mark Steedman. A massively parallel corpus: the Bible in 100 languages. *Language resources and evaluation*, 49(2):375–395, 2015.

Yoeng-Jin Chu and Tseng-Hong Liu. On shortest arborescence of a directed graph. *Scientia Sinica*, 14(10):1396, 1965.

Martin Čmejrek, Jan Cuřín, Jiří Havelka, Jan Hajič, and Vladislav Kuboň. Prague Czech-English dependency treebank: Syntactically annotated resources for machine translation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, volume V, pages 1597–1600, Lisboa, 2004. European Language Resources Association. ISBN ISBN 2-9517408-1-6.

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 50–61, Stroudsburg, PA, USA, 2011. ACL. ISBN 978-1-937284-11-4. URL `http://dl.acm.org/citation.cfm?id=2145432.2145439`.

Michael Collins. Discriminative training methods for Hidden Markov Models: Theory and experiments with Perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing -*

Volume 10, EMNLP '02, pages 1–8, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1118693.1118694. URL https://doi.org/10.3115/1118693.1118694.

Michael Collins. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637, 2003. doi: 10.1162/089120103322753356. URL https://doi.org/10.1162/089120103322753356.

Michael Collins, Lance Ramshaw, Jan Hajič, and Christoph Tillmann. A statistical parser for Czech. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 505–512, Stroudsburg, PA, USA, 1999. Association for Computational Linguistics.

Michael Collins, Philipp Koehn, and Ivona Kučerová. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 531–540. Association for Computational Linguistics, 2005.

Çağrı Çöltekin and Taraka Rama. Tübingen system in vardial 2017 shared task: Experiments with language identification and cross-lingual parsing. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 146–155, 2017.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. *CoRR*, abs/1710.04087, 2017. URL http://arxiv.org/abs/1710.04087.

Michael A Covington. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th annual ACM southeast conference*, pages 95–102. Citeseer, 2001.

Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *The Journal of Machine Learning Research*, 3:951–991, 2003.

Mathias Creutz and Krista Lagus. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology Helsinki, 2005.

Dóra Csendes, János Csirik, Tibor Gyimóthy, and András Kocsor. The Szeged treebank. In Václav Matoušek, Pavel Mautner, and Tomáš Pavelka, editors, *Text, Speech and Dialogue*, pages 123–131, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31817-0.

Jan Cuřín, Martin Čmejrek, Jiří Havelka, Jan Hajič, Vladislav Kuboň, and Zdeněk Žabokrtský. Prague Czech-English dependency treebank version 1.0. In *Linguistic Data Consortium (LDC)*, number LDC2004T25, University of Pennsylvania, 2004. Linguistic Data Consortium (LDC). ISBN 1-58563-321-6.

Mihaela Călăcean. Data-driven dependency parsing for Romanian. Master's thesis, Uppsala University, 2008. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.153.6068&rep=rep1&type=pdf.

Dipanjan Das and Slav Petrov. 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-Volume 1*, pages 600–609. Association for Computational Linguistics, 2011.

Marie-Catherine De Marneffe and Christopher D Manning. The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8. Association for Computational Linguistics, 2008.

Marie-Catherine de Marneffe, Natalia Silveira, Timothy Dozat, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of LREC'14*, Reykjavík, Iceland, 2014. ELRA.

Ludmila Dimitrova, Nancy Ide, Vladimir Petkevic, Tomaz Erjavec, Heiki Jaan Kaalep, and Dan Tufis. 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*, ACL '98, pages 315–319, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics. doi: 10.3115/980845.980897. URL `https://doi.org/10.3115/980845.980897`.

Matthew S. Dryer and Martin Haspelmath, editors. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig, 2013. URL `http://wals.info/`.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. Cross-lingual transfer for unsupervised dependency parsing without parallel data. In *CoNLL*, pages 113–122, 2015.

Nadir Durrani, Hassan Sajjad, Alexander Fraser, and Helmut Schmid. Hindi-to-Urdu machine translation through transliteration. In *Proceedings of the 48th Annual meeting of the Association for Computational Linguistics*, pages 465–474. Association for Computational Linguistics, 2010.

Nadir Durrani, Hassan Sajjad, Hieu Hoang, and Philipp Koehn. Integrating an unsupervised transliteration model into statistical machine translation. In *EACL*, volume 14, pages 148–153, 2014.

Sašo Džeroski, Tomaž Erjavec, Nina Ledinek, Petr Pajas, Zdeněk Žabokrtský, and Andreja Žele. Towards a Slovene dependency treebank. In *Proc. of LREC 2006*, pages 1388–1391, Genova, Italy, 2006. European Language Resources Association (ELRA). URL `http://hnk.ffzg.hr/bibl/lrec2006/summaries/133.html`.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 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*, pages 644–648, Atlanta, Georgia, June 2013.

Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N13-1073.

Jack Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240, 1967.

Jason M. Eisner. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 340–345, Stroudsburg, PA, USA, 1996. ACL. doi: 10.3115/992628.992688. URL http://dx.doi.org/10.3115/992628.992688.

Cédrick Fairon, Sébastien Paumier, and Patrick Watrin. Can we parse without tagging? In *2nd Language & Technology Conference (LTC'05)*, pages 473–477, Poznan, Poland, 2005. 2nd Language & Technology Conference (LTC'05). URL https://hal-upec-upem.archives-ouvertes.fr/hal-00636997.

Philip Gage. A new algorithm for data compression. *The C Users Journal*, 12 (2):23–38, 1994.

Yoav Goldberg and Michael Elhadad. 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*, HLT '10, pages 742–750, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. ISBN 1-932432-65-5. URL http://dl.acm.org/citation.cfm?id=1857999.1858114.

Stephan Gouws and Anders Søgaard. Simple task-specific bilingual word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1386–1390, 2015.

Nathan David Green and Zdeněk Žabokrtský. Hybrid combination of constituency and dependency trees into an ensemble dependency parser. In *Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data*, HYBRID '12, pages 19–26, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2388632.2388635.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 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)*, volume 1, pages 1234–1244, 2015.

Jan Hajič, Otakar Smrž, Petr Zemánek, Petr Pajas, Jan Šnaidauf, Emanuel Beška, Jakub Kráčmar, and Kamila Hassanová. Prague Arabic dependency treebank 1.0. Linguistic Data Consortium, 2004a. ISBN 1-58563-319-4.

Jan Hajič, Otakar Smrž, Petr Zemánek, Jan Šnaidauf, and Emanuel Beška. Prague Arabic dependency treebank: Development in data and tools. In

Mahtab Nikkhou, editor, *Proceedings of the NEMLAR International Conference on Arabic Language Resources and Tools*, pages 110–117, Cairo, 2004b. ELDA.

Jan Hajič, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, Zdeněk Žabokrtský, and Magda Ševčíková-Razímová. Prague Dependency Treebank 2.0. CD-ROM, Linguistic Data Consortium, LDC Catalog No.: LDC2006T01, Philadelphia, 2006. URL `http://hdl.handle.net/11858/00-097C-0000-0001-B098-5`.

