Cross-Lingual Transfer of Natural Language Processing Systems

Mohammad Sadegh Rasooli

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2019

ABSTRACT

Cross-Lingual Transfer of Natural Language Processing Systems

Mohammad Sadegh Rasooli

Accurate natural language processing systems rely heavily on annotated datasets. In the absence of such datasets, transfer methods can help to develop a model by transferring annotations from one or more rich-resource languages to the target language of interest. These methods are generally divided into two approaches: 1) annotation projection from translation data, aka parallel data, using supervised models in rich-resource languages, and 2) direct model transfer from annotated datasets in rich-resource languages.

In this thesis, we demonstrate different methods for transfer of dependency parsers and sentiment analysis systems. We propose an annotation projection method that performs well in the scenarios for which a large amount of in-domain parallel data is available. We also propose a method which is a combination of annotation projection and direct transfer that can leverage a minimal amount of information from a small out-of-domain parallel dataset to develop highly accurate transfer models. Furthermore, we propose an unsupervised syntactic reordering model to improve the accuracy of dependency parser transfer for non-European languages. Finally, we conduct a diverse set of experiments for the transfer of sentiment analysis systems in different data settings.

A summary of our contributions are as follows:

- We develop accurate dependency parsers using parallel text in an annotation projection framework. We make use of the fact that the density of word alignments is

a valuable indicator of reliability in annotation projection.

- We develop accurate dependency parsers in the absence of a large amount of parallel data. We use the Bible data, which is in orders of magnitude smaller than a conventional parallel dataset, to provide minimal cues for creating cross-lingual word representations. Our model is also capable of boosting the performance of annotation projection with a large amount of parallel data. Our model develops cross-lingual word representations for going beyond the traditional delexicalized direct transfer methods. Moreover, we propose a simple but effective word translation approach that brings in explicit lexical features from the target language in our direct transfer method.
- We develop different syntactic reordering models that can change the source treebanks in rich-resource languages, thus preventing learning a wrong model for a non-related language. Our experimental results show substantial improvements over non-European languages.
- We develop transfer methods for sentiment analysis in different data availability scenarios. We show that we can leverage cross-lingual word embeddings to create accurate sentiment analysis systems in the absence of annotated data in the target language of interest.

We believe that the novelties that we introduce in this thesis indicate the usefulness of transfer methods. This is appealing in practice, especially since we suggest eliminating the requirement for annotating new datasets for low-resource languages which is expensive, if not impossible, to obtain.

Contents

List of Figures

List of Tables

Acknowledgments

"Praise belongs to God who was kind to us through Mohammad, His Prophet (God bless him and his Household) to the exclusion of past communities and bygone generations, displaying thereby His power, which nothing can render incapable though it be great, and nothing can escape though it be subtle."

I would like to thank my advisor, Michael Collins, for his invaluable guidance, feedback, and support. It was an amazing experience to collaborate with such a knowledgeable, professional, respectful, and considerate advisor. There were many aspects of research that I was not really aware of when I started working with Mike. He taught me how to systematically approach a problem, how to think about empirical results, how to focus on my research goals, how to write and present my work in a precise and clear style, and to not give up when facing a real challenge in research. Many times I was shocked by the depth of his knowledge. He gave me the freedom to explore different ideas and challenge myself with new obstacles. He never overwhelmed me with irrelevant problems, and always asked me to solve research problems that would be beneficial to the world even if the benefits wouldn't manifest in the short term. He gave me the great opportunity to teach at his classes and, whenever applicable, made me aware of techniques to make a lecture more understandable. One great lesson that I learned from Mike was that

it is easy to present research or science material in a complicated way that is incomprehensible to almost everyone; but the more difficult task is to present that material in an easy-to-digest and coherent way.

When I was blessed with a baby during the last years of my Ph.D., Mike supported me and gave me more freedom to take care of my family. When I faced problems with my immigration documents at the end of my Ph.D that lead to many ancillary problems in my life, Mike supported me in different ways that many advisors wouldn't. He reached out to different people in the university international office, housing office, and other departments to make sure all other problems are alleviated. Thank you Mike for being such a great advisor. I will never forget all these years working under your supervision, the discussions that we had about different topics, the books on different topics including literature that you recommended I read, and the intellectual freedom that you gave me.

In the first two years of my Ph.D., I was advised by Nizar Habash. Nizar was the person who admitted me to Columbia, taught me how to do research, and after he decided to leave Columbia, recommended me to Mike. We closely collaborated with Owen Rambow, because of a joint project. I felt that I had two advisors in my first two years at Columbia. Thanks, Owen and Nizar, for teaching me how to conduct research, supporting me, and being very considerate. I had the chance to collaborate directly with Kathleen McKeown in a project. She always encouraged me through her kind words, great support, and inspiring ideas. I really appreciate her for all the help and guidance.

I had a great opportunity to intern at four different companies. At Nuance Communications, I worked in Ron Kaplans' group and conducted research under Joel Tetreault's supervision. Joel was so supportive, kind, and professional that words cannot convey my

appreciation. He is a great friend. I consulted with him in different stages of my professional life, and we had a chance to collaborate once more at Yahoo in Amanda Stent's group. I had the opportunity to work at Slav Petrov's group at Google, and conduct research under Kuzman Ganchev's supervision. I partly worked with Emily Pitler and benefited from collaborating with other researchers at Google. I am indebted to the great lessons that I learned from them. I had the opportunity to work at the language modeling team in Microsoft managed by Shawn Chang and I worked with Partha Parthasarathy. Partha is a knowledgeable researcher with many years of great experience in industry. Thanks, Partha and Shawn, for having me at your team. And thanks to all the researchers at Nuance, Yahoo, Google, and Microsoft for supporting me.

I would like to thank all my teachers at Columbia, namely Michael Collins, Nizar Habash, Michael Picheny, Bhuvana Ramabhadran, Stanley Chen, Shay Cohen, Clifford Stein, David Blei, Garud Iyengar, and the TAs that helped me understand concepts more deeply. A special thanks to my Ph.D committee, Kathleen McKeown, Owen Rambow, David Blei, Kuzman Ganchev, and Michael Collins, for all the time and energy they have put into reviewing my work and providing useful feedback and questions. I also want to thank Luis Gravano for serving in my proposal committee. I would like to thank other professors and research scientists in the NLP group, especially Smaranda Muresan and Julia Hirschberg.

Staying for six years at Columbia would not be possible without my kind colleagues at the Columbia NLP group and also CCLS. I would like to thank Karl Stratos, Avner May, Yinwen Chang, Andrei Simion, Sasha Rush, Noura Farra, Ramy Eskandar, Ahmed El Kholy, Ali Elkahky, Sakhar Alkhereyf, Erica Cooper, Tom Effland, Chris Hidey, Chris

Kedzie, Fei-tzin Lee, Sarah Ita Levitan, Jessica Ouyang, Wael Salloum, Victor Soto, Elsbeth Turcan, Olivia Winn, Apoorv Agarwal, Daniel Bauer, Or Biran, Heba Elfardy, Weiwei Guo, Yves Petinot, Vinodkumar Prabhakaran, Axinia Radeva, Sara Rosenthal, Kapil Thadani, Boyi Xie, Nadi Tomeh, Sarah Alkuhlani, Mahmoud Ghoneim, Melody Ju, and many others whose names I have forgoten to mention. Much of what I learned about research, life in the US, and cultural differences were from the bright people at Columbia. I would also like to deeply thank the Columbia libraries, especially the management and staff at the Butler library, for providing such as rich source of books in different languages especially Persian.

With an Iranian single-entry visa in the US, I was practically incarcerated in a huge prison. I didn't have thea chance to visit my country since 2012. I was not even able to present my papers in conferences outside the US. Without my Iranian and Muslim friends in the US, it would be impossible to bear with loneliness and restrictions I faced during this time. First and foremost, I would like to thank all the brothers and sisters of "Habl-e Matin," an informal metropolitan area religious community that later got this meaningful name. I also thank the great Islamic communities in both the New York City metropolitan area and the Bay area in California, for their support during my internships.

I arrived in the US with my wife, Maryam. She supported me. She consoled me when I faced problems in my life. She was the person that I could rely on. She showed a great deal of altruism in her life. First, she came with me knowing that during this journey she would not have a chance to visit her family. Then, she was willing to do her Ph.D. research remotely, even though we could have stayed near her school. Allah blessed us with a son, Ali. Without my wife it was not possible to bear all the difficulties of being away from

our family, having a baby, and continuing research. Ali played a big role in our life. He brought joy to our life. With the deepest possible love, thank you Maryam and thank you Ali. May Allah bless you both with his Mercy. I would like to thank my family, especially my mother, and my wife's family, especially my mother-in-law and brothers-in-law for their great help and support.

And our call will close with the words, "All praise is due to God, the Sustainer of all beings." "O Allah, be, for Your representative, Your Hujjah, Your blessings be on him and his forefathers, in this hour and in every hour, a guardian, a protector, a leader, a helper, a proof, and an eye, until You make him live on the earth, in obedience to You, and cause him to live in it for a long time."

To the memory of Ayatollah Mojtaba Tehrani (1937-2013) who always reminded us “Slogans can be found everywhere, but what must be sought is wisdom and intellect.”

Chapter 1

Introduction

> *"And among His Signs is the creation of the heavens and the earth, and the variations in your languages and your colors: verily in that are Signs for those who know."*
>
> — Quran, The Romans (30), 22.

Natural language processing, aka computational linguistics, is the task of understanding human languages with computers. One exciting aspect of this task is its interdisciplinary essence: researchers in different fields including artificial intelligence, applied linguistics, and social sciences are interested in making use of natural language processing to achieve a better understanding of language and human interactions. The explosion in using the internet, for which textual data is a fundamental content in it, has led industry to invest considerable resources in natural language processing.

There are different levels of linguistic information that are important to understanding languages. These levels include morphology (understanding internal word structure), syntax, semantics, discourse, and pragmatics [Jurafsky and Martin, 2009]. For example, Figure 1.1 shows an example of different types of linguistic information for a simple English sentence. Many artificial intelligence models have been used to make natural language processing more accurate; among them, machine learning has shown promising improvements in languages such as English. Figure 1.3 shows the progress in different tasks for the English language. These improvements are often achieved by learning a

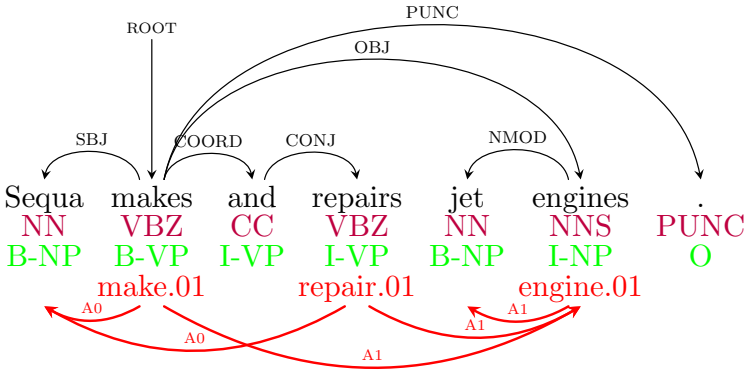


Figure 1.1: An English sentence with different types of linguistic information: arrows on top show syntactic dependencies based on the dependency grammar, arrows at the bottom show the semantic dependencies (dependency-based semantic role labeling), the two lines bellow the words show part-of-speech, aka word category, information and phrase boundary information where "B-" shows the start of a phrase and "I-" shows the middle of a phrase. The third row bellow the words show the word sense of predicate words (predicate word sense disambiguation).

machine learning model from a large corpus of text data annotated with linguistic information; these annotations are usually obtained by hiring expert linguists.

We focus on the tasks of dependency parsing and sentiment analysis in this thesis. In the following sections we first give some background on these tasks, then give an overview of the developed methods in this thesis.

1.1 Dependency Parsing

A dependency tree is a representation inspired by the dependency grammar, a grammar that is introduced by the French linguist, Lucien Tesniére [Tesnière, 1953] and has been reformulated in different linguistic studies such as the meaning-text theory [Mel'čuk and Polguere, 1987], and the word grammar [Hudson, 1984]. The basic assumption underlying dependency grammar is the idea that asymmetrical dependency relations between

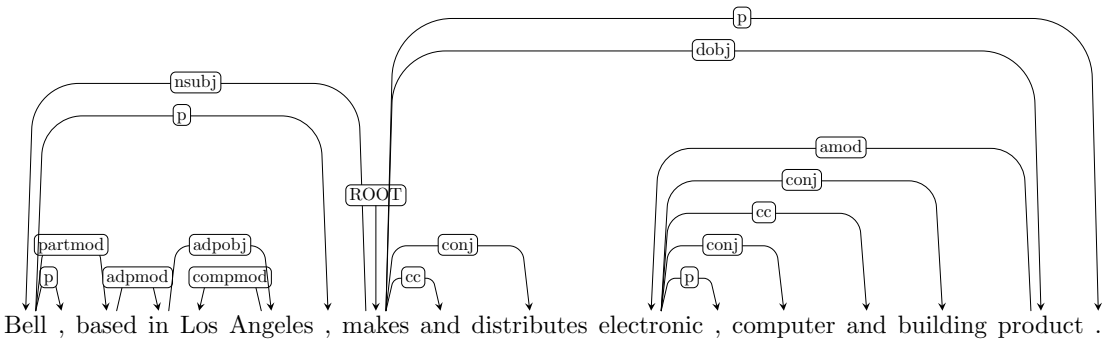


Figure 1.2: An example (projective) dependency tree from the Wall Street Journal treebank [Marcus *et al.*, 1993]. The dependencies are derived from the Google universal treebank annotation standard [McDonald *et al.*, 2013].

words create a dependency tree for a sentence [Kübler *et al.*, 2009]. In a dependency tree, each word has exactly one parent and an unrestricted number of dependents. The root for a tree is represented using the dummy *Root* token. An example dependency tree is shown in Figure 1.2. Statistical dependency parsers attempt to determine a dependency tree for a sentence such that the likelihood or score of that dependency tree is greater than that of other possible trees for that particular sentence. The likelihood or score can either be defined by a generative or a discriminative model.