Jan Hajič, Silvie Cinková, Kristýna Čermáková, Lucie Mladová, Anja Nedolužko, Petr Pajas, Jiří Semecký, Jana Šindlerová, Josef Toman, Kristýna Tomšů, Matěj Korvas, Magdaléna Rysová, Kateřina Veselovská, and Zdeněk Žabokrtský. Prague English dependency treebank 1.0, 2009.

Jan Hajič. Building a syntactically annotated corpus: The Prague Dependency Treebank. In Eva Hajičová, editor, *Issues of Valency and Meaning. Studies in Honor of Jarmila Panevová*, pages 12–19. Prague Karolinum, Charles University Press, 1998.

Péter Halácsy, András Kornai, and Csaba Oravecz. HunPos: an open source trigram tagger. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 209–212. Association for Computational Linguistics, 2007.

Chung-hye Han, Na-Rae Han, Eon-Suk Ko, Martha Palmer, and Heejong Yi. Penn Korean treebank: Development and evaluation. In *Proceedings of the 16th Pacific Asia Conference on Language, Information and Computation*, pages 69–78, 2002.

Christian Hardmeier, Sara Stymne, Jörg Tiedemann, Aaron Smith, and Joakim Nivre. Anaphora models and reordering for phrase-based SMT. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 122–129, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W14-3312`.

Katri Haverinen, Timo Viljanen, Veronika Laippala, Samuel Kohonen, Filip Ginter, and Tapio Salakoski. Treebanking Finnish. In Markus Dickinson, Kaili Müürisep, and Marco Passarotti, editors, *Proc. of the Ninth International Workshop on Treebanks and Linguistic Theories (TLT9)*, pages 79–90, 2010. URL `http://hdl.handle.net/10062/15936`.

Kenneth Heafield. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197. Association for Computational Linguistics, 2011.

Samar Husain, Prashanth Mannem, Bharat Ambati, and Phani Gadde. The ICON-2010 tools contest on Indian language dependency parsing. In *Proc. of ICON-2010 Tools Contest on Indian Language Dependency Parsing*, Kharagpur, India, 2010.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11:11–311, 2005.

Nancy Ide and Jean Véronis. Multext: Multilingual text tools and corpora. In *Proceedings of the 15th Conference on Computational Linguistics - Volume 1*, COLING '94, pages 588–592, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics. doi: 10.3115/991886.991990. URL `https://doi.org/10.3115/991886.991990`.

Angelina Ivanova, Stephan Oepen, and Lilja Øvrelid. Survey on parsing three dependency representations for English. In *ACL (Student Research Workshop)*, pages 31–37, 2013.

Dan Jurafsky and James H Martin. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2 edition, 2009.

Dan Jurafsky and James H Martin. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition. Draft of the 3rd edition, Aug 2017. URL `https://web.stanford.edu/~jurafsky/slp3/`.

Yasuhiro Kawata and Julia Bartels. Stylebook for the Japanese treebank in Verbmobil. In *Report 240*, Tübingen, Germany, 2000.

Dan Klein and Christopher D Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 478. Association for Computational Linguistics, 2004.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume, Proceedings of the Student Research Workshop, Proceedings of Demo and Poster Sessions, Tutorial Abstracts*, pages 177–180, Praha, Czechia, 2007. Univerzita Karlova v Praze, Association for Computational Linguistics. ISBN 978-1-932432-87-9.

Matthias T. Kromann, Line Mikkelsen, and Stine Kern Lynge. Danish dependency treebank, 2004. URL `http://code.google.com/p/copenhagen-dependency-treebank/`.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, pages 79–86, 1951.

Lillian Lee. On the effectiveness of the skew divergence for statistical language analysis. In *Artificial intelligence and statistics*, volume 2001, pages 65–72, 2001.

Geoffrey Leech and Andrew Wilson. EAGLES recommendations for the morphosyntactic annotation of corpora. Technical Report EAG–TCWG–MAC/R, Expert Advisory Group on Language Engineering Standards, March 1996.

Uri Lerner and Slav Petrov. Source-side classifier preordering for machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 513–523, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D13-1049`.

Pierre Lison and Jörg Tiedemann. Opensubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association, 2016.

Marco Lui and Timothy Baldwin. langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations*, pages 25–30. Association for Computational Linguistics, 2012.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. The Penn Arabic treebank: Building a large-scale annotated Arabic corpus. In *NEMLAR conference on Arabic language resources and tools*, volume 27, pages 466–467. Cairo, 2004.

Martin Majliš and Zdeněk Žabokrtský. Language richness of the web. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pages 2927–2934, İstanbul, Turkey, 2012. European Language Resources Association. ISBN 978-2-9517408-7-7.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330, 1993.

David Mareček. Twelve years of unsupervised dependency parsing. In Broňa Brejová, editor, *Proceedings of the 16th ITAT: Slovenskočeský NLP workshop (SloNLP 2016)*, volume 1649 of *CEUR Workshop Proceedings*, pages 56–62, Bratislava, Slovakia, 2016a. Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics, CreateSpace Independent Publishing Platform. ISBN 978-1537016740.

David Mareček. Delexicalized and minimally supervised parsing on Universal Dependencies. In Pavel Král and Carlos Martín-Vide, editors, *Lecture Nites in Artificial Intelligence, Statistical Language and Speech Processing*, number 9918 in Lecture Notes in Computer Science, pages 30–42, Cham, Switzerland, 2016b. NTIS research centre at the Faculty of Applied Sciences of the University of West Bohemia, Springer International Publishing. ISBN 978-3-319-45924-0.

David Mareček and Zdeněk Žabokrtský. Exploiting reducibility in unsupervised dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 297–307, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. ISBN 978-1-937284-43-5.

David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Daniel Zeman, Zdeněk Žabokrtský, and Jan Hajič. HamleDT - HArmonized Multi-LanguagE Dependency Treebank, 2011.

David Mareček, Zhiwei Yu, Daniel Zeman, and Zdeněk Žabokrtský. Deltacorpus 1.1, 2016. URL `http://hdl.handle.net/11234/1-1743`. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

André FT Martins, Noah A Smith, Eric P Xing, Pedro MQ Aguiar, and Mário AT Figueiredo. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44. Association for Computational Linguistics, 2010.

Jan Mašek. Detection and correction of inconsistencies in the multilingual treebank HamleDT. Master's thesis, Charles University in Prague, Faculty of Mathematics and Physics, Praha, Czechia, 2015.

David McClosky, Eugene Charniak, and Mark Johnson. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 337–344, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. doi: 10.3115/1220175.1220218. URL `https://doi.org/10.3115/1220175.1220218`.

David McClosky, Eugene Charniak, and Mark Johnson. Automatic domain adaptation for parsing. In *Proceedings of HLT-NAACL*, pages 28–36. ACL, 2010.

Ryan McDonald and Fernando Pereira. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, volume 6, pages 81–88, 2006.

Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91–98. ACL, 2005a.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT-EMNLP*, pages 523–530. ACL, 2005b.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 216–220, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1596276.1596317`.

Ryan McDonald, Slav Petrov, and Keith Hall. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, 2011.

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, and Jungmee Lee. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 92–97, 2013.

Avihai Mejer and Koby Crammer. Are you sure? confidence in prediction of dependency tree edges. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 573–576, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. ISBN 978-1-937284-20-6. URL `http://dl.acm.org/citation.cfm?id=2382029.2382119`.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119, 2013.

Simonetta Montemagni, Francesco Barsotti, Marco Battista, Nicoletta Calzolari, Ornella Corazzari, Alessandro Lenci, Antonio Zampolli, Francesca Fanciulli, Maria Massetani, Remo Raffaelli, Roberto Basili, Maria Teresa Pazienza, Dario Saracino, Fabio Zanzotto, Nadia Mana, Fabio Pianesi, and Rodolfo Delmonte. Building the Italian syntactic-semantic treebank. In Anne Abeillé, editor, *Building and using Parsed Corpora*, Language and Speech series, pages 189–210, Dordrecht, 2003. Kluwer.

Tahira Naseem, Regina Barzilay, and Amir Globerson. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the ACL: Long Papers - Volume 1*, ACL '12, pages 629–637, Stroudsburg, PA, USA, 2012. ACL. URL `http://dl.acm.org/citation.cfm?id=2390524.2390613`.

Jens Nilsson, Johan Hall, and Joakim Nivre. MAMBA meets TIGER: Reconstructing a Swedish treebank from antiquity. In *Proc. of the NODALIDA Special Session on Treebanks*, 2005. URL `http://www.msi.vxu.se/users/nivre/research/Talbanken05.html`.

Joakim Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*, pages 149–160, 2003.

Joakim Nivre. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57. Association for Computational Linguistics, 2004.

Joakim Nivre. 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: Volume 1-Volume 1*, pages 351–359. Association for Computational Linguistics, 2009.

Joakim Nivre. Towards a universal grammar for natural language processing. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, pages 3–16, Cham, 2015. Springer International Publishing. ISBN 978-3-319-18111-0.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Joakim Nivre, Cristina Bosco, Jinho Choi, Marie-Catherine de Marneffe, Timothy Dozat, Richárd Farkas, Jennifer Foster, Filip Ginter, Yoav Goldberg, Jan Hajič, Jenna Kanerva, Veronika Laippala, Alessandro Lenci, Teresa Lynn, Christopher Manning, Ryan McDonald, Anna Missilä, Simonetta Montemagni, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Maria Simi, Aaron Smith, Reut Tsarfaty, Veronika Vincze, and Daniel Zeman. Universal Dependencies 1.0, 2015. URL `http://hdl.handle.net/11234/1-1464`. `http://hdl.handle.net/11234/1-1464`.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666, Portorož, Slovenia, 2016a.

Joakim Nivre et al. Universal dependencies 1.4, 2016b. URL `http://hdl.handle.net/11234/1-1827`. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre et al. Universal dependencies 2.1, 2017. URL `http://hdl.handle.net/11234/1-2515`. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

Robert Östling and Jörg Tiedemann. Efficient word alignment with Markov chain Monte Carlo. *The Prague Bulletin of Mathematical Linguistics*, 106(1): 125–146, 2016.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

Barbara Plank and Gertjan Van Noord. Effective measures of domain similarity for parsing. In *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies-Volume 1*, pages 1566–1576. ACL, 2011.

Prokopis Prokopidis, Elina Desipri, Maria Koutsombogera, Harris Papageorgiou, and Stelios Piperidis. Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160, 2005.

Loganathan Ramasamy and Zdeněk Žabokrtský. Prague dependency style treebank for Tamil. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pages 23–25, İstanbul, Turkey, 2012. European Language Resources Association. ISBN 978-2-9517408-7-7.

Loganathan Ramasamy, David Mareček, and Zdeněk Žabokrtský. Multilingual dependency parsing: Using machine translated texts instead of parallel corpora. *The Prague Bulletin of Mathematical Linguistics*, 102:93–104, 2014. ISSN 0032-6585.

Mohammad Sadegh Rasooli, Amirsaeid Moloodi, Manouchehr Kouhestani, and Behrouz Minaei-Bidgoli. A syntactic valency lexicon for Persian verbs: The first steps towards Persian dependency treebank. In *5th Language & Technology Conference (LTC): Human Language Technologies as a Challenge for Computer Science and Linguistics*, pages 227–231, Poznań, Poland, 2011.

Rudolf Rosa. Automatic post-editing of phrase-based machine translation outputs. Master's thesis, Charles University in Prague, Faculty of Mathematics and Physics, Praha, Czechia, 2013.

Rudolf Rosa. MSTperl parser (2015-05-19), 2015a. URL `http://hdl.handle.net/11234/1-1480`. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Rudolf Rosa. MSTperl delexicalized parser transfer scripts and configuration files, 2015b. URL `http://hdl.handle.net/11234/1-1485`. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Rudolf Rosa. Multi-source cross-lingual delexicalized parser transfer: Prague or Stanford? In Eva Hajičová and Joakim Nivre, editors, *Proceedings of the Third International Conference on Dependency Linguistics, Depling 2015*, Uppsala, Sweden, 2015c. Uppsala University, Uppsala University.

Rudolf Rosa. MonoTrans: Statistical machine translation from monolingual data. In Jaroslava Hlaváčová, editor, *Proceedings of the 17th conference ITAT 2017:*

Slovenskočeský NLP workshop (SloNLP 2017), volume 1885 of *CEUR Workshop Proceedings*, pages 201–208, Praha, Czechia, 2017. ÚFAL MFF UK, CreateSpace Independent Publishing Platform. ISBN 978-1974274741.

Rudolf Rosa and David Mareček. Dependency relations labeller for Czech. In Petr Sojka, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *Text, Speech and Dialogue: 15th International Conference, TSD 2012. Proceedings*, number 7499 in Lecture Notes in Computer Science, pages 256–263, Berlin / Heidelberg, 2012. Masarykova univerzita v Brně, Springer Verlag. ISBN 978-3-642-32789-6.

Rudolf Rosa and Zdeněk Žabokrtský. MSTParser model interpolation for multi-source delexicalized transfer. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 71–75, Stroudsburg, PA, USA, 2015a. Euskal Herriko Unibertsitatea, Association for Computational Linguistics. ISBN 978-1-941643-98-3.

Rudolf Rosa and Zdeněk Žabokrtský. KL_{cpos^3} – a language similarity measure for delexicalized parser transfer. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Short Papers*, Stroudsburg, PA, USA, 2015b. Association for Computational Linguistics.

Rudolf Rosa and Zdeněk Žabokrtský. Error analysis of cross-lingual tagging and parsing. In Jan Hajič, editor, *Proceedings of the 16th International Workshop on Treebanks and Linguistic Theories*, pages 106–118, Praha, Czechia, 2017. Univerzita Karlova, Univerzita Karlova. ISBN 978-80-88132-04-2.

Rudolf Rosa, Ondřej Dušek, David Mareček, and Martin Popel. Using parallel features in parsing of machine-translated sentences for correction of grammatical errors. In *Proceedings of Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-6), ACL*, pages 39–48, Jeju, Korea, 2012a. ACL. ISBN 978-1-937284-38-1.