Dependency parsing, compared to the phrase structure grammar, is a more flexible way to represent a tree structure for free word order languages [Kübler *et al.*, 2009]. Many applications including machine translation (e.g Quirk *et al.* [2005], Shen *et al.* [2008], Katz-Brown *et al.* [2011a], and Sennrich and Haddow [2016]), open information extraction (e.g. Wu and Weld [2010] , Angeli *et al.* [2015], and Bhutani *et al.* [2016]), semantic role labeling (e.g. Hacioglu [2004], Björkelund *et al.* [2009], and Marcheggiani and Titov [2017]), grammatical error correction [Tetreault *et al.*, 2010], disfluency detection (e.g. Rasooli and Tetreault [2013], Honnibal and Johnson [2014], and Yoshikawa *et al.* [2016]), and question

answering (e.g. Cui *et al.* [2005], and Hakimov *et al.* [2017]) make use of dependency trees.

1.2 Sentiment Analysis

Sentiment analysis is a standard multi-label classification problem. The input is a sequence of words and the output is the sentiment label of that sequence, usually among three possible labels: negative, positive, and neutral. The input text can be a sentence, a document or an entity in a sentence or document [Liu, 2012a]. For many simple examples in natural languages, one can employ simple indicator features such as the presence of the words "good" or "lazy" (e.g. "He is a good student." vs. "He is a lazy student.") to develop a sentiment analysis system. However, many real examples in natural languages comprise complicated linguistic structures that are not easy to understand by a system based on bag-of-words features. In recent years, different machine learning algorithms have been used to overcome the challenges in sentiment analysis [Pang *et al.*, 2002; Pang and Lee, 2004; Agarwal *et al.*, 2011; Socher *et al.*, 2011; Kim, 2014].

1.3 Our Motivation

When one wants to apply a state-of-the-art technique on a new language, the primary challenge would be to find an appropriate and relatively large annotated dataset. Unfortunately, this requirement might not be satisfied for many languages in the world. Moreover, annotating a new dataset is very expensive and time-consuming; e.g., annotating the Chinese treebank took five years of effort [Hwa *et al.*, 2005]. Unsupervised learning could be a potential solution to this problem. In unsupervised language learning, the goal is to learn a language without using annotated data in that language. There

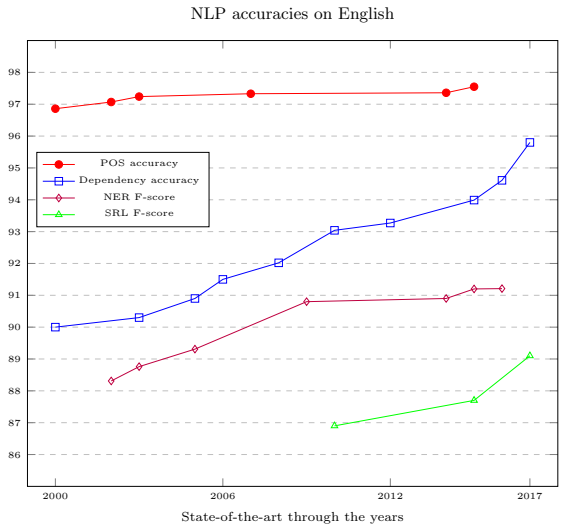


Figure 1.3: State-of-the-art progress on English natural language processing tasks including part-of-speech (POS) tagging, dependency parsing (based on unlabeled attachment score), named entity recognition (NER) based of F-score, and dependency-based semantic role labeling (SRL) F-score.

has been a considerable amount of work on unsupervised learning in different tasks, e.g., Figure 1.4 shows the progress in unsupervised dependency parsing on English as a simulation for a truly low-resource language. As shown in the figure, the improvements are substantial but compared to a supervised parser, the best accuracy of an unsupervised parser lags considerably behind that of a supervised parser.

An alternative approach is to use *transfer* methods. The core idea behind transfer methods is to make use of annotations in rich-resource languages and transfer those annotations to a language without annotation. Thus we can benefit from the fact that despite idiosyncratic linguistic phenomena in different languages, there are still many similarities that we can use to transfer information from one language to another one. Transfer methods can leverage parallel data, aka translated text, along with supervised models in rich-resource languages. For example, in *annotation projection*, we have parallel data for

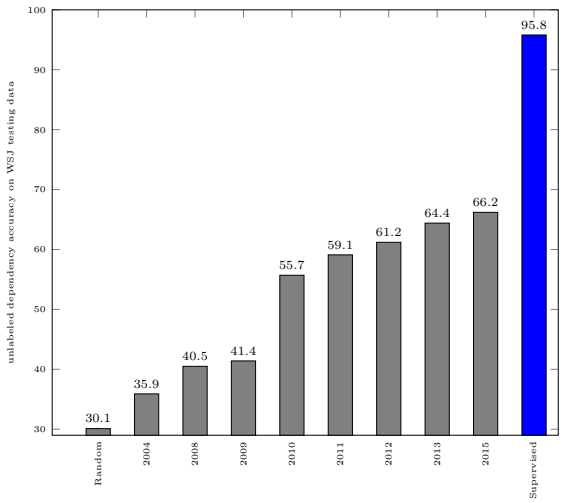


Figure 1.4: State-of-the-art progress in unsupervised dependency parsing of English. The models are from Klein and Manning [2004], Cohen *et al.* [2009], Cohen and Smith [2009], Blunsom and Cohn [2010], Spitkovsky *et al.* [2011a], Spitkovsky *et al.* [2012], Spitkovsky *et al.* [2013], and Le and Zuidema [2015]. The supervised model is from the result of Dozat and Manning [2016].

which sentences in a rich-resource language are translated to sentences in a low-resource language. We can process the sentences in the rich-resource language with a supervised model, and then project that supervised information using the translation cues.

1.4 Transfer Methods

Transfer methods aim to transfer common universal linguistic information from one or many rich-resource languages to a low-resource target language. There are generally two types of transfer methods: annotation projection and direct transfer, aka direct model transfer. A combination of annotation projection and direct transfer can also be used: in chapters 5 and 7, we show that we can leverage the combined method.

1.4.1 Annotation Projection

Annotation projection is a classic but effective cross-lingual transfer technique, especially when parallel corpora from multiple sources are available. In *single-source* annotation projection, it is assumed that a parallel corpus is available where sentences $\{s^{(i)}\}_{i=1}^{m}$ from the source language L_s are aligned to sentences $\{t^{(i)}\}_{i=1}^{m}$ in a low-resource language L_t; i.e. $s^{(i)}$ and $t^{(i)}$ are translations of each other for $i = 1 \dots m$.

Labeled training examples $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ exist for the source language L_s. Thus, a supervised system $\mathcal{M}_{sup}$ is first trained on $\mathcal{D}$ and afterward, the trained system $\mathcal{M}_{sup}$ is used to predict the labels of the source-side parallel translated text $\{y_{s^{(i)}}\}_{i=1}^{m}$. Those labels are projected to the target-side translation; i.e. $y_{s^{(i)}} \rightarrow y_{t^{(i)}}$. After applying projection, the target-side translation text $\{t^{(i)}\}_{i=1}^{m}$ is used along with the projected labels $\{y_{t^{(i)}}\}_{i=1}^{m}$ to train the target model $\mathcal{M}_{proj}$. The trained model $\mathcal{M}_{proj}$ is the final model for analyzing text in the target language.

Multi-source transfer is an extension to single-source transfer. In this setting, there are k source languages, each with separate labeled training data. It is also assumed that there exists multi-parallel corpora where every target language sentence has k source language translations. In this setting, one can apply the same projection technique to create a supervised model $\mathcal{M}_{sup}^{i}$ for each source language L_i ($i = 1, \cdots k$) and project labels to the target language sentence. Then majority voting is applied on the k projected labels for each target sentence to get the most reliable projected label. As with single-source projection, a supervised system is then trained on the projected data to get the final model $\mathcal{M}_{proj}$.

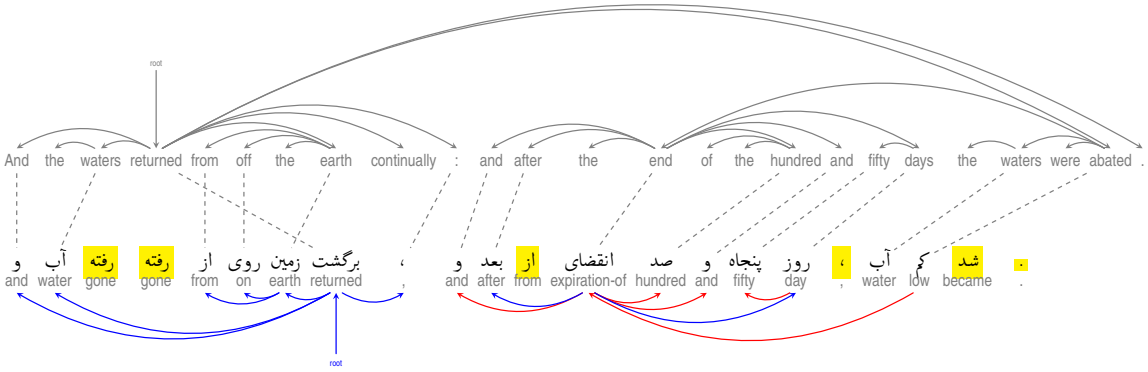


Figure 1.5: Example of annotation projection for dependency parsing from English to Persian in the Bible data. Persian words are written from left to right for the ease of representation, and the lexical translations are written bellow each word. Red dependencies are the wrong ones, and yellow words have missing alignment.

Projecting labels to a target language is straightforward for traditional classification tasks such as sentiment analysis. In structured prediction, such as dependency parsing, projection is more complicated than simply projecting one label from a source sentence to a target sentence. Multi-source projection might need a dynamic programming algorithm to extract a tree (or a set of trees in the case of having missing alignments) from a set of possible trees in the projected dependencies. Figure 1.5 shows an example from an English sentence to a Persian sentence. As shown in the figure, different types of error occur due to different reasons, such as an incorrect supervised parse, incorrect alignments and translation divergence.

1.4.2 Direct Transfer

In direct transfer, a system can be trained on a source language dataset directly, with an unrestricted number of source languages. In other words, a supervised system can be trained on the concatenation of k labeled datasets $\cup_{i=1}^{k} \mathcal{D}_i$ from k source languages

or it can use the outcome of an ensemble of k single-source direct transfer models. The advantage of direct transfer is the ability to directly use the gold labels of the source data in a single model without having to train separate source-language and target-language models. With multi-source, it can directly combine training data from different source languages in a single model. Moreover, the dependence on parallel data is reduced; it is possible to train a direct transfer model without any parallel data, namely if a bilingual dictionary or parallel data is available.

The main problem with direct transfer is that most of the common features do not generalize beyond each source language: for example, lexical features in one language are unlikely to appear in other languages. Moreover, a direct transfer model might be incorrect due to the differences in syntactic word order or other linguistic differences across languages.

1.5 Thesis Overview

The primary contribution of this thesis is to show the effectiveness of different transfer approaches for natural language processing both in scenarios for which a large set of translation data is available and in scenarios for which we have limited access to annotated data. The summary of our contributions are as follows:

- **Dependency parsing**:
  - We present a model (chapter 4) that can leverage a large amount of parallel text to develop dependency parsers without supervision in target languages. Our experimental results show that our models outperform substantially over

previous work in transfer methods and unsupervised parsing.

  - We present a model (chapter 5) that tailors a small parallel text such as the Bible data as well as treebanks in rich-resource languages to develop very accurate dependency parsers in a target language.
  - We present a state-of-the-art transfer method (chapter 6) that tries to reorder syntactic trees in rich-resource treebanks before using them in a transfer scenario. We show that this method improves over strong transfer methods especially for non-European languages, in the absence of large parallel data.

- **Sentiment analysis**: We conduct a broad set of experiments to show that similar to dependency parsing, transfer methods can work well in sentiment analysis for low-resource languages (chapter 7) .

The mentioned contributions are important to us in different aspects: first and foremost, we give different solutions for developing accurate natural language processing models in the absence of annotated data. For example, when using the small Bible parallel dataset, our final dependency model improves the unlabeled attachment accuracy by a 10% absolute difference over a strong baseline on 68 datasets (38 languages). Table 1.1 compares our results in Chapter 5 to previous work: as shown in the table, our models outperforms all previous work with a large gap, and performs close to a supervised parser. There is a clear reason why not relying on annotated data is important in low-resource languages. We are not required to spend a huge amount of money to annotate a data. Moreover, our approach can give us a natural language processing model in a short amount of time as opposed to a long time annotation effort.