Rudolf Rosa, David Mareček, and Ondřej Dušek. DEPFIX: A system for automatic correction of Czech MT outputs. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 362–368, Montréal, Canada, 2012b. Association for Computational Linguistics. ISBN 978-1-937284-20-6.

Rudolf Rosa, Jan Mašek, David Mareček, Martin Popel, Daniel Zeman, and Zdeněk Žabokrtský. HamleDT 2.0: Thirty dependency treebanks stanfordized. In *Proceedings of LREC 2014*, pages 2334–2341, Reykjavík, Iceland, 2014. ELRA. ISBN 978-2-9517408-8-4.

Rudolf Rosa, Daniel Zeman, David Mareček, and Zdeněk Žabokrtský. Slavic forest, Norwegian wood. In Preslav Nakov, Marcos Zampieri, Nikola Ljubešić, Jörg Tiedemann, Shervin Malmasi, and Ahmed Ali, editors, *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial4)*, pages 210–219, Stroudsburg, PA, USA, 2017. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 978-1-945626-43-2.

Kenji Sagae and Alon Lavie. Parser combination by reparsing. In *Proceedings of HLT-NAACL*, pages 129–132. ACL, 2006.

Sebastian Schuster and Christopher D Manning. Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In *LREC*, 2016.

Roy Schwartz, Omri Abend, and Ari Rappoport. Learnability-based syntactic annotation design. In *Proceedings of COLING 2012: Technical Papers*, 2012.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1715–1725, 2016.

Petr Sgall. Functional sentence perspective in a generative description. *Prague studies in mathematical linguistics*, 2(203-225), 1967.

Kiril Simov and Petya Osenova. Extending the annotation of BulTreeBank: Phase 2. In *The Fourth Workshop on Treebanks and Linguistic Theories (TLT 2005)*, pages 173–184, Barcelona, 2005.

Otakar Smrž, Viktor Bielický, Iveta Kouřilová, Jakub Kráčmar, Jan Hajič, and Petr Zemánek. Prague Arabic dependency treebank: A word on the million words. In *Proc. of the Workshop on Arabic and Local Languages (LREC 2008)*, pages 16–23, Marrakech, Morocco, 2008. European Language Resources Association. ISBN 2-9517408-4-0.

Anders Søgaard. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies: short papers-Volume 2*, pages 682–686. ACL, 2011.

Anders Søgaard. An empirical study of differences between conversion schemes and annotation guidelines. In *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013), Prague, Czech Republic: Charles University in Prague, Matfyzpress*, pages 298–307, 2013.

Anders Søgaard and Julie Wulff. An empirical etudy of non-lexical extensions to delexicalized transfer. In *COLING (Posters)*, pages 1181–1190, 2012.

Anders Søgaard, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. Inverted indexing for cross-lingual NLP. In *The 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2015)*, 2015.

Milan Straka, Jan Hajič, Jana Straková, and Jan Hajič jr. Parsing universal dependency treebanks using neural networks and search-based oracle. In *Proceedings of Fourteenth International Workshop on Treebanks and Linguistic Theories (TLT 14)*, December 2015.

Milan Straka, Jan Hajič, and Jana Straková. UDPipe: trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, Paris, France, May 2016. European Language Resources Association (ELRA). ISBN 978-2-9517408-9-1.

Jana Straková, Milan Straka, and Jan Hajič. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P14/P14-5003.pdf`.

Katsuhito Sudoh, Xianchao Wu, Kevin Duh, Hajime Tsukada, and Masaaki Nagata. Post-ordering in statistical machine translation. In *Proc. MT Summit*, 2011.

Mihai Surdeanu and Christopher D. Manning. Ensemble models for dependency parsing: Cheap and good? In *Proceedings of HLT-NAACL*, HLT '10, pages 649–652, Stroudsburg, PA, USA, 2010. ACL. ISBN 1-932432-65-5. URL `http://dl.acm.org/citation.cfm?id=1857999.1858090`.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proc. of CoNLL*, 2008.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the ACL: Human Language Technologies*, NAACL HLT '12, pages 477–487, Stroudsburg, PA, USA, 2012. ACL, Association for Computational Linguistics. ISBN 978-1-937284-20-6. URL `http://dl.acm.org/citation.cfm?id=2382029.2382096`.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12, 2013a.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. Target language adaptation of discriminative transfer parsers. In *Proceedings of NAACL-HLT 2013*, pages 1061–1071, 2013b.

Mariona Taulé, Maria Antònia Martí, and Marta Recasens. AnCora: Multilevel annotated corpora for Catalan and Spanish. In *LREC*, 2008.

Lucien Tesnière. *Eléments de syntaxe structurale*. Éditions Klincksieck, Paris, 1959.

Lucien Tesnière. *Elements of structural syntax, translated by Timothy Osborne and Sylvain Kahane*. Benjamins, Amsterdam/Philadelphia, 2015.

Jörg Tiedemann. Parallel data, tools and interfaces in OPUS. In *LREC*, volume 2012, pages 2214–2218, 2012.

Jörg Tiedemann. Rediscovering annotation projection for cross-lingual parser induction. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1854–1864, August 2014.

Jörg Tiedemann. Improving the cross-lingual projection of syntactic dependencies. In *Proceedings of the 20th Nordic Conference of Computational Linguistics, NODALIDA 2015, May 11-13, 2015, Vilnius, Lithuania*, number 109, pages 191–199. Linköping University Electronic Press, 2015.

Jörg Tiedemann. Cross-lingual dependency parsing for closely related languages – Helsinki's submission to VarDial 2017. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Valencia, Spain, 2017.

Jörg Tiedemann and Željko Agić. Synthetic treebanking for cross-lingual dependency parsing. *Journal of Artificial Intelligence Research*, 2016.

Jörg Tiedemann, Željko Agić, and Joakim Nivre. Treebank translation for cross-lingual parser induction. In *Eighteenth Conference on Computational Natural Language Learning (CoNLL 2014)*, 2014.

Vincent Van Asch and Walter Daelemans. Using domain similarity for performance estimation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 31–36. ACL, 2010.

Leonoor van der Beek, Gosse Bouma, Jan Daciuk, Tanja Gaustad, Robert Malouf, Gertjan van Noord, Robbert Prins, and Begoña Villada. Chapter 5. the Alpino dependency treebank. In *Algorithms for Linguistic Processing NWO PIONIER Progress Report*, Groningen, The Netherlands, 2002. URL `http://odur.let.rug.nl/~vannoord/trees/Papers/report_ch5.pdf`.

Dániel Varga, Péter Halácsy, András Kornai, Viktor Nagy, László Németh, and Viktor Trón. Parallel corpora for medium density languages. *Amsterdam studies in the theory and history of linguistic science series 4*, 292:247, 2007.

Mária Šimková and Radovan Garabík. Sintaksičeskaja razmetka v Slovackom nacional'nom korpuse. In *Trudy meždunarodnoj konferencii Korpusnaja lingvistika – 2006*, pages 389–394, Sankt-Peterburg, Russia, 2006. St. Petersburg University Press. ISBN 5-288-04181-4.

William E. Winkler. String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. In *Proceedings of the Section on Survey Research Methods (American Statistical Association)*, pages 354–359, 1990. URL `http://www.amstat.org/sections/srms/Proceedings/papers/1990_056.pdf`.