Language	MX14	LA16	ZB15	GCY16	AMB16	Our Model	Supervised
en	–	–	70.5	–	–	**77.5**	93.8
de	74.3	76.0	62.5	65.0	65.2	**82.1**	85.3
es	75.5	78.9	78.0	79.0	80.2	**82.6**	86.7
fr	76.5	80.8	78.9	77.6	80.6	**83.9**	86.3
it	77.7	79.4	79.3	78.4	80.7	**84.4**	88.8
pt	76.6	–	78.6	81.8	81.2	**85.6**	89.4
sv	79.3	83.0	75.0	78.2	79.0	**84.5**	88.1
$avg_{\backslash en}$	76.7	–	75.4	76.3	77.8	**83.9**	87.4

Table 1.1: Comparison of our work using the Europarl data with previous work in terms of unlabeled attachment score: MX14 [Ma and Xia, 2014], LA16 [Lacroix *et al.*, 2016], ZB15 [Zhang and Barzilay, 2015], GCY16 [Guo *et al.*, 2016], and AMB 16 [Ammar *et al.*, 2016b],. "Supervised" refers to the performance of the parser trained on fully gold standard data in a supervised fashion (i.e. the practical upper-bound of our model). "$avg_{\backslash en}$" refers to the average accuracy for all datasets except English.

This thesis is organized as follows: Chapter 2 provides background on natural language processing and machine learning, Chapter 3 briefly overviews related work on transfer methods in dependency parsing and sentiment analysis, Chapter 4 describe a transfer method for dependency parsing based on annotation projection, Chapter 5 describes our approach for transfer of dependency parsers in the absence of large amounts of parallel data, Chapter 6 describes a method for improving the transfer of dependency parsers with syntactic reordering of source treebanks in rich-resource languages, Chapter 7 conducts a broad set of experiments on different transfer methods for sentiment analysis in different data settings, and Chapter 8 makes a conclusion.

The work in Chapter 4 is published as a conference paper [Rasooli and Collins, 2015], and the work in Chapter 5 is published as a journal article [Rasooli and Collins, 2017]. The work in Chapter 7 is a joint work with colleagues at Columbia University and is published as a journal article [Rasooli *et al.*, 2017]. The latter work has some other parts such as using comparable corpora but since I was not directly involved in that part of research, I have

not included that in Chapter 7.

Chapter 2

Background

In this chapter, we briefly overview some of the natural language processing and machine learning methods that we use in this thesis.

2.1 Natural Language Processing

We now overview some of the algorithms in natural language processing that we use in this thesis.

2.1.1 Automatic Word Alignment

Word alignment models aim to find the best translation links between words in a pair of translated sentences. Previous work [Brown *et al.*, 1993; Vogel *et al.*, 1996; Och and Ney, 2003; Dyer *et al.*, 2010; Dyer *et al.*, 2011; Dyer *et al.*, 2013; Simion *et al.*, 2013] has considered automatic word alignment as an unsupervised learning problem. The seminal IBM models [Brown *et al.*, 1993] and the HMM-based model [Och and Ney, 2003] are the ones that we use in our experiments.

In general, for a sentence pair $(s^{(i)}, t^{(i)})$ where $s^{(i)} = [s_1^{(i)}, \cdots, s_{m_i}^{(i)}]$ and $t^{(i)} = [t_1^{(i)}, \cdots, t_{k_i}^{(i)}]$, the goal is to find source-to-target alignments $a^{(i)} = a_1^{(i)}, \cdots, a_{k_i}^{(i)}$ where $a_j^{(i)}$ shows the index of a source word that is aligned to the jth target word where $a_j^{(i)} = 0$

indicates a NULL alignment. The IBM models assume that a source word can be aligned to zero or many target words. The relation between an alignment a, a source sentence s and a target sentence t is formulated as the joint probability of the target sentence and its alignment links given the source sentence. The best alignment parameter θ^* is found as:

$$\begin{aligned}\theta^* &= \arg\max_{\theta} \prod_{i=1}^{N} \sum_{a^{(i)}} p_\theta(t^{(i)}, a^{(i)}|s^{(i)}) \\ &= \arg\max_{\theta} \prod_{i=1}^{N} \sum_{a_1^{(i)}=0}^{m_i} \sum_{a_2^{(i)}=0}^{m_i} \cdots \sum_{a_{k_i}^{(i)}=0}^{m_i} p_\theta(t^{(i)}, a_1^{(i)}, \cdots, a_{k_i}^{(i)}|s^{(i)})\end{aligned}$$

The Expectation-Maximization (EM) algorithm [Dempster *et al.*, 1977] is used to obtain the best parameter θ^* for the alignment model. After finding θ^*, we can find the best scoring alignment $a^{*(i)}$ for each sentence pair $(s^{(i)}, t^{(i)})$ with a greedy argmax function.

Although one-to-many alignment is a realistic phenomenon is natural languages, some tasks such as annotation projection for dependency parsing need one-to-one alignments. One simple way to obtain one-to-one alignments is to run the alignment model twice: once from source to target, and once from target to source. By getting the intersected alignments, we miss some of the alignments that only occur in one side of the alignments but the final alignments will always be one-to-one and are high precision. Figure 2.1 shows a simple example of word alignment.

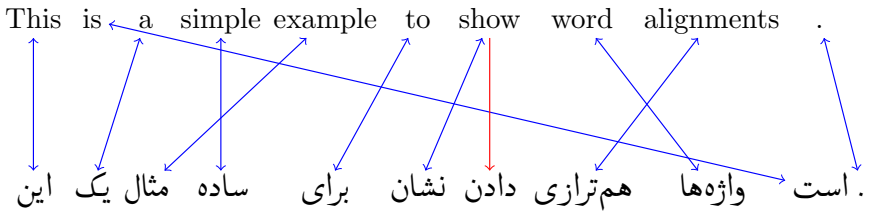


Figure 2.1: A simple example of word alignments for an English-Persian translation pair. The Persian sentence is shown from left to right for the ease of representation, while Persian is written from right to left. Bidirectional arrays indicate that an alignment is found both in the source-to-target and target-to-source directions. The red arrow indicates that the word in the Persian sentence does not get any target-to-source alignments, leading to a NULL alignment for "دادن" after running alignment intersection.

2.1.2 Dependency Parsing

In this section, after providing some formal definitions, we describe those parsing algorithms that we use throughout this thesis.

Definition 2.1 ***Dependency tree****: A dependency tree y for sentence $x = [x_1, \cdots, x_n]$ is defined as a set of dependency arcs $y = \{(i \rightarrow j);\ \forall\, 1 \leq j \leq n \,\&\, 1 \leq i \leq n+1 \,\&\, i \neq j\}$, such that the following conditions are hold [Kübler* et al.*, 2009]:*

- *Root property:* $\exists\, j \ \ s.t \ \ (n+1 \rightarrow j) \in y$.
- *Single-head property:* $(i \rightarrow j) \in y \ \Rightarrow \ \nexists\, k \neq i \ s.t. \ (k \rightarrow j) \in y$.
- *Acyclicity property:* $(i \rightarrow j) \in y \ \Rightarrow \ \nexists\, j \rightarrow^* i$*; where* $j \rightarrow^* i$ *denotes a dependency path from node* j *to* i.

Definition 2.2 ***Projective tree****: A tree is projective if and only if one can draw its dependency arcs on top of words without creating any crossing arcs (as in Figure 1.2). Formally,*

the following condition should hold for a tree to be projective:

$$\nexists\ (k, m, l, u) \texttt{ such that}$$

$$m < l < k < u \texttt{ or } l < k < u < m$$

$$\texttt{and } ((l \rightarrow u) \in y \texttt{ or } (l \leftarrow u) \in y)$$

$$\texttt{and } ((m \rightarrow k) \in y \texttt{ or } (m \leftarrow k) \in y)$$

Any tree that is not projective is called nonprojective; nonprojective trees occur more frequently in free-word order languages. Projective parsing is usually easier [McDonald and Satta, 2007] and most sentences in many languages are projective. There exists different parsing algorithms for deriving projective and nonprojective dependency trees.

Graph-based Parsing

Graph-based parsers define the parsing problem as finding the maximum spanning tree of a directed graph. These parsers define the score of a tree as the sum of the score of each arc in the tree. Figure 2.2 shows one such example in which the best tree is recovered from the space of all possible trees, an exponential number of possibilities. Thanks to independence assumption between arcs, dynamic programming solutions for such a problem exist for projective parsing [Eisner, 1996; McDonald *et al.*, 2005a] and an efficient algorithm exists for nonprojective parsing [Chu, 1965; Edmonds, 1967; Tarjan, 1977; McDonald *et al.*, 2005b].

Definition 2.3 *Score of a dependency tree y in a first-order graph-based model is defined as*

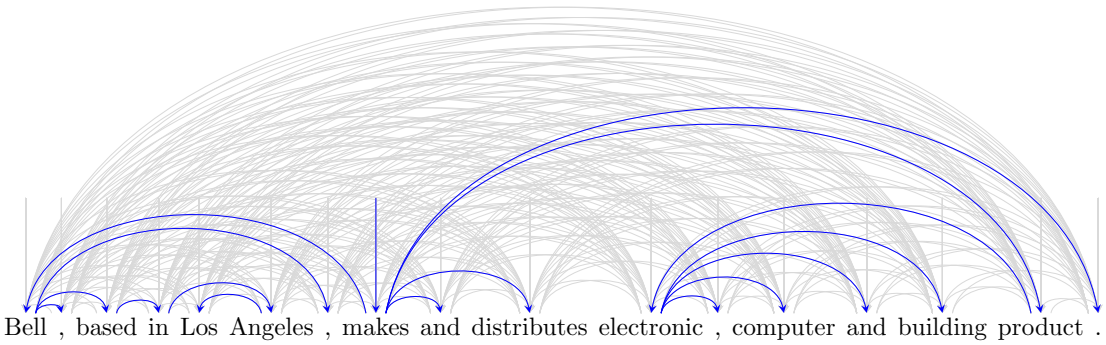


Figure 2.2: All possible dependency arcs for the English sentence in Figure 1.2. The dark edges show the correct dependencies. Graph-based parsers use dynamic programming to recover the best tree in polynomial time.

the sum of the score of each arc in that tree:

$$score(y) = \sum_{(i \rightarrow j) \in y} \lambda(i, j)$$

where $\lambda(i, j)$ *is the score of the* $(i \rightarrow j)$ *arc.*

Definition 2.4 *The goal of a first-order graph-based parser is to find the highest-scoring dependency tree for a sentence* x*:*

$$y^* = \arg \max_{y \in \mathcal{Y}(x)} score(y)$$

where $\mathcal{Y}(x)$ *is the space of all possible dependency trees for sentence* x*. The* arg max *function is often solved by a dynamic programming algorithm. Depending on the projectivity constraint,* $\mathcal{Y}(x)$ *can only enumerate projective, or both projective and nonprojective trees.*

In this thesis, we only use the first-order graph-based parsers: the Eisner algo-

rithm [Eisner, 1996] can recover the best projective tree in $O(n^3)$ for a sentence of n words; and the Chu-Liu-Edmunds algorithm [Chu, 1965; Edmonds, 1967] does nonprojective parsing with $O(n^2)$ time complexity. It is worth noting that there are different extensions of graph-based parsing such as the higher-order arc-factored models (e.g. McDonald and Pereira [2006], Carreras [2007], Koo and Collins [2010], and Ma and Zhao [2012]).

Transition-based Parsing

Transition-based, aka shift-reduce, parsing is one of the most popular frameworks in dependency parsing. A transition system is an abstract machine that contains a *configuration* (a current parsing state) and transitions between configurations. The goal is to map a sentence x to a tree y by applying incremental transitions. The core idea is to start from a pre-defined initial state configuration and apply a sequence of deterministic transitions to reach a pre-defined terminal state.

Definition 2.5 *Given a set R of dependency types, and a sentence $x = [x_1, ...x_n, x_{n+1}]$ where x_{n+1} is the* Root *token, a configuration for x is a triple $c = (\sigma, \beta, A)$, where*

1. *σ is a sequence of indices i corresponding to positions in x; σ is usually referred as a* stack*,*
2. *β is a sequence of indices i corresponding to positions in x; β is usually referred as a* buffer*,*
3. *A is a set of dependency arcs $(i \xrightarrow{r} j)$ where $1 \leq i \leq n+1, 1 \leq j \leq n$, and $r \in R$.*

The initial configuration comprises a buffer with all the words of the sentence in order in which they appear, and an empty stack. In the terminal state, the buffer contains the

dummy *Root* token and the stack is empty.[1]

Different transitions have been suggested for transition-based parsing: arc-eager [Nivre, 2003], arc-standard [Nivre, 2004] and arc-hybrid [Kuhlmann *et al.*, 2011], etc. Usually these algorithms perform comparably, with slight deviations depending on the dataset to which they are applied.

Definition 2.6 *The arc-eager transitions are as follows [Nivre, 2003]:*

- **Shift** $(\sigma, i|\beta, A) \Rightarrow (\sigma|i, \beta, A)$
- **Right-Arc(r)** $(\sigma|i, j|\beta, A) \Rightarrow (\sigma|i|j, \beta, A \cup (i \xrightarrow{r} j))$
- **Left-Arc(r)** $(\sigma|i, j|\beta, A) \Rightarrow (\sigma, j|\beta, A \cup (j \xrightarrow{r} i))$
- **Reduce** $(\sigma|i, \beta, A) \Rightarrow (\sigma, \beta, A)$

Figure 2.3 shows a sequence of transitions for a short English sentence with the arc-eager algorithm.

One of the challenges in training transition-based parsers is the construction of training data; there is spurious ambiguity in transition-based parsers such that the same tree can be derived from multiple different transition sequences. Early work [Nivre *et al.*, 2006] considered a set of rules to create static oracles that generate a deterministic transition for a configuration with more than one possible gold-standard transition . A better alternative is to use dynamic oracles [Goldberg and Nivre, 2013], which are dynamically created depending on the current classifier weights and the current sentence. One inter-

[1] Depending on the transition system and the initial place of the dummy root token, this condition might slightly differ.