Fei Xia and Michael McCord. Improving a statistical MT system with automatically learned rewrite patterns. In *Proceedings of the 20th international conference on Computational Linguistics*, page 508. Association for Computational Linguistics, 2004.

Fei Xia and Martha Palmer. Converting dependency structures to phrase structures. In *Proceedings of the First International Conference on Human Language Technology Research*, HLT '01, pages 1–5, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics. doi: 10.3115/1072133.1072147. URL https://doi.org/10.3115/1072133.1072147.

Min Xiao and Yuhong Guo. Distributed word representation learning for cross-lingual dependency parsing. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 119–129, 2014.

Deyi Xiong, Qun Liu, and Shouxun Lin. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 521–528, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. doi: 10.3115/1220175.1220241. URL http://dx.doi.org/10.3115/1220175.1220241.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. The Penn Chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207–238, 2005.

David Yarowsky, Grace Ngai, and Richard Wicentowski. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the First International Conference on Human Language Technology Research*, HLT '01, pages 1–8, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics. doi: 10.3115/1072133.1072187. URL https://doi.org/10.3115/1072133.1072187.

Zhiwei Yu, David Mareček, Zdeněk Žabokrtský, and Daniel Zeman. If you even don't have a bit of Bible: Learning delexicalized POS taggers. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 96–103, Paris, France, 2016. European Language Resources Association. ISBN 978-2-9517408-9-1.

Marcos Zampieri, Shervin Malmasi, Nikola Ljubešić, Preslav Nakov, Ahmed Ali, Jörg Tiedemann, Yves Scherrer, and Noëmi Aepli. Findings of the VarDial evaluation campaign 2017. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Valencia, Spain, 2017.

Daniel Zeman. Reusable tagset conversion using tagset drivers. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*, pages 213–218, Marrakech, Morocco, 2008. European Language Resources Association. ISBN 2-9517408-4-0.

Daniel Zeman. Hard problems of tagset conversion. In Alex Fang, Nancy Ide, and Jonathan Webster, editors, *Proceedings of the Second International Conference on Global Interoperability for Language Resources*, pages 181–185, Hong Kong, China, 2010. City University of Hong Kong, City University of Hong Kong. ISBN 978-962-442-323-5.

Daniel Zeman and Philip Resnik. Cross-language parser adaptation between related languages. In *IJCNLP 2008 Workshop on NLP for Less Privileged Languages*, pages 35–42, Hyderabad, India, 2008. Asian Federation of Natural Language Processing, International Institute of Information Technology.

Daniel Zeman and Zdeněk Žabokrtský. Improving parsing accuracy by combining diverse dependency parsers. In *Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT)*, pages 171–178, Vancouver, BC, Canada, 2005. Simon Fraser University, ACL. ISBN 1-932432-58-2.

Daniel Zeman, David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Zdeněk Žabokrtský, and Jan Hajič. HamleDT: To parse or not to parse? In *Proceedings of LREC'12*, Istanbul, Turkey, May 2012. ELRA. ISBN 978-2-9517408-7-7. URL `http://www.lrec-conf.org/proceedings/lrec2012/pdf/429_Paper.pdf`.

Daniel Zeman, David Mareček, Jan Mašek, Martin Popel, Loganathan Ramasamy, Rudolf Rosa, Jan Štěpánek, and Zdeněk Žabokrtský. HamleDT 2.0, 2014. URL `http://hdl.handle.net/11858/00-097C-0000-0023-9551-4`. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gökırmak, Anna Nedoluzhko, Silvie Cinková, jr. Jan Hajič, Jaroslava Hlaváčová, Václava Kettnerová, Zdeňka Urešová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, 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 Mendonça, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 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*, pages 1–19, Stroudsburg, PA, USA, 2017. Charles University, Association for Computational Linguistics. ISBN 978-1-945626-70-8. URL `http://www.aclweb.org/anthology/K/K17/K17-3001.pdf`.

Yuan Zhang and Regina Barzilay. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1857–1867, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL `http://aclweb.org/anthology/D15-1213`.

Yue Zhang and Stephen Clark. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing using beam-search.

In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 562–571, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1613715.1613784.

List of Figures

List of Tables

List of Abbreviations

BPE	Byte Pair Encoding
EBC	Edinburgh Bible Corpus
GSD	Google Stanford Dependencies
LAS	Labelled Attachment Score
MGA	Monolingual Greedy Aligner
MIRA	Margin-infused Relaxed Algorithm
MST	Maximum Spanning Tree
MT	Machine Translation
NLP	Natural Language Processing
PBMT	Phrase-Based Machine Translation
PDT	Prague Dependency Treebank
PennTB	Penn Treebank
POS	Part of Speech
SD	Stanford Dependencies
SMT	Statistical Machine Translation
UAS	Unlabelled Attachment Score
UD	Universal Dependencies
UDHR	Universal Declaration of Human Rights
UPOS	Universal Part of Speech
UPT	Universal Part of Speech Tagset
USD	Universal Stanford Dependencies
WALS	World Atlas of Language Structures
WTC	Watchtower Corpus

List of Publications

Joakim Nivre et al. Universal dependencies 1.4, 2016. URL `http://hdl.handle.net/11234/1-1827`. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Rudolf Rosa. Automatic post-editing of phrase-based machine translation outputs. Master's thesis, Charles University in Prague, Faculty of Mathematics and Physics, Praha, Czechia, 2013.

Rudolf Rosa. MSTperl parser (2015-05-19), 2015a. URL `http://hdl.handle.net/11234/1-1480`. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Rudolf Rosa. MSTperl delexicalized parser transfer scripts and configuration files, 2015b. URL `http://hdl.handle.net/11234/1-1485`. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Rudolf Rosa. Multi-source cross-lingual delexicalized parser transfer: Prague or Stanford? In Eva Hajičová and Joakim Nivre, editors, *Proceedings of the Third International Conference on Dependency Linguistics, Depling 2015*, Uppsala, Sweden, 2015c. Uppsala University, Uppsala University.

Rudolf Rosa. MonoTrans: Statistical machine translation from monolingual data. In Jaroslava Hlaváčová, editor, *Proceedings of the 17th conference ITAT 2017: Slovenskočeský NLP workshop (SloNLP 2017)*, volume 1885 of *CEUR Workshop Proceedings*, pages 201–208, Praha, Czechia, 2017a. ÚFAL MFF UK, CreateSpace Independent Publishing Platform. ISBN 978-1974274741.

Rudolf Rosa. Terminal-based CoNLL-file viewer, v2, 2017b. URL `http://hdl.handle.net/11234/1-2514`. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Rudolf Rosa and David Mareček. Dependency relations labeller for Czech. In Petr Sojka, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *Text, Speech and Dialogue: 15th International Conference, TSD 2012. Proceedings*, number 7499 in Lecture Notes in Computer Science, pages 256–263, Berlin / Heidelberg, 2012. Masarykova univerzita v Brně, Springer Verlag. ISBN 978-3-642-32789-6.

Rudolf Rosa and Zdeněk Žabokrtský. MSTParser model interpolation for multi-source delexicalized transfer. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 71–75, Stroudsburg, PA, USA, 2015a. Euskal Herriko Unibertsitatea, Association for Computational Linguistics. ISBN 978-1-941643-98-3.

Rudolf Rosa and Zdeněk Žabokrtský. KL_{cpos^3} – a language similarity measure for delexicalized parser transfer. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Short Papers*, Stroudsburg, PA, USA, 2015b. Association for Computational Linguistics.

Rudolf Rosa and Zdeněk Žabokrtský. Error analysis of cross-lingual tagging and parsing. In Jan Hajič, editor, *Proceedings of the 16th International Workshop on Treebanks and Linguistic Theories*, pages 106–118, Praha, Czechia, 2017. Univerzita Karlova, Univerzita Karlova. ISBN 978-80-88132-04-2.

Rudolf Rosa, Ondřej Dušek, David Mareček, and Martin Popel. Using parallel features in parsing of machine-translated sentences for correction of grammatical errors. In *Proceedings of Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-6), ACL*, pages 39–48, Jeju, Korea, 2012. ACL. ISBN 978-1-937284-38-1.

Rudolf Rosa, Jan Mašek, David Mareček, Martin Popel, Daniel Zeman, and Zdeněk Žabokrtský. HamleDT 2.0: Thirty dependency treebanks stanfordized. In *Proceedings of LREC 2014*, pages 2334–2341, Reykjavík, Iceland, 2014. ELRA. ISBN 978-2-9517408-8-4.

Rudolf Rosa, Daniel Zeman, David Mareček, and Zdeněk Žabokrtský. Slavic forest, Norwegian wood (scripts), 2017a. URL `http://hdl.handle.net/11234/1-1970`. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Rudolf Rosa, Daniel Zeman, David Mareček, and Zdeněk Žabokrtský. Slavic forest, Norwegian wood. In Preslav Nakov, Marcos Zampieri, Nikola Ljubešić, Jörg Tiedemann, Shervin Malmasi, and Ahmed Ali, editors, *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial4)*, pages 210–219, Stroudsburg, PA, USA, 2017b. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 978-1-945626-43-2.

Daniel Zeman, David Mareček, Jan Mašek, Martin Popel, Loganathan Ramasamy, Rudolf Rosa, Jan Štěpánek, and Zdeněk Žabokrtský. HamleDT 2.0, 2014. URL `http://hdl.handle.net/11858/00-097C-0000-0023-9551-4`. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Attachments

A Universal relation labels v1

List of the 40 universal dependency relation labels used in Universal Dependencies v1, reproduced from the official website:[9]

acl	clausal modifier of noun (adjectival clause)
advcl	adverbial clause modifier
advmod	adverbial modifier
amod	adjectival modifier
appos	appositional modifier
aux	auxiliary
auxpass	passive auxiliary
case	case marking
cc	coordinating conjunction
ccomp	clausal complement
compound	compound
conj	conjunct
cop	copula
csubj	clausal subject
csubjpass	clausal passive subject
dep	unspecified dependency
det	determiner
discourse	discourse element
dislocated	dislocated elements
dobj	direct object
expl	expletive
foreign	foreign words
goeswith	goes with
iobj	indirect object
list	list
mark	marker
mwe	multi-word expression
name	name
neg	negation modifier
nmod	nominal modifier
nsubj	nominal subject
nsubjpass	passive nominal subject
nummod	numeric modifier
parataxis	parataxis
punct	punctuation
remnant	remnant in ellipsis
reparandum	overridden disfluency
root	root
vocative	vocative
xcomp	open clausal complement

[9] `http://universaldependencies.org/docsv1/u/dep/index.html`

B List of Watchtower languages

B.1 Watchtower corpus

The WTC of Agić et al. [2016] contains texts in 135 languages. For each language, we report its ISO code as used in the dataset, its name according to WALS or Wikipedia, and the number of sentences, i.e. lines in the text file for that language in WTC.

af Afrikaans (144,194)
am Amharic (142,754)
ar Arabic (136,565)
ay Aymara (61,913)
az Azerbaijani (93,326)
bci Baule (78,066)
bcl Central Bikol (116,068)
bem Bemba (142,863)
bg Bulgarian (129,709)
bi Bislama (126,420)
bn Bengali (114,477)
cat Catalan (28,293)
ceb Cebuano (139,075)
cs Czech (153,642)
da Danish (146,251)
de German (159,244)
ee Ewe (141,134)
efi Efik (134,668)
el Greek (131,653)
en English (149,604)
es Spanish (144,939)
et Estonian (143,944)
fa Persian (84,966)
fi Finnish (138,408)
fj Fijian (130,713)
fr French (154,479)
gaa Gã (130,768)
gil Kiribati (82,170)
gug Guaraní (56,028)
gu Gujarati (123,994)
guw Gungbe (126,776)
he Hebrew (127,336)
hil Hiligaynon (137,922)
hi Hindi (124,822)
ho Hiri Motu (109,805)
hr Croatian (151,108)
ht Haitian (86,950)
hu Hungarian (152,073)
hy Armenian (153,674)
id Indonesian (150,193)
ig Igbo (138,487)
ilo Ilocano (137,980)
is Icelandic (98,489)
it Italian (148,543)
ja Japanese (44,017)
ka Georgian (138,997)
khk Khalkha (76,750)
ki Kikuyu (27,447)
kk Kazakh (83,946)
kl Kalaallisut (82,766)
km Khmer (40,666)
kn Kannada (127,783)
ko Korean (150,090)
kqn Kaonde (87,178)
kwy Iwoyo (64,779)
ky Kirghiz (86,264)
lg Ganda (95,453)
ln Lingala (144,733)
loz Lozi (120,701)
lt Lithuanian (145,741)
lue Luvale (105,597)
lun Lunda (50,034)
luo Luo (53,239)
lv Latvian (126,340)
mg Malagasy (146,349)
mk Macedonian (138,298)
ml Malayalam (129,985)
mos Mossi (83,629)
mr Marathi (117,743)
mt Maltese (122,785)
my Burmese (46,389)
ne Nepali (114,714)
nl Dutch (144,533)
no Norwegian (149,099)
nso Northern Sotho (144,428)
nyk Nyaneka (43,337)
ny Chichewa (148,638)
om Oromo (65,177)
os Ossetian (103,918)
pag Pangasinan (116,125)

pap Papiamentu (118,750)
pa Panjabi (123,884)
pis Solomons Pijin (118,876)
pl Polish (146,432)
pon Pohnpeian (82,001)
pt Portuguese (146,554)
qu Quechua (64,741)
quy Ayacucho Quechua (52,686)
quz Cuzco Quechua (51,584)
ro Romanian (153,409)
run Rundi (132,091)
ru Russian (126,196)
rw Kinyarwanda (140,668)
sg Sango (118,253)
si Sinhala (166,468)
sk Slovak (145,333)
sl Slovenian (143,777)
sm Samoan (136,201)
sn Shona (143,197)
sq Albanian (150,830)
srn Sranan (152,377)
st Southern Sotho (141,252)
sv Swedish (147,864)
swc Swahili (29,723)
sw Swahili (144,896)
ta Tamil (134,742)
te Telugu (125,027)
th Thai (125,410)
ti Tigrinya (146,282)
tl Tagalog (140,432)
tn Tswana (140,873)
toi Tonga/Zambezi (91,078)
to Tonga (119,007)
tpi Tok Pisin (151,673)
tr Turkish (161,164)
ts Tsonga (141,690)
tt Tatar (81,896)
tum Tumbuka (95,362)
tvl Tuvaluan (74,376)
tw Twi (138,146)
ty Tahitian (93,513)
tzo Tzotzil (56,900)
uk Ukrainian (132,368)
umb Umbundu (81,928)
ur Urdu (111,508)
uz Uzbek (59,741)
ve Venda (64,441)
vi Vietnamese (149,592)
war Waray-Waray (114,875)
wls Wallisian (101,211)
xh Xhosa (135,372)
yo Yoruba (142,807)
yua Yucatec (76,130)
zai Isthmus Zapotec (68,070)
zu Zulu (134,792)