Act.	Stack	Buffer	Arc(h,d)
Shift	[]	[I_1, $want_2$, to_3, $parse_4$, a_5, $sentence_6$, .$_7$, $ROOT_8$]	
Left-Arc(nsubj)	[I_1]	[$want_2$, to_3, $parse_4$, a_5, $sentence_6$, .$_7$, $ROOT_8$]	($2 \xrightarrow{nsubj} 1$)
Shift	[]	[$want_2$, to_3, $parse_4$, a_5, $sentence_6$, .$_7$, $ROOT_8$]	
Shift	[$want_2$]	[to_3, $parse_4$, a_5, $sentence_6$, .$_7$, $ROOT_8$]	
Left-arc(aux)	[$want_2$, to_3]	[$parse_4$, a_5, $sentence_6$, .$_7$, $ROOT_8$]	($4 \xrightarrow{aux} 3$)
Right-arc(xcomp)	[$want_2$]	[$parse_4$, a_5, $sentence_6$, .$_7$, $ROOT_8$]	($2 \xrightarrow{xcomp} 4$)
Shift	[$want_2$, $parse_4$]	[a_5, $sentence_6$, .$_7$, $ROOT_8$]	
Left-arc(det)	[$want_2$, $parse_4$, a_5]	[$sentence_6$, .$_7$, $ROOT_8$]	($6 \xrightarrow{det} 5$)
Right-arc(dobj)	[$want_2$, $parse_4$]	[$sentence_6$, .$_7$, $ROOT_8$]	($4 \xrightarrow{dobj} 6$)
Reduce	[$want_2$, $parse_4$, $sentence_6$]	[.$_7$, $ROOT_8$]	
Reduce	[$want_2$, $parse_4$]	[.$_7$, $ROOT_8$]	
Right-arc(punct)	[$want_2$]	[.$_7$, $ROOT_8$]	($2 \xrightarrow{punct} 7$)
Reduce	[$want_2$, .$_7$]	[$ROOT_8$]	
Left-arc(root)	[$want_2$]	[$ROOT_8$]	($8 \xrightarrow{root} 2$)
DONE!		[$ROOT_8$]	

Figure 2.3: A sample action sequence with arc-eager actions for the sentence "I want to parse a sentence.".

esting property of dynamic oracles is that they enable us to easily train on partial trees or parse a partial tree such that the given arcs are preserved after parsing.

Beam Decoding for Transition-Based Parsing A greedy transition-based parser might make a wrong decision in the earlier configurations. An early mistake propagates to next states, leading to a incorrect parse tree with many incorrect dependencies. This is in contrast to the graph-based parser in which the decisions are optimal according to local features. Beam decoding is a way to compensate for the greedy mistakes. For a transition system with $2n$ actions, for every action step from 1 to $2n$, the parser keeps a maximum of b different configurations in memory, and expands all of them to generate new configurations for the parser. The beam decoder selects the b highest scoring new configurations among the generated ones to step forward to the next action step. In the $2n$th step, the parser picks the highest scoring configuration as the final decision and picks its arc set as the output parse tree. In practice, adding the size of the beam more

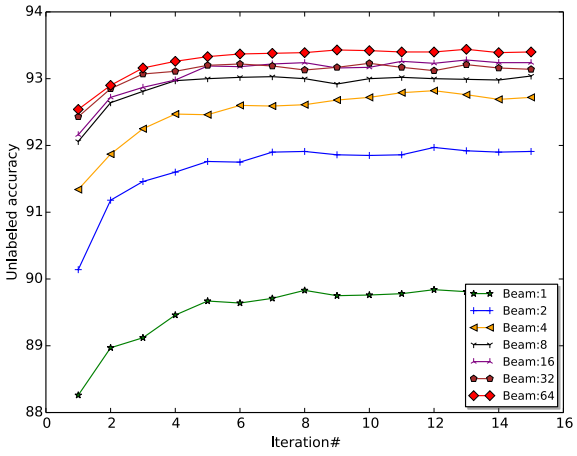


Figure 2.4: The influence of beam size on each training iterations for Yara parser [Rasooli and Tetreault, 2015] on the Penn Treebank development data [Marcus *et al.*, 1993].

than a certain threshold—usually between 32 and 100— does not lead to any improvement in the final performance. Figure 2.4 shows the accuracy of beam search for the arc-eager algorithm with different beam sizes for an arc-eager parser [Rasooli and Tetreault, 2015]. As shown in the figure, the accuracy of the parser converges to a certain number after the beam size of 64.

In practice, one should train a beam decoder in beam mode [Zhang and Clark, 2008]. In this mode, a scoring function from a classifier helps the current model to expand the beam and find the best possible search path to the final state. Afterwards, the parser updates the classifier weights with respect to the best search path vs. the gold-standard path. This strategy is called "standard update": the parser waits until the decoder reaches the final state and then updates the parameters of the classifier. In practice, this strategy does not lead to a set of useful parameters: the parser might explore some search spaces that are very different from the gold-standard path. Moreover, it slows down learning and gives lower performance [Huang *et al.*, 2012]. There are different strategies to compensate for

this problem, in which two of them are frequently used in transition-based dependency parsing:

- Early update [Collins and Roark, 2004]: The parser stops exploring the beam whenever the gold-standard path falls off the beam.
- Max-violation update [Huang *et al.*, 2012]: The decoder explores the beam until the terminal state but updates the beam with respect to a state in which the highest-scoring element in the beam has the largest score difference to the correct action sequence; the correct action sequence does not necessarily need to stay in the beam. Max-violation represents the worst mistake that the classifier makes in the beam compared to the gold action. In general, this strategy leads to a fewer number of training epochs and slightly better performance for dependency parsing.

2.1.3 Word Representations

The main type of features used in different natural language processing applications are lexical features; i.e. words. Depending on the task, other features such as part-of-speech tags are used but the most influential type of feature is the lexical features. Empirically, ignoring lexical features decreases the final performance of any natural language processing system significantly. There are three main issues when one deals with lexical features:

- **Out-of-vocabulary words**: Word frequencies in languages follow the Zipfian distribution: the frequency of each word is inversely proportional to its rank in the frequency list of a large sample text of that language [Zipf, 1935]. That being said, most

of the words in a text corpus have a very low frequency even in a large sample, leading to a distribution with a heavy tail. As a consequence, many words in any testing data are not seen in the training data: the problem of out-of-vocabulary words is a challenge in natural language processing. Figure 2.5 shows the logarithm-scale frequency of words in the Penn Treebank [Marcus *et al.*, 1993]. As shown in the figure, nearly half of the word types occur only once in the data.

- **Representing categorical features**: In traditional machine learning methods such as the maximum entropy models [Ratnaparkhi, 1996], each feature is converted to a binary indicator feature, indicating whether a categorical feature exists in the input (as value 1) or not (as value 0) . That leads to a huge sparse feature vector in which only a few of the values in the vector are non-zero.
- **Cross-lingual lexical features**: When dealing with cross-lingual data, as in our case, vocabularies have a small overlap across languages. One solution would be to represent lexical features of all languages in a shared space where semantically similar words are similar in the shared space. One can create a lexical abstraction by mapping words to vectors or cluster ids.

We use two types of word representation throughout this thesis: 1) hierarchical word clusters, and 2) word embedding vectors.

Hierarchical Word Clusters

A *clustering* is a function $C(w)$ that maps each word w in a vocabulary to a cluster $C(w) \in \{1 \dots K\}$, where K is the number of clusters. A *hierarchical clustering* is a func-

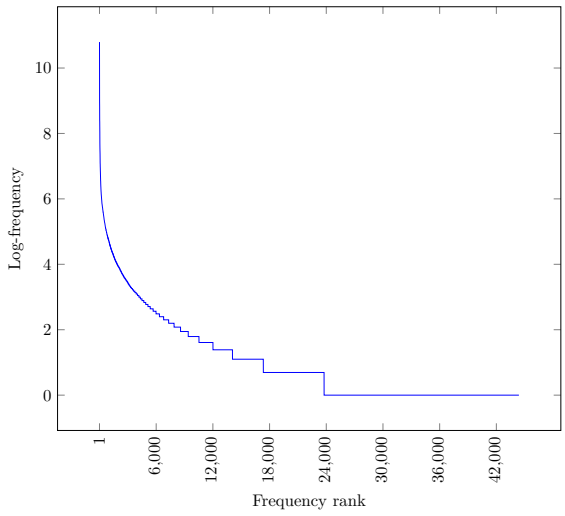


Figure 2.5: Zipfian distribution of the English words in the training section of the Penn Treebank [Marcus *et al.*, 1993].

tion $C(w, l)$ that maps a word w together with an integer l to a cluster at level l in the hierarchy. The goal of hierarchical word clustering is to map each word in a vocabulary to a point in a hierarchy in which words in a close neighborhood in the hierarchy should be syntactically or semantically similar. As one example, the Brown clustering algorithm [Brown *et al.*, 1992] gives a hierarchical clustering. The level l allows cluster features at different levels of granularity.

Brown Clustering Brown clustering [Brown *et al.*, 1992] is a class-based bigram language model in which each word in the vocabulary receives a unique bitstring as its cluster identity. Figure 2.6 shows a small vocabulary with the hierarchical bitstring assignments from the Brown clustering algorithm. One can view this algorithm as unsupervised part-of-speech induction with a bigram transition model such that every word in the training data can have only one possible part-of-speech tag. Probability of a data with N tokens and n unique words $w_1, \cdots, w_n$ is defined as the product of their bigram transitions. Here

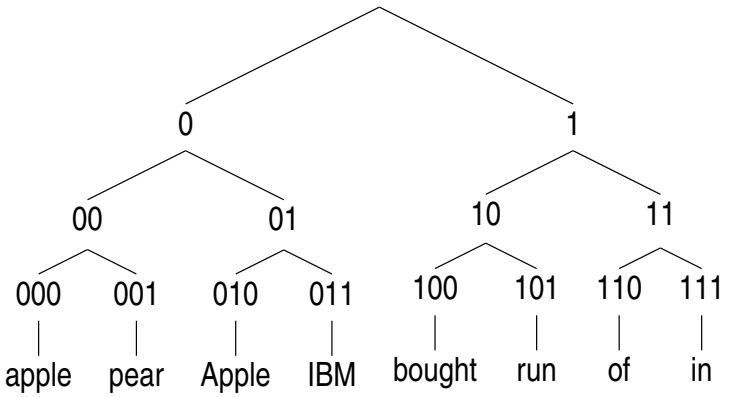


Figure 2.6: A simplified depiction of the hierarchical word clusters for some English words. This example is taken from the hierarchy shown by Koo *et al.* [2008].

the algorithm assumes that every word w_i has a unique cluster assignment $C(w_i)$:

$$P(w_i|w_{i-1}) = P(w_i|C(w_i))P(C(w_i)|C(w_{i-1}))$$

Definition 2.7 *Given a clustering assignment C, the quality measure for that assignment with respect to the data is defined as following:*

$$Quality(C) = \frac{\log P(w_1, \cdots, w_n)}{n} = \frac{\log \prod_{i=1}^{n} P(w_i|C(w_i))P(C(w_i)|C(w_{i-1}))}{n}$$

The original work by Brown *et al.* [1992] uses a greedy heuristic algorithm to find the best clustering of the data. A naive implementation has a time complexity of $O(n^5)$. Brown *et al.* [1992] proposed and algorithm to reduce this runtime to $O(n^3)$. This runtime is still not tractable for a large corpus. To solve this issue, a precomputation trick is used to reduce this runtime to $O(N + nm^2)$ in which m is the number of clusters. The

Inputs: Corpus with N tokens and n distinct word types $w_1, \cdots, w_n$ ordered by decreasing frequency; m: number of clusters where $m \leq n$.

Algorithm:

Initialize active clusters $C = \{\{w_1\}, \cdots, \{w_m\}\}$

for i = m+1 to $n + m - 1$ **do**

  if $i \leq n$ **then**

    Set $C = C \cup \{\{w_i\}\}$

  Merge $c, c' \in C$ that cause the smallest decrease in the likelihood of the corpus (def. 2.7).

Output: The clustering $\mathcal{C}$.

Figure 2.7: Brown clustering algorithm.

algorithm in figure 2.7 shows the pseudo-code for deriving the Brown clusters using the greedy heuristic. As shown in the pseudo-code, the algorithm starts with assigning m unique clusters to the m most frequent words. It then visits other words according to their decreasing frequency order, assigns a new ($(m+1)$th) cluster to the new word, and merges two of the clusters based on the quality measure. Thus at each step, the algorithm ends up with exactly m clusters.

When dealing with a very large corpus, even the $O(N+nm^2)$ is very time-consuming. There are other alternatives, such as [Stratos *et al.*, 2015], to obtain the clustering assignments with a more efficient computational complexity. Using word clusters has shown promising results in dependency parsing [Koo *et al.*, 2008], part-of-speech tagging and named entity recognition [Turian *et al.*, 2010]. Word clusters are usually used as additional features in a traditional classifier such as the log-linear model or the Perceptron algorithm.