B.2 Watchtower online

Based on the language selection box on the Watchtower Online website,[10] it seems that the Watchtower texts are available in the following 301 languages as of 28th March 2018:

1. Acholi
2. Afrikaans
3. Ahanta
4. Aja
5. Albanian
6. Albanian Sign Language
7. Amharic
8. Arabic
9. Arabic (Jordan)
10. Arabic (Lebanon)
11. Argentinean Sign Language
12. Armenian
13. Armenian (West)
14. Attié
15. Aukan
16. Australian Sign Language
17. Aymara
18. Azerbaijani
19. Azerbaijani (Cyrillic)
20. Bambara
21. Baoule
22. Bariba
23. Bashkir
24. Basque
25. Bassa (Cameroon)
26. Bassa (Liberia)
27. Batak (Toba)
28. Bengali
29. Betsimisaraka (Northern)
30. Betsimisaraka (Southern)
31. Bicol
32. Bidayuh (Bukar)
33. Bislama
34. Bissau Guinean Creole
35. Blackfoot
36. Bomu
37. Bosnian
38. British Sign Language
39. Bulgarian
40. Cambodian
41. Catalan
42. Cebuano
43. Changana (Zimbabwe)
44. Chichewa
45. Chilean Sign Language
46. Chin (Falam)
47. Chin (Hakha)
48. Chinese Mandarin (Simplified)
49. Chinese Mandarin (Traditional)
50. Chitonga (Malawi)
51. Chitonga (Zimbabwe)
52. Chitumbuka
53. Chiyao
54. Chokwe
55. Cinamwanga
56. Croatian
57. Croatian Sign Language
58. Czech
59. Dagaare
60. Dangme
61. Danish
62. Dari
63. Douala
64. Drehu
65. Dusun
66. Dutch
67. Dutch Sign Language
68. English
69. Esan
70. Estonian
71. Estonian Sign Language
72. Ethiopian Sign Language
73. Ewe
74. Fante
75. Faroese
76. Fataluku
77. Fijian
78. Finnish
79. Finnish Sign Language
80. Fon
81. Frafra
82. French

[10] `https://wol.jw.org/`

83. French Sign Language
84. Frisian
85. Fulfulde (Cameroon)
86. Ga
87. Galician
88. Garifuna
89. Georgian
90. German
91. Greek
92. Greenlandic
93. Guarani
94. Guerze
95. Gun
96. Guéré
97. Haitian Creole
98. Haya
99. Hebrew
100. Hiligaynon
101. Hindi
102. Hmong (White)
103. Hungarian
104. Iban
105. Ibinda
106. Icelandic
107. Igbo
108. Iloko
109. Indonesian
110. Irish
111. Irish Sign Language
112. Isoko
113. Italian
114. Italian Sign Language
115. Japanese
116. Japanese Sign Language
117. Javanese
118. Jula
119. Kabiye
120. Kachin
121. Kadazan
122. Kalanga (Zimbabwe)
123. Karen (S'gaw)
124. Kazakh
125. Kikaonde
126. Kikongo
127. Kikuyu
128. Kimbundu
129. Kinyarwanda
130. Kirghiz
131. Kisi
132. Konkani (Devanagari)
133. Konkani (Roman)
134. Korean
135. Kpelle
136. Krio
137. Kurdish Kurmanji
138. Kurdish Kurmanji (Caucasus)
139. Kurdish Kurmanji (Cyrillic)
140. Kwanyama
141. Kyangonde
142. Latgalian
143. Latvian
144. Latvian Sign Language
145. Lingala
146. Lithuanian
147. Logo
148. Loma
149. Low German
150. Luganda
151. Macedonian
152. Madura
153. Mahorian (Roman)
154. Maithili
155. Makasae
156. Malagasy
157. Malay
158. Malayalam
159. Malaysian Sign Language
160. Mam
161. Mambae (Ermera)
162. Mandinka
163. Mandjak
164. Mangbetu
165. Maninkakan (Eastern)
166. Mano
167. Mapudungun
168. Marathi
169. Mauritian Creole
170. Mazahua
171. Mende
172. Meru
173. Mingrelian
174. Mixtec (Huajuapan)
175. Mizo
176. Mongolian
177. Moore
178. Myanmar

179. Nahuatl (Huasteca)
180. Nambya
181. Ndebele (Zimbabwe)
182. Ndonga
183. Nepali
184. Nepali Sign Language
185. Newari
186. Ngangela
187. Nias
188. Nigerian Pidgin
189. Niuean
190. Nivaclé
191. Norwegian
192. Nyaneka
193. Nzema
194. Obolo
195. Oromo
196. Otomi (Mezquital Valley)
197. Pangasinan
198. Papel
199. Papiamento (Aruba)
200. Papiamento (Curaçao)
201. Pehuenche
202. Pemon
203. Pennsylvania German
204. Persian
205. Piaroa
206. Pidgin (Cameroon)
207. Pilagá
208. Polish
209. Pomeranian
210. Portuguese
211. Portuguese (Portugal)
212. Pular
213. Punjabi
214. Punjabi (Shahmukhi)
215. Quebec Sign Language
216. Quechua (Ayacucho)
217. Quechua (Cuzco)
218. Quechua (Huaylla Wanca)
219. Quichua (Chibuleo)
220. Quichua (Chimborazo)
221. Quichua (Imbabura)
222. Quichua (Pastaza)
223. Quichua (Santiago del Estero)
224. Quichua (Tena)
225. Rapa Nui
226. Rarotongan
227. Romanian
228. Romany (Albania)
229. Romany (Argentina)
230. Romany (Eastern Slovakia)
231. Rungus
232. Runyankore
233. Russian
234. Rutoro
235. Saramaccan
236. Sarnami
237. Scottish Gaelic
238. Sehwi
239. Senoufo (Cebaara)
240. Sepedi
241. Serbian
242. Serbian (Roman)
243. Serer
244. Setswana
245. Seychelles Creole
246. Shona
247. Shuar
248. Shughni
249. Sidama
250. Sindhi
251. Sinhala
252. Slovak
253. Slovenian
254. Somali
255. Spanish
256. Sranantongo
257. Susu
258. Svan (Lower)
259. Svan (Upper)
260. Swahili
261. Swati
262. Swedish
263. Tagalog
264. Tahitian
265. Tajiki
266. Tandroy
267. Tankarana
268. Tarascan
269. Tatar
270. Telugu
271. Tetun Dili
272. Thai
273. Thai Sign Language
274. Ticuna