Word Embeddings

Word embedding models embed each word into a d-dimensional vector by creating a matrix in $\mathbb{R}^{N\times d}$ from a vocabulary with N words. The goal is to obtain a set of vectors for words in the vocabulary such that semantically similar words are similar in vector space. One famous model for obtaining word embeddings is the Skip-gram model by Mikolov *et al.* [2013b]: given a sequence of training words $w_1, \cdots, w_T$, the objective function of the Skip-gram model maximizes the average log-probability of the data:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_O = w_{t+j}|w_I = w_t)$$

where c is the size of the training context. The probability $p(w_O|w_I)$ can be defined proportional to the dot product of the "word" representation of w_O and "context" representation of w_I. Defining two different vectors for context words and words gives more flexibility to the model for distinguishing between the word being targeted and its context:

$$p(w_O|w_I) = \frac{exp(v'^{\top}_{w_O} v_{w_I})}{\sum_{w\in\mathcal{V}} exp(v'^{\top}_{w} v_{w_I})}$$

where $\mathcal{V}$ is the vocabulary. The softmax function in the above equation normalizes over a large vocabulary $\mathcal{V}$. This normalization is computationally expensive and memory intensive. Mikolov *et al.* [2013b] use an approximate softmax function by sampling k words from the vocabulary:

$$\log\sigma(v'^{\top}_{w_O} v_{w_I}) + \sum_{i=1}^{k}\mathbb{E}_{w_i\sim P_n(\mathcal{V})}[\log\sigma(v'^{\top}_{w_i} v_{w_I})]$$

where σ is the sigmoid function, and $P_n(w)$ is the noise distribution that draws k sample words from the vocabulary. Mikolov *et al.* [2013b] empirically found that the unigram distribution proportional to the unigram count of each word raised by $\frac{3}{4}$rd power gives a better performance than the uniform and flat unigram distributions in standard word analogy and similarity tasks. With this approximate method, called *negative sampling*, we are able to maximize the likelihood of a pair of input and context words in contrast to k noise context examples.

Skip-gram model is not the only way to achieve word embedding vectors. Other methods such as the continuous bag-of-words (CBOW) model [Mikolov *et al.*, 2013a], Glove [Pennington *et al.*, 2014], and the spectral method of Stratos *et al.* [2015] exist. It is not clear which model gives the best accuracy: it depends on the target task in which those embeddings might be used.

2.2 Machine Learning for NLP

In this section, we briefly describe two of the main machine learning algorithms that we use in this thesis. In general, machine learning methods can be divided into three types: 1) *supervised learning*, 2) *unsupervised learning*, and 3) *semi-supervised learning*. In supervised learning, data with output labels are provided to the learner, while in unsupervised learning, the learner aims to learn a pattern from input data without knowing the output labels. Semi-supervised learning is used in scenarios in which some (but not enough) supervision is provided to the learner. Most importantly, semi-supervised learning uses both labeled and unlabeled data. In this chapter, we overview two of the supervised learning methods that we use in this thesis.

2.2.1 Structured Learning with Perceptron

The Perceptron algorithm is one of the earliest machine learning methods that tried to mimic the behavior of the human brain. It was originally proposed by Rosenblatt [1958] as an intuitive way to learn from labeled data with binary labels. There are formal proofs [Novikoff, 1963; Freund and Schapire, 1999] that show this algorithm makes a finite number of mistakes for learning a linearly separable data until it reaches a vector that can perfectly discriminate between examples having the `zero` and `one` labels. In a traditional sense, Perceptron is a binary discriminative classifier that can give linear classifiers. The algorithm is as follows: given a dataset $D = \{(x_1, y_1), \cdots, (x_n, y_n)\}$ such that each data point x_i is a d-dimensional real vector and $y_i \in \{0, 1\}$, Perceptron learns a weight vector $\omega \in \mathbb{R}^d$ and a bias term b, such that:

$$z_i = \mathbb{I}(\omega^\top x_i + b \geq 0)$$

where $\mathbb{I}$ is the indicator function. It starts with a zero weight vector and bias term, and visits every data point once at a time. Whenever the above condition does not hold for z_i, i.e. $z_i \neq y_i$, it updates its weights according to the following equation:

$$\omega = \omega + (y_i - z_i)x_i$$

and

$$b = b + (y_i - z_i)$$

The algorithm converges when no mistake is made by the classifier.

Learning Structures with Perceptron

The Perceptron algorithm is a binary classifier while many problems in natural language processing are multi-label structured prediction. Moreover, the features in natural language problems are categorical: they are words, part-of-speech tags, and other types of string-based features. A standard way to make use of any classifier including the Perceptron algorithm is to convert each feature to a binary indicator feature. For example, if presence of a word is important to us, we can have a separate binary feature for that word. For example, the following feature shows a binary indicator feature for input x and output label y (among all possible labels in $Y = \{y_1, \cdots, y_l\}$) such that this feature is non-zero only if the input x is the word "`going`" and the output label is "`subject`".

$$f_k(x, y) = \begin{cases} 1 & \texttt{if } x = \texttt{going and } y = \texttt{subject} \\ 0 & \texttt{otherwise} \end{cases}$$

where k is the feature index in the binary feature vector. It is worth noting that one can extend this trick to other ways of representing features in data; e.g. joint existence of two words, or a pair of word and tag, or even the count of occurrence of a word, with the expense of making features more complicated and sparser in the final feature vector. Thus, each feature in the sparse feature vector can be defined with arbitrary or even overlapping features.

Collins [2002] proposed a simple but effective way to incorporate multi-label struc-

tured prediction with Perceptron: in the case of using structures, such as dependency parsing, a feature vector of a tree can be defined as the sum of all feature vectors in the structure. For example, for graph-based parsing, each substructure is defined as the arcs of the tree. In other words, a structure y is decomposed into substructures or arcs, where arc $r = (head(i) \rightarrow i)$ gets its own feature vector $f(x, r) \in \mathbb{R}^D$. The final feature vector of the structure can be seen as the sum of the feature vectors:

$$f(x, y) = \sum_{r \in y} f(x, r) \in \mathbb{R}^D$$

Thus the score of a structure y from input x can be defined as multiplication of its feature vector by the weight vector $\omega \in \mathbb{R}^D$:

$$score(y|x; \omega) = \omega^\top f(x, y) = \omega^\top \sum_{r \in y} f(x, r) = \sum_{r \in y} \omega^\top f(x, r)$$

Therefore given an input x, the best structure can be defined as the maximum scoring structure among (usually exponential number of) possible structures.

$$y^* = \arg \max_{y \in \mathcal{Y}(x)} score(y|x; \omega)$$

where $\mathcal{Y}(x)$ is the set of all possible structures. Dynamic programming or beam search is usually used to solve argmax function.

Collins [2002] found that averaging all parameters during all iterations [Freund and Schapire, 1999] gives more reliable parameters in different natural language processing

Inputs: 1) Training data $D = \{(x_1, y_1), \cdots, (x_n, y_n)\}$, 2) Feature function $f(x, y)$ that maps an (x, y) pair to a D-dimensional vector; 3) T: number of training epochs.
Initialization: Set $a\omega_j = 0, \omega_j = 0 \; \forall j \in \{1, \cdots, D\}$.

Algorithm:

```
c = 0
for t = 1 to T do
    for i = 1 to n do
        c = c + 1
        z_i = arg max_{y ∈ Y(x_i)} ω^⊤ f(x_i, y)      ▷ Use dynamic programming or beam search.
        if z_i ≠ y_i then                              ▷ Apply sparse updates.
            for each j where f_j(x_i, y_i) ≠ 0 do
                Set ω_j = ω_j + f_j(x_i, y_i)
                Set aω_j = aω_j + c · f_j(x_i, y_i)
            for each j where f_j(x_i, z_i) ≠ 0 do
                Set ω_j = ω_j − f_j(x_i, z_i)
                Set aω_j = aω_j − c · f_j(x_i, z_i)
aω = ω − aω/c                                          ▷ Calculate averaged weights.
```

Output: $a\omega$.

Figure 2.8: Pseudo-code for the *averaged* structured Perceptron algorithm. The trivial implementation of the averaging trick is not efficient. Daumé III [2006] used the trick in this pseudo-code to make averaging efficient.

tasks:

$$a\omega = \frac{\sum_{i=1}^{T \times n} \omega^{(i)}}{T \times n}$$

The above averaging technique is inherently expensive for sparse feature vectors. Daumé III [2006] propose a simple trick to get the averaged values with less complexity. Figure 2.8 depicts the algorithm with the averaging trick from Daumé III [2006] for structured prediction.

2.2.2 Learning with Artificial Neural Networks

Although linear classifiers have shown promising results in previous work, they are inherently restricted. First and foremost, they are linear classifiers; they work well if the data is

linearly separable. This is not the case in many problems. Second, feature engineering is a cumbersome step in traditional classifiers. Moreover, extracting binary indicator features is not computationally efficient; e.g. He *et al.* [2013] showed that 80% of computation time in a parser with binary indicator features is dedicated to feature computation. In [Bohnet, 2010], within a highly engineered parser, 91% of the computation is dedicated to features. Vector-based features such as word embeddings are good alternatives for binary indicator features but it is not straightforward to incorporate higher order features with embedding vectors (see [Bansal *et al.*, 2014] for a set of parsing experiments with a traditional classifier for a graph-based parser using the word embedding features).

Neural networks, aka deep learning, with several layers of computation can address this problem with the expense of lack of convexity and formal proof of convergence. More importantly, deep models are able to learn representations, even for input features, either from scratch or initialized by pre-trained embedding vectors. This ability has proven to be empirically useful for many natural language processing tasks.

This section describes some of the deep learning models that we use in this thesis.

Feedforward Networks

We start by going back to the definition of parameters in the multi-label Perceptron algorithm (§2.2.1). In the multi-label Perceptron algorithm, for input x and set of l possible labels in $\mathcal{Y}$, we have the following parameters:

- A feature function $f(x, y) \in \mathbb{R}^D$ where $y \in \mathcal{Y}$.
- Output weight matrix $\omega \in \mathbb{R}^{l \times D}$ and bias term $b \in \mathbb{R}^l$. We show the bias term for

each label y as $b_y \in \mathbb{R}$.

The only parameters that are learned during training with Perceptron are the output weight matrix and the bias terms. The score of a label y is calculated as the following:

$$score(y|x;\omega,b) = \omega^\top f(x,y) + b_y$$

In contrast, a multi-class feedforward network has the following parameters [Collins, 2017]:

- A parameter set θ which is a collection of feedforward parameters.
- A function $\phi(x;\theta) \in \mathbb{R}^d$ that maps input x to a *feedforward* representation.
- Output weight matrix $\omega \in \mathbb{R}^{l\times d}$ and bias term $b \in \mathbb{R}^l$. We show the weight vector and the bias term for each label $y \in \mathcal{Y}$ as ω_y and b_y.

The probability of each label is defined by the score of that label normalized by the softmax function:

$$p(y|x;\theta,\omega,b) = \frac{e^{score(y|x;\theta,\omega,b)}}{\sum_{y'\in\mathcal{Y}} e^{score(y'|x;\theta,\omega,b)}}$$

where the score function is defined as:

$$score(y|x;\theta,\omega,b) = \omega_y^\top \phi(x;\theta) + b_y$$

There are two main differences in the feedforward model compared to Perceptron:

1. The feature function $\phi(x; \theta)$ is not dependent on the output label. Thus, we do not need to compute the feature function for every label, thereby, making the computation much faster.
2. The parameters in the feature function θ are arbitrary. It can be any non-linear differentiable function.

Feature function example The following feature function is used in the feedforward transition-based parser of Weiss *et al.* [2015][2]:

- For each configuration, 20 word features, 20 part-of-speech features, and 12 dependency label features are used.
- Each word feature w_i for $i = 1, \cdots, 20$ is embedded with a vector using a word embedding dictionary: $E^{(w)}[w_i] \in \mathbb{R}^{d_w}$.
- Each part-of-speech feature t_i is embedded with a vector using a POS embedding dictionary: $E^{(t)}[t_i] \in \mathbb{R}^{d_t}$.
- Each dependency label feature r_i is embedded with a vector using a dependency label embedding dictionary: $E^{(r)}[r_i] \in \mathbb{R}^{d_r}$.
- All embedding values are updateable with backpropagation. The input is defined as the concatenation of all embedding features:

$$I(\mathcal{S}) = [E^{(w)}[w_1]; \cdots; E^{(w)}[w_{20}]; E^{(t)}[t_1]; \cdots; E^{(t)}[t_{20}]; E^{(r)}[r_1]; \cdots; E^{(r)}[r_{12}]]$$

where $I(\mathcal{S}) \in \mathbb{R}^{d_E}$ and $d_E = 20 \cdot (d_w + d_t) + 12 \cdot d_r$.

[2]This feature function is inspired by Chen and Manning [2014].

- The embedding layer is multiplied by a hidden layer with matrix $H_1 \in \mathbb{R}^{d_{h_1} \times d_E}$ and summed with a hidden bias term $B_1 \in \mathbb{R}^{d_{h_1}}$. The output is activated by the rectified linear unit (RELU) [Nair and Hinton, 2010] function:

$$h_1 = RELU(H_1 I(\mathcal{S}) + B_1) = max(H_1 I(\mathcal{S}) + B_1, 0) \in \mathbb{R}^{d_{h_1}}$$

- The output of the first hidden layer is fed to the second hidden layer with $H_2 \in \mathbb{R}^{d_{h_2} \times d_{h_1}}$ and $B_2 \in \mathbb{R}^{d_{h_2}}$:

$$h_2 = RELU(H_2 \cdot h_1 + B_2) = max(H_2 \cdot h_1 + B_2, 0) \in \mathbb{R}^{d_{h_2}}$$

In summary, the feature function of the model by Weiss *et al.* [2015] is not linear anymore. That leads to a function that models higher levels of interaction between different word, part-of-speech and dependence label features.

Gradient-Based Learning In gradient-based learning, one should define a differentiable loss function, such as the log-likelihood loss function $L(\theta, \omega, b) = -\sum_{i=1}^{n} \log p(y_i|x_i; \theta, \omega, b)$. In stochastic gradient descent (SGD) [Bottou, 2010], one random example or a small *minibatch* of examples are taken into account in each iteration of training. There are two main benefits to this strategy: first, the model can be scaled to large datasets; and second, the model empirically converges faster than computing the loss with respect to all of the training data instances. With single-layer models, calculating the derivatives is straightforward, but in cases where the number of layers and activation functions is more than one and the layers have shared sources (such as word

Inputs: 1) Training data $D = \{(x_1, y_1), \cdots, (x_n, y_n)\}$, 2) Feedforward feature function $\phi(x; \theta)$; 3) T: number of training epochs; and 4) a sequence of learning rates $\eta^1, \cdots, \eta^T$.
Initialization: Initialize parameters θ, ω, and b randomly.

Algorithm:
for t = 1 to T **do**
Sample a minibatch $x_{r_1}, \cdots, x_{r_b}$ drawn from b random integers $\{r_i\}_{i=1}^{b}$ uniformly from $\{1 \cdots n\}$.
Set $L(\theta, \omega, b) = \frac{-\sum_{i=1}^{b} \log p(y_i | x_{r_i}; \theta, \omega, b)}{b}$
for each parameter $\theta_j \in \theta$ **do**
Set $\theta_j = \theta_j - \eta^{(t)} \cdot \frac{dL(\theta,\omega,b)}{d\theta_j}$
for each label $y \in \mathcal{Y}$ **do**
Set $\omega_y = \omega_y - \eta^{(t)} \cdot \frac{dL(\theta,\omega,b)}{d\omega_y}$
Set $b_y = b_y - \eta^{(t)} \cdot \frac{dL(\theta,\omega,b)}{db_y}$

Output: θ, ω, and b .

Figure 2.9: Pseudo-code for the *minibatched* stochastic gradient descent (SGD) algorithm with the log-likelihood objective function.

embedding features), calculating the derivatives is not straightforward. The backpropagation algorithm tries to solve this problem by defining the notion of *computation graph*. We leave details of computation graph: for more information, see Goldberg [2017]. Figure 2.9 shows the pseudo-code of the SGD algorithm with the log-likelihood function.

Recurrent Neural Networks

One drawback of feedforward networks is their insensitivity to long-distance contextual dependencies in sequential data such as sentences. For example, a word in a sentence might convey different senses in different contexts. Besides polysemy, a word with one sense might have a different syntactic role depending on its position in a sentence.

Recurrent neural networks (RNNs) define a generic recurrent function that converts a sequence $x_1, \cdots, x_n$ to a sequence of vectors $h_1, \cdots, h_n$ where $h_i \in \mathbb{R}^h$ is the recurrent

representation for the ith position. The RNN function gets the recurrent representation from the previous position and the current input, and outputs the vector representation of the current position:

$$h_i = RNN(x_i, h_{i-1}; \theta)$$

where θ is the set of parameters in the recurrent network. The backpropagation algorithm can be applied through time steps.

Long short-term memories (LSTM) Long short-term memories (LSTM) [Hochreiter and Schmidhuber, 1997] is a variant of recurrent neural networks that have recently become very popular. In each state, a forget cell, an input cell, and an output cell are calculated to decide which of the previous inputs should be ignored and which ones should be kept for the next step. The original RNN function involves multiplication while LSTM involves entrywise products and summations. Thus LSTM does not have the vanishing gradient problem in the original RNN function. LSTM has many variants; the following is one of the variants of the LSTM function [Goldberg, 2017]:

$$\texttt{Forget gate}: f = \sigma(x_j W^{x^f} + h_{j-1}^\top W^{h^f})^\top$$

$$\texttt{Input gate}: i = \sigma(x_j W^{x^i} + h_{j-1}^\top W^{h^i})^\top$$

$$z = tanh(x_j W^{x^z} + h_{j-1}^\top W^{h^z})^\top$$

$$\texttt{Memory cell}: c_j = f \odot c_{j-1} + i \odot tanh(z)$$

$$\texttt{Output gate}: o = \sigma(x_j W^{x^o} + h_{j-1}^\top W^{h^o})^\top$$

$$\texttt{LSTM representation}: h_j = o \odot tanh(c_j)$$

where $\odot$ is the entrywise product, and σ is the sigmoid function, the following dimensions are used for the matrices:

$$x_j \in \mathbb{R}^{d_x}$$

$$c_j, h_j, i, f, o, z \in \mathbb{R}^{d_h}$$

$$W^{x^i}, W^{x^f}, W^{x^o}, W^{x^z} \in \mathbb{R}^{d_x \times d_h}$$

$$W^{h^i}, W^{h^f}, W^{h^o}, W^{h^z} \in \mathbb{R}^{d_h \times d_h}$$

Birdirectional Deep RNNs In many applications, it is more effective to read a sequence both from right to left and left to right. In other words, two different RNNs are used to model sequences one from left to right and the other from right to left.

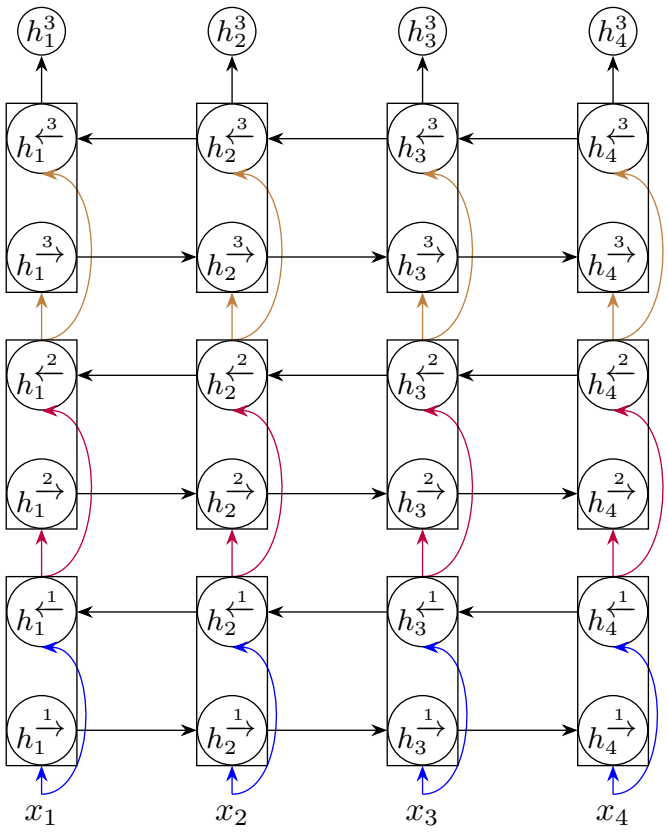


Figure 2.10: Example of a deep BiRNN for a input sentence with four words. There are 3 layers of bidirectional RNNs in this example..

$$h_j = BiRNN(x_{1:n}, j; \theta^{\rightarrow}, \theta^{\leftarrow}) = [h_j^{\rightarrow}; h_j^{\leftarrow}]$$

$$h_j^{\rightarrow} = RNN(x_j, h_{j-1}^{\rightarrow}; \theta^{\rightarrow})$$

$$h_j^{\leftarrow} = RNN(x_j, h_{j+1}^{\leftarrow}; \theta^{\leftarrow})$$

Moreover, it is shown that one can stack different layers of RNN or BiRNNs to obtain

a deeper representation of a sequence:

$$h_j^n = BiRNN(h_{1:n}^{n-1}, j; \theta^{\overrightarrow{n}}, \theta^{\overleftarrow{n}})$$

$$\ldots$$

$$h_j^2 = BiRNN(h_{1:n}^1, j; \theta^{\overrightarrow{2}}, \theta^{\overleftarrow{2}})$$

$$h_j^1 = BiRNN(x_{1:n}, j; \theta^{\overrightarrow{1}}, \theta^{\overleftarrow{1}})$$

Figure 2.10 shows a simple example of deep bidirectional recurrent neural network model.

Chapter 3

Related Work

This chapter briefly overview related work for dependency parsing and sentiment analysis.

3.1 Cross-Lingual Transfer of Dependency Parsers

There has recently been a great deal of work on syntactic transfer. As mentioned in 1.4, there are two main categories of transfer methods: annotation projection and direct transfer.

3.1.1 Annotation Projection for Parsing

The annotation projection approach, where dependencies from one language are transferred through translation alignments to another language, has been considered by several authors [Hwa *et al.*, 2005; Ganchev *et al.*, 2009; McDonald *et al.*, 2011; Ma and Xia, 2014; Lacroix *et al.*, 2016; Agić *et al.*, 2016; Schlichtkrull and Søgaard, 2017]. This prior work involves various innovations such as the use of posterior regularization [Ganchev *et al.*, 2009], the use of entropy regularization and parallel guidance [Ma and Xia, 2014], and a method for training on partial annotations that are projected from source to target language [Spreyer and Kuhn, 2009].

3.1.2 Direct Syntactic Transfer

A number of methods [Zeman and Resnik, 2008; McDonald *et al.*, 2011; Cohen *et al.*, 2011; Naseem *et al.*, 2012; Täckström *et al.*, 2013; Rosa and Zabokrtsky, 2015] directly learn delexicalized models that can be trained on universal treebank data from one or more source languages, then applied to the target language. More recent work has introduced cross-lingual representations—for example cross-lingual word-embeddings—that can be used to improve performance [Zhang and Barzilay, 2015; Guo *et al.*, 2015; Duong *et al.*, 2015a; Duong *et al.*, 2015b; Guo *et al.*, 2016; Ammar *et al.*, 2016b]. These cross-lingual representations are usually learned from parallel translation data.

A number of authors have considered incorporating universal syntactic properties, such as dependency order, by selectively learning syntactic attributes from similar source languages or learning universal rules [Naseem *et al.*, 2012; Täckström *et al.*, 2013; Zhang and Barzilay, 2015; Ammar *et al.*, 2016a]. More recently, Wang and Eisner [2016] have developed a synthetic treebank as a universal treebank to help learn parsers for new languages. Martínez Alonso *et al.* [2017] try a very different approach in cross-lingual transfer by using a ranking approach.

A number of authors [Täckström *et al.*, 2012; Guo *et al.*, 2015; Guo *et al.*, 2016] have introduced methods that learn cross-lingual representations that are then used in syntactic transfer. Most of these approaches introduce constraints to a clustering or embedding algorithm that encourage words that are translations of each other to have similar representations.

One approach would be to use hand-crafted translation lexicons, for example, Pan-

Lex [Baldwin *et al.*, 2010] (e.g. see Duong *et al.* [2015b]), which covers 1253 language varieties, Google translate (e.g., see Ammar *et al.* [2016c]), or Wiktionary (e.g., see Durrett *et al.* [2012] for an approach that uses Wiktionary for cross-lingual transfer). These resources are potentially very rich sources of information.

3.1.3 Treebank Translation

Recent work [Tiedemann *et al.*, 2014; Tiedemann and Agić, 2016] has considered treebank translation, where a statistical machine translation system (e.g., MOSES [Koehn *et al.*, 2007]) is used to translate a source language treebank into the target language, complete with reordering of the input sentence. Practically, a reliable machine translation system needs a large amount of parallel data. If this assumption holds, one might be able to use it for dependency parsing. Previous work on treebank translation [Tiedemann *et al.*, 2014; Tiedemann and Agić, 2016] gives a lower performance than annotation projection. The lexicalization approach described in our work is a simple form of treebank translation, where we use a word-to-word translation model. In spite of its simplicity, it is an effective approach.

3.1.4 Unsupervised Parsing

There has been a huge amount of work on purely unsupervised methods for learning dependency syntax [Klein and Manning, 2004; Smith and Eisner, 2005; Headden III *et al.*, 2009; Cohen and Smith, 2009; Berg-Kirkpatrick and Klein, 2010; Naseem *et al.*, 2010; Gillenwater *et al.*, 2010; Brody, 2010; Spitkovsky *et al.*, 2011b; Mareček and Straka, 2013;

Spitkovsky *et al.*, 2013; Cerisara *et al.*, 2013; Jiang *et al.*, 2016; Qiao *et al.*, 2016; Cai *et al.*, 2017]. These methods are interesting in their own right, as a model of how natural language syntax can be acquired in an unsupervised fashion; however, their performance is some way below the performance of methods using cross-lingual transfer. For example, the results of Spitkovsky *et al.* [2013] are 64.4% accuracy on section 23 of the Wall Street Journal treebank; results on the German, Italian, Portuguese and Spanish CONLL data sets are 56.2%, 56.8%, 74.5%, and 61.7% respectively (average over these 4 languages is 62.3%). This is compared to an average accuracy of 72.2% for the approach of Ma and Xia [2014] on these 4 languages on the CONLL datasets, and an average accuracy of 76.05% on the universal treebank. Some of this prior work [Naseem *et al.*, 2010; Rasooli and Faili, 2012] has made use of linguistic universals in the induction of unsupervised models: the work of Naseem *et al.* [2010] reports accuracy of 66.1% on dependency parsing of sentences of length 20 words or less, again significantly lower than the results of the annotation projection model of Ma and Xia [2014].

3.2 Cross-Lingual Transfer for Sentiment Analysis

While the vast majority of work in sentiment analysis has focused on English [Hu and Liu, 2004; Liu, 2012b; Socher *et al.*, 2013], there have been many studies on cross-lingual analysis and transfer across languages.