275. Tigrinya
276. Tiv
277. Toba
278. Tojolabal
279. Tok Pisin
280. Tongan
281. Toupouri
282. Turkish
283. Ukrainian
284. Umbundu
285. Urdu
286. Urhobo
287. Uruguayan Sign Language
288. Uzbek
289. Uzbek (Roman)
290. Valencian
291. Venezuelan Sign Language
292. Vezo
293. Vietnamese
294. Voru
295. Wallisian
296. Welsh
297. Wichi
298. Wolof
299. Xhosa
300. Yoruba
301. Yukpa

C Source-target language similarities

This attachment lists the $KL^{-4}_{cpos^3}$ source-target similarities for all source and target languages from the UD 1.4 dataset, computed on POS tags cross-lingually projected on multi-parallel Watchtower data.

See Table C.1, Table C.2 and Table C.3. The oracle source language for the delexicalized parser transfer is marked in bold; note that for different setups, the oracle source language may be different. Also note that this does not say anything about how appropriate the other sources are. We also mark the first 5 and first 7 most similar source languages, as these are used in the lexicalized parser and tagger combination, respectively.

Target	da		el		hu		id	
1.	**sv**	44.21	en	11.16	ru	3.58	ro	7.06
2.	no	40.27	sv	6.84	en	3.01	no	3.00
3.	en	23.81	no	4.94	**sv**	2.87	ru	2.80
4.	nl	4.89	ro	3.79	fi	2.62	sv	2.71
5.	sl	2.45	nl	2.74	cs	2.34	cs	2.57
6.	ro	2.34	sl	2.15	nl	2.25	en	2.04
7.	bg	1.25	ru	1.60	sl	1.98	he	2.03
8.	cs	1.25	**bg**	1.41	no	1.28	sl	1.79
9.	pt	1.12	cs	1.40	ro	0.96	hr	1.17
10.	ru	1.12	pt	1.19	he	0.73	fi	0.92
11.	it	1.08	he	1.04	bg	0.66	bg	0.89
12.	es	1.06	it	0.94	de	0.63	nl	0.83
13.	he	0.77	es	0.86	pt	0.38	**pt**	0.55
14.	de	0.72	de	0.58	et	0.33	it	0.43
15.	fr	0.59	fr	0.53	it	0.30	et	0.41
16.	ca	0.49	ca	0.46	hr	0.29	es	0.36
17.	fi	0.37	fi	0.28	es	0.26	fr	0.24
18.	hr	0.25	hr	0.20	fr	0.15	de	0.22
19.	et	0.10	fa	0.10	ca	0.14	ca	0.21
20.	fa	0.07	et	0.06	fa	0.13	fa	0.12
21.	hi	0.04	hi	0.05	hi	0.05	ar	0.05
22.	ar	0.02	ar	0.03	ar	0.02	hi	0.04

Table C.1: $KL^{-4}_{cpos^3}$ source-target similarities, part 1.

Target	ja		kk		lv		pl	
1.	nl	0.59	ru	1.27	en	3.94	ru	11.08
2.	en	0.43	en	1.12	sv	3.02	**cs**	10.43
3.	cs	0.37	fi	1.02	cs	2.39	sv	6.71
4.	no	0.30	sv	1.01	sl	2.30	sl	6.33
5.	sv	0.28	ro	0.96	ru	2.30	en	5.06
6.	bg	0.26	nl	0.87	**fi**	2.25	ro	3.58
7.	**fi**	0.26	cs	0.83	no	2.14	no	3.32
8.	sl	0.25	no	0.81	nl	2.08	nl	2.68
9.	ro	0.23	sl	0.67	ro	1.37	fi	2.34
10.	pt	0.18	he	0.56	bg	0.86	bg	2.23
11.	ru	0.14	hr	0.44	et	0.68	he	1.58
12.	hr	0.13	et	0.36	he	0.62	hr	0.94
13.	et	0.13	bg	0.35	hr	0.61	pt	0.89
14.	he	0.11	pt	0.26	de	0.43	de	0.69
15.	it	0.10	de	0.22	pt	0.40	it	0.67
16.	de	0.09	it	0.18	it	0.38	et	0.58
17.	es	0.06	fa	0.18	es	0.35	es	0.54
18.	fr	0.06	es	0.14	fr	0.18	fr	0.28
19.	fa	0.04	fr	0.10	ca	0.17	ca	0.22
20.	ca	0.03	ca	0.08	fa	0.10	fa	0.14
21.	ar	0.03	**hi**	0.07	hi	0.04	hi	0.05
22.	hi	0.02	ar	0.05	ar	0.03	ar	0.04

Table C.2: $KL^{-4}_{cpos^3}$ source-target similarities, part 2.

Target	sk		ta		tr		uk		vi	
1.	**cs**	15.38	fi	2.50	**fi**	4.25	**ru**	8.04	ro	1.40
2.	ru	9.99	ru	1.30	cs	2.52	cs	5.15	sl	0.92
3.	sv	8.31	en	1.05	ru	2.27	en	4.58	**cs**	0.90
4.	sl	7.82	cs	0.90	et	1.35	sv	4.35	he	0.88
5.	en	6.46	nl	0.82	sv	1.17	sl	4.05	ru	0.86
6.	no	3.43	sv	0.81	en	1.13	fi	2.94	en	0.81
7.	fi	2.73	sl	0.77	ro	1.01	ro	2.31	sv	0.76
8.	nl	2.61	ro	0.75	nl	1.00	no	2.24	fi	0.61
9.	bg	2.27	he	0.59	sl	0.99	nl	1.97	no	0.56
10.	ro	2.25	et	0.55	hr	0.66	bg	1.55	nl	0.53
11.	he	0.99	no	0.49	he	0.62	he	1.21	hr	0.50
12.	pt	0.90	hr	0.44	no	0.55	hr	0.93	pt	0.27
13.	hr	0.78	bg	0.28	bg	0.37	pt	0.75	et	0.21
14.	it	0.74	pt	0.22	fa	0.31	et	0.61	it	0.19
15.	es	0.63	fa	0.19	de	0.29	it	0.46	es	0.19
16.	et	0.59	de	0.16	pt	0.23	de	0.40	bg	0.19
17.	de	0.58	it	0.14	it	0.16	es	0.37	fa	0.16
18.	fr	0.28	es	0.12	es	0.13	fr	0.23	fr	0.11
19.	ca	0.26	fr	0.08	hi	0.09	ca	0.16	ca	0.11
20.	fa	0.11	**hi**	0.07	fr	0.09	fa	0.12	de	0.10
21.	hi	0.04	ca	0.06	ca	0.07	hi	0.05	hi	0.06
22.	ar	0.03	ar	0.04	ar	0.04	ar	0.04	ar	0.05

Table C.3: $KL^{-4}_{cpos^3}$ source-target similarities, part 3.