3.2.1 Machine Translation for Sentiment Transfer

Cross-lingual sentiment systems have mostly focused on projection methods using in-domain corpora [Mihalcea *et al.*, 2007], and on machine translation methods which either

translate the target language text into a resource-rich language such as English and apply an English sentiment model [Wan, 2008; Salameh *et al.*, 2015], or translate the English labeled data into the target language and build a target-language system [Balahur and Turchi, 2014]. Recent neural network approaches include that of Zhou *et al.* [2016a], who developed a hierarchical attention model by translating the training data into the target language and modeling both the source and target side using a bidirectional LSTM, and Zhou *et al.* [2015] who also translated the source training data and used denoising autoencoders to create bilingual embeddings incorporating sentiment information from labeled data and their translations. These approaches rely on machine translation of in-domain text and would unlikely be applicable in a low resource setting where in-domain parallel data or high quality MT are not available. Previous work has also focused on developing language-specific systems for a new language, e.g. Mukund and Srihari [2010]. for Urdu and and Joshi *et al.* [2010] for Hindi; these methods when available are beneficial to augment cross-lingual techniques.

Duh *et al.* [2011] has argued that even if perfect machine translation were available, it would not be a complete solution for the cross-lingual sentiment task, for if that were viewed as a domain adaptation task, machine translation systems would produce "domain-mismatch" between vocabulary distributions in the original and translated data, with limited sentiment vocabulary in the translated data.

3.2.2 Direct Sentiment Transfer

Meng *et al.* [2012] avoided the use of an MT system by instead developing a cross-lingual mixture model which maximizes the likelihood of generating a bilingual Chinese-English parallel corpus and determining word generation probabilities for the sentiment classes, where labeled data in the target language need not be available. This approach is complementary to our direct transfer method. Instead of using a cross-lingual mixture model, we train systems directly on source language data using cross-lingual word vector representations. As with Meng *et al.* [2012], we utilize parallel data. In our case, the parallel data is used either to create automatic dictionaries and cross-lingual embeddings, or for annotation projection. Moreover, we explore many source-target language pairs as well as parallel corpora from religious text that is out-of-domain but more likely to be available for under-resourced languages.

It is worth noting that there are a few complementary methods that are similar to the direct transfer model without using machine translation. For example Zhou *et al.* [2014] used a bilingual corpus and stacked autoencoders to create shared sentence representations for Chinese and English sentences from which sentiment models were trained directly on English labels. Chen *et al.* [2016] trained a direct transfer model using an adverserial deep averaging network, whereby the model tries to create language-invariant bilingual representations that are not recognizable by a language predictor. Unlike these approaches, our strategy leverages data from different source languages and parallel data from both in-domain and religious text, and our projection method applies a new configuration of majority voting when more than one translation is available for a target

language sentence.

Part I

Cross-Lingual Transfer of Dependency Parsers

Chapter 4

Density-Driven Cross-Lingual Transfer of Dependency Parsers

4.1 Introduction

In recent years there has been a great deal of interest in dependency parsing models for natural languages. Supervised learning methods have been shown to produce highly accurate dependency-parsing models; unfortunately, these methods rely on human-annotated data, which is expensive to obtain, leading to a significant barrier to the development of dependency parsers for new languages. Recent work has considered unsupervised methods (e.g. [Klein and Manning, 2004; Headden III *et al.*, 2009; Gillenwater *et al.*, 2011; Mareček and Straka, 2013; Spitkovsky *et al.*, 2013; Le and Zuidema, 2015; Grave and Elhadad, 2015]), or methods that transfer linguistic structures across languages (e.g. [Cohen *et al.*, 2011; McDonald *et al.*, 2011; Ma and Xia, 2014; Tiedemann, 2015; Guo *et al.*, 2015; Zhang and Barzilay, 2015; Xiao and Guo, 2015]), in an effort to reduce or eliminate the need for annotated training examples. Unfortunately the accuracy of these methods generally lags quite substantially behind the performance of fully supervised approaches.

This chapter describes novel methods for the transfer of syntactic information between languages. As in previous work [Hwa *et al.*, 2005; Ganchev *et al.*, 2009; McDonald *et*

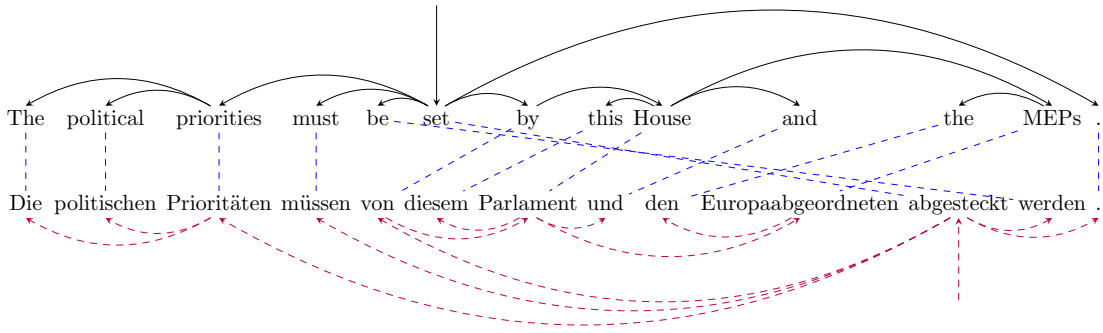


Figure 4.1: An example projection from English to German in the EuroParl data [Koehn, 2005]. The English parse tree is the output from a supervised parser, while the German parse tree is projected from the English parse tree using translation alignments from GIZA++.

al., 2011; Ma and Xia, 2014], our goal is to induce a dependency parser in a target language of interest without any direct supervision (i.e., a treebank) in the target language: instead we assume access to parallel translations between the target and one or more source languages, and to supervised parsers in the source languages. We can then use alignments induced using tools such as GIZA++ [Och and Ney, 2003], to transfer dependencies from the source language(s) to the target language (example projections are shown in Figure 4.1). A target language parser is then trained on the projected dependencies.

Our contributions are as follows:

- We demonstrate the utility of *dense* projected structures when training the target-language parser. In the most extreme case, a "dense" structure is a sentence in the target language where the projected dependencies form a fully projective tree that includes all words in the sentence (we will refer to these structures as "full" trees). In more relaxed definitions, we might include sentences where at least some proportion (e.g., 80%) of the words participate as a modifier in some dependency,

or where long sequences (e.g., 7 words or more) of words all participate as modifiers in some dependency. We give empirical evidence that dense structures give particularly high accuracy for their projected dependencies.

- We describe a training algorithm that builds on the definitions of dense structures. The algorithm initially trains the model on full trees, then iteratively introduces increasingly relaxed definitions of density. The algorithm makes use of a training method that can leverage partial (incomplete) dependency structures, and also makes use of confidence scores from a perceptron-trained model.

In spite of the simplicity of our approach, our experiments demonstrate significant improvements in accuracy over previous work. In experiments on transfer from a single source language (English) to a single target language (German, French, Spanish, Italian, Portuguese, and Swedish), our average dependency accuracy is 78.89%. When using multiple source languages, average accuracy is improved to 82.18%. This is a 5.51% absolute improvement over the previous best results reported on this data set, 76.67% for the approach of Ma and Xia [2014]. To give another perspective, our accuracy is close to that of the fully supervised approach of McDonald *et al.* [2005b], which gives 84.29% accuracy on this data. To the best of our knowledge these are the highest accuracy parsing results for an approach that makes no use of treebank data for the language of interest.

4.2 Our Approach

This section describes our approach, giving definitions of parallel data and of dense projected structures; describing preliminary exploratory experiments on transfer from

German to English; describing the iterative training algorithm used in our work; and finally describing a generalization of the method to transfer from multiple languages.

4.2.1 Parallel Data Definitions

We assume that we have parallel data in two languages. The source language, for which we have a supervised parser, is assumed to be English. The target language, for which our goal is to learn a parser, will be referred to as the "foreign" language. We describe the generalization to more than two languages in §4.2.5.

We use the following notation. Our parallel data is a set of examples $(e^{(k)}, f^{(k)})$ for $k = 1 \dots n$, where each $e^{(k)}$ is an English sentence, and each $f^{(k)}$ is a foreign sentence. Each $e^{(k)} = e_1^{(k)} \dots e_{s_k}^{(k)}$ where $e_i^{(k)}$ is a word, and s_k is the length of k'th source sentence. Similarly, $f^{(k)} = f_1^{(k)} \dots f_{t_k}^{(k)}$ where $f_j^{(k)}$ is a word, and t_k is the length of k'th foreign sentence.

A **dependency** is a four-tuple (l, k, h, m) where $l \in \{e, f\}$ is the language, k is the sentence number, h is the head index, m is the modifier index. Note that if $l = e$ then we have $0 \leq h \leq s_k$ and $1 \leq m \leq s_k$, conversely if $l = f$ then $0 \leq h \leq t_k$ and $1 \leq m \leq t_k$. We use $h = 0$ when h is the root of the sentence.

For any $k \in \{1 \dots n\}$, $j \in \{0 \dots t_k\}$, $A_{k,j}$ is an integer specifying which word in $e_1^{(k)} \dots e_{s_k}^{(k)}$, word $f_j^{(k)}$ is aligned to. It is `NULL` if $f_j^{(k)}$ is not aligned to anything. We have $A_{k,0} = 0$ for all k: that is, the root in one language is always aligned to the root in the other language.

In our experiments, we use intersected alignments from GIZA++ [Och and Ney, 2003]

to provide the $A_{k,j}$ values.

4.2.2 Projected Dependencies

We now describe various sets of projected dependencies. We use $\mathcal{D}$ to denote the set of all dependencies in the source language: these dependencies are the result of parsing the English side of the translation data using a supervised parser. Each dependency $(l, k, h, m) \in \mathcal{D}$ is a four-tuple as described above, with $l = e$. We will use $\mathcal{P}$ to denote the set of all projected dependencies from the source to target language. The set $\mathcal{P}$ is constructed from $\mathcal{D}$ and the alignment variables $A_{k,j}$ as follows:

$$\mathcal{P} = \{(l, k, h, m) : l = f \wedge \ (e, k, A_{k,h}, A_{k,m}) \in \mathcal{D}\}$$

We say the k'th sentence receives a *full* parse under the dependencies $\mathcal{P}$ if the dependencies (f, k, h, m) for k form a projective tree over the entire sentence: that is, each word has exactly one head, the root symbol is the head of the entire structure, and the resulting structure is a projective tree. We use $\mathcal{T}_{100} \subseteq \{1 \dots n\}$ to denote the set of all sentences that receive a full parse under $\mathcal{P}$. We then define the following set:

$$\mathcal{P}_{100} = \{(l, k, h, m) \in \mathcal{P} : k \in \mathcal{T}_{100}\}$$

We say the k'th sentence receives a *dense* parse under the dependencies $\mathcal{P}$ if the dependencies of the form (f, k, h, m) for k form a projective tree over at least 80% of the words in the sentence. We use $\mathcal{T}_{80} \subseteq \{1 \dots n\}$ to denote the set of all sentences that

receive a dense parse under $\mathcal{P}$. We then define the following set,

$$\mathcal{P}_{80} = \{(l, k, h, m) \in \mathcal{P} : k \in \mathcal{T}_{80}\}$$

We say the k'th sentence receives a *span-s* parse where s is an integer if there is a sequence of at least s consecutive words in the target language that are all seen as a modifier in the set $\mathcal{P}$. We use $\mathcal{S}_s$ to refer to the set of all sentences with a span-s parse. We define the sets

$$\mathcal{P}_{\geq 7} = \{(l, k, h, m) \in \mathcal{P} : k \in \mathcal{S}_7\}$$

$$\mathcal{P}_{\geq 5} = \{(l, k, h, m) \in \mathcal{P} : k \in \mathcal{S}_5\}$$

$$\mathcal{P}_{\geq 1} = \{(l, k, h, m) \in \mathcal{P} : k \in \mathcal{S}_1\}$$

Finally, we also create datasets that only include projected dependencies that are consistent with respect to part-of-speech (POS) tags for the head and modifier words in source and target data. We assume a function $\text{POS}(k, j, i)$ which returns `TRUE` if the POS tags for words $f_j^{(k)}$ and $e_i^{(k)}$ are consistent. The definition of POS-consistent projected dependencies is then as follows:

$$\bar{\mathcal{P}} = \{(l, k, h, m) \in \mathcal{P} : \text{POS}(k, h, A_{k,h}) \wedge \text{POS}(k, m, A_{k,m})\}$$

We experiment with two definitions for the POS function. The first imposes a hard constraint, that the POS tags in the two languages must be identical. The second imposes a soft constraint, that the two POS tags must fall into the same equivalance class: the

POS Constraints	$\mathcal{P}$		$dense$		$\mathcal{P}_{100}$		Train on $\mathcal{P}_{100}$
	#sen	Acc.	#sen	Acc.	#sen	Acc.	
No Restriction	$968k$	74.0	$65k$	81.4	$23k$	83.0	69.5
Hard match	$927k$	80.1	$26k$	88.0	$8k$	90.1	68.0
Soft match	$904k$	80.0	$52k$	84.9	$18k$	85.8	70.6

Table 4.1: Statistics showing the accuracy for various definitions of projected trees: see §4.2.2 for definitions of $\mathcal{P}$, $\mathcal{P}_{100}$ etc. Columns labeled "Acc." show accuracy when the output of a supervised German parser is used as gold standard data. Columns labeled "#sen" show number of sentences. "dense" shows $\mathcal{P}_{100} \cup \mathcal{P}_{80} \cup \mathcal{P}_{\geq 7}$ and "Train" shows accuracy on test data of a model trained on the $\mathcal{P}_{100}$ trees.

equivalence classes used are listed in §4.3.1.

Given this definition of $\bar{\mathcal{P}}$, we can create sets $\bar{\mathcal{P}}_{100}$, $\bar{\mathcal{P}}_{80}$, $\bar{\mathcal{P}}_{\geq 7}$, $\bar{\mathcal{P}}_{\geq 5}$, and $\bar{\mathcal{P}}_{\geq 1}$, using analogous definitions to those given above.

4.2.3 Preliminary Experiments with Transfer from English to German

Throughout the experiments in this chapter, we used German as the target language for development of our approach. Table 4.1 shows some preliminary results on transferring dependencies from English to German. We can estimate the accuracy of dependency subsets such as $\mathcal{P}_{100}$, $\mathcal{P}_{80}$, $\mathcal{P}_{\geq 7}$ and so on by comparing these dependencies to the dependencies from a supervised German parser on the same data. That is, we use a supervised parser to provide gold standard annotations. The full set of dependencies $\mathcal{P}$ give 74.0% accuracy under this measure; results for $\mathcal{P}_{100}$ are considerably higher in accuracy, ranging from 83.0% to 90.1% depending on how POS constraints are used.

As a second evaluation method, we can test the accuracy of a model trained on the $\mathcal{P}_{100}$ data. The benefit of the soft-matching POS definition is clear. The hard match definition harms performance, presumably because it reduces the number of sentences used to train

Inputs: Sets $\mathcal{P}_{100}$, $\mathcal{P}_{80}$, $\mathcal{P}_{\geq 7}$, $\mathcal{P}_{\geq 5}$, $\mathcal{P}_{\geq 1}$ as defined in §4.2.2

Algorithm:

$\theta^1 = \text{Train}(\mathcal{P}_{100})$
$\mathcal{P}^1_{100} = \text{CDecode}(\mathcal{P}_{80} \cup \mathcal{P}_{\geq 7}, \theta^1)$
$\theta^2 = \text{Train}(\mathcal{P}_{100} \cup \text{Top}(\mathcal{P}^1_{100}, \theta^1))$
$\mathcal{P}^2_{100} = \text{CDecode}(\mathcal{P}_{80} \cup \mathcal{P}_{\geq 5}, \theta^2)$
$\theta^3 = \text{Train}(\mathcal{P}_{100} \cup \text{Top}(\mathcal{P}^2_{100}, \theta^2))$
$\mathcal{P}^3_{100} = \text{CDecode}(\mathcal{P}_{\geq 1}, \theta^3)$
$\theta^4 = \text{Train}(\mathcal{P}_{100} \cup \text{Top}(\mathcal{P}^3_{100}, \theta^3))$

Output: Parameter vectors $\theta^1, \theta^2, \theta^3, \theta^4$.

Figure 4.2: The learning algorithm of the density-driven method.

the model.

Throughout the rest of this chapter, we use the soft POS constraints in all projection algorithms.[1]

4.2.4 The Training Procedure

We now describe the training procedure used in our experiments. We use a Perceptron-trained shift-reduce parser, similar to that of Zhang and Nivre [2011]. We assume that the parser is able to operate in a "constrained" mode, where it returns the highest scoring parse that is consistent with a given subset of dependencies. This can be achieved via zero-cost dynamic oracles [Goldberg and Nivre, 2013].

We assume the following definitions:

- Train($\mathcal{D}$) is a function that takes a set of dependency structures $\mathcal{D}$ as input, and returns a model θ as its output. The dependency structures are assumed to be full

[1] The hard constraint is also used by Ma and Xia [2014].

trees: that is, they correspond to fully projected trees with the root symbol as their root.

- CDecode$(\mathcal{P}, \theta)$ is a function that takes a set of partial dependency structures $\mathcal{P}$, and a model θ as input, and as output returns a set of full trees $\mathcal{D}$. It achieves this by constrained decoding of the sentences in $\mathcal{P}$ under the model θ, where for each sentence we use beam search to search for the highest scoring projective full tree that is consistent with the dependencies in $\mathcal{P}$.
- Top$(\mathcal{D}, \theta)$ takes as input a set of full trees $\mathcal{D}$, and a model θ. It returns the top m highest scoring trees in $\mathcal{D}$ (in our experiments we used $m = 200,000$), where the score for each tree is the Perceptron-based score normalized by the sentence length. Thus we return the 200,000 trees that the Perceptron is most confident on. In cases where $|\mathcal{D}| < m$, the entire set $\mathcal{D}$ is returned.

Figure 4.2 shows the learning algorithm. It generates a sequence of parsing models, $\theta^1 \ldots \theta^4$. In the first stage of learning, the model is initialized by training on $\mathcal{P}_{100}$. The method then uses this model to fill in the missing dependencies on $\mathcal{P}_{80} \cup \mathcal{P}_{\geq 7}$ using the CDecode method; this data is added to $\mathcal{P}_{100}$ and the model is retrained. The method is iterated, at each point adding in additional partial structures (note that $\mathcal{P}_{\geq 7} \subseteq \mathcal{P}_{\geq 5} \subseteq \mathcal{P}_{\geq 1}$, hence at each stage we expand the set of training data that is parsed using CDecode).

4.2.5 Generalization to Multiple Languages

We now consider the generalization to learning from multiple languages. We again assume that the task is to learn a parser in a single target language, for example German.

We assume that we now have multiple source languages. For example, in our experiments with German as the target, we used English, French, Spanish, Portuguese, Swedish, and Italian as source languages. We assume that we have fully supervised parsers for all source languages. We will consider two methods for combining information from the different languages:

Method 1: Concatenation In this approach, we form sets $\mathcal{P}$, $\mathcal{P}_{100}$, $\mathcal{P}_{80}$, $\mathcal{P}_{\geq 7}$ etc. from each of the languages separately, and then concatenate the data to give new definitions of $\mathcal{P}$, $\mathcal{P}_{100}$,$\mathcal{P}_{80}$, $\mathcal{P}_{\geq 7}$ etc. That is, dependency structures projected from different languages are taken to be entirely separate from each other.

Method 2: Voting In this case, we assume that each target language sentence is aligned to a source language sentence in each of the source languages. This is the case, for example, in the Europarl data, where we have translations of the same material into multiple languages. We can then create the set $\mathcal{P}$ of projected dependencies using a voting scheme. For any word (k, j) seen in the target language, each source language will identify a headword (this headword may be NULL if there is no alignment giving a dependency). We simply take the most frequent headword chosen by the languages. After creating the set $\mathcal{P}$, we can create subsets such as $\mathcal{P}_{100}$, $\mathcal{P}_{80}$, $\mathcal{P}_{\geq 7}$ in exactly the same way as before.

Once the various projected dependency training sets have been created, we train the dependency parsing model using the algorithm given in §4.2.4.

L	en→trgt				concat→trgt				voting→trgt			
	θ^1	θ^2	θ^3	θ^4	θ^1	θ^2	θ^3	θ^4	θ^1	θ^2	θ^3	θ^4
de	70.56	72.86	73.74	74.32	73.47	75.17	75.59	76.34	78.17	79.29	79.36	79.68
es	75.69	77.27	77.29	78.17	79.53	79.57	79.67	80.28	79.82	80.76	81.16	80.86
fr	77.03	78.54	78.70	79.91	81.23	81.79	82.30	82.24	82.17	82.75	82.47	82.72
it	77.35	78.64	79.06	79.46	81.49	82.25	82.02	82.49	82.58	82.95	83.45	83.67
pt	75.98	77.96	78.29	79.38	80.29	81.73	81.53	82.23	80.12	81.70	81.69	82.07
sv	78.68	80.28	80.81	82.11	82.53	83.78	83.83	83.80	82.85	83.76	83.85	84.06
avg	75.88	77.59	77.98	78.89	79.76	80.72	80.82	81.23	80.95	81.87	82.00	82.18

Table 4.2: Parsing accuracies of different methods on the test data using the gold standard POS tags. The models $\theta^1 \ldots \theta^4$ are described in §4.2.4. "en→trgt" is the single-source setting with English as the source language. "concat→trgt" and "voting→trgt" are results with multiple source languages for the concatenation and voting methods.

4.3 Experiments

We now describe experiments using our approach. We first describe data and tools used in the experiments, and then describe results.

4.3.1 Data and Tools

Data We use the EuroParl data [Koehn, 2005] as our parallel data and the Google universal treebank (v2; standard data) [McDonald *et al.*, 2013] as our evaluation data, and as our training data for the supervised source-language parsers. We use seven languages that are present in both Europarl and the Google universal treebank: English (used only as the source language), and German, Spanish, French, Italian, Portuguese and Swedish.

Word Alignments We use Giza++[2] [Och and Ney, 2003] to induce word alignments. Sentences with length greater than 100 and single-word sentences are removed from the parallel data. We follow common practice in training Giza++ for both translation direc-

[2] `http://www.statmt.org/moses/giza/GIZA++.html`

tions, and taking the intersection of the two sets as our final alignment. Giza++ default alignment model is used in all of our experiments.

The Parsing Model For all parsing experiments we use the Yara parser[3] [Rasooli and Tetreault, 2015], a reimplementation of the k-beam arc-eager parser of Zhang and Nivre [2011]. We use a beam size of 64, and Brown clustering features[4] [Brown *et al.*, 1992; Liang, 2005]. The parser gives performance close to the state of the art: for example on section 23 of the Penn WSJ treebank [Marcus *et al.*, 1993], it achieves 93.32% accuracy, compared to 92.9% accuracy for the parser of [Zhang and Nivre, 2011].

POS Consistency As mentioned in §4.2.2, we define a soft POS consistency constraint to prune some projected dependencies. A source/target language word pair satisifies this constraint if one of the following conditions hold: 1) the POS tags for the two words are identical; 2) the word forms for the two words are identical (this occurs frequently for numbers, for example); 3) both tags are in one of the following equivalence classes: {ADV $\leftrightarrow$ ADJ} {ADV $\leftrightarrow$ PRT} {ADJ $\leftrightarrow$ PRON} {DET $\leftrightarrow$ NUM} {DET $\leftrightarrow$ PRON} {DET $\leftrightarrow$ NOUN} {PRON $\leftrightarrow$ NOUN} {NUM $\leftrightarrow$ X} {X $\leftrightarrow$.}. These rules were developed primarily on German, with some additional validation on Spanish. These rules required a small amount of human engineering, but we view this as relatively negligible.

Parameter Tuning We used German as a target language in the development of our approach, and in setting hyper-parameters. The parser is trained using the averaged

[3] `https://github.com/yahoo/YaraParser`

[4] `https://github.com/percyliang/brown-cluster`

Model	$en \rightarrow trgt$	$concat$	$voting$	$sup(1st)$	$sup(ae)$
de	73.01	74.70	78.77	80.29	84.25
es	76.31	78.33	79.17	82.17	84.66
fr	77.54	79.71	80.77	81.33	84.95
it	78.14	80.82	82.03	83.90	87.03
pt	78.14	80.81	80.67	84.80	88.08
sv	79.31	80.81	82.03	81.12	84.87
avg	77.08	79.20	80.57	82.27	85.64

Table 4.3: Parsing results with automatic part of speech tags on the test data. Sup (1st) is the supervised first-order dependency parser [McDonald *et al.*, 2005b] and sup (ae) is the Yara arc-eager parser [Rasooli and Tetreault, 2015].

structured perceptron algorithm [Collins, 2002] with max-violation updates [Huang *et al.*, 2012]. The number of iterations over the training data is 5 when training model θ^1 in any setting, and 2, 1 and 4 when training models $\theta^2, \theta^3, \theta^4$ respectively. These values are chosen by observing the performance on German. We use θ^4 as the final output from the training process: this is found to be optimal in English to German projections.

4.3.2 Results

This section gives results of our approach for the single source, multi-source (concatenation) and multi-source (voting) methods. Following previous work [Ma and Xia, 2014], we use gold-standard part-of-speech (POS) tags on test data. We also provide results with automatic POS tags.

Results with a Single Source Language The first set of results are with a single source language; we use English as the source in all of these experiments. Table 4.2 shows the accuracy of parameters $\theta^1 \ldots \theta^4$ for transfer into German, Spanish, French, Italian, Portuguese, and Swedish. Even the lowest performing model, θ^1, which is trained only on

Model	ge15	zb15	zb_s15	mph11	mx14	$en \rightarrow trgt$	$concat$	$voting$	$sup(1st)$	$sup(ae)$
de	51.0	62.5	74.2	69.77	74.30	$74.32_{(+0.02)}$	$76.34_{(+2.04)}$	$79.68_{(+5.38)}$	81.65	85.34
es	59.2	78.0	78.4	68.72	75.53	$78.17_{(+2.64)}$	$80.28_{(+4.75)}$	$80.86_{(+5.33)}$	83.92	86.69
fr	59.0	78.9	79.6	73.13	76.53	$79.91_{(+3.38)}$	$82.24_{(+5.71)}$	$82.72_{(+6.19)}$	83.51	86.24
it	55.6	79.3	80.9	70.74	77.74	$79.46_{(+1.72)}$	$82.49_{(+4.75)}$	$83.67_{(+5.93)}$	85.47	88.83
pt	57.0	78.6	79.3	69.82	76.65	$79.38_{(+2.73)}$	$82.23_{(+5.58)}$	$82.07_{(+5.42)}$	85.67	89.44
sv	54.8	75.0	78.3	75.87	79.27	$82.11_{(+2.84)}$	$83.80_{(+4.53)}$	$84.06_{(+4.79)}$	85.59	88.06
avg	56.1	75.4	78.4	71.34	76.67	$78.89_{(+2.22)}$	$81.23_{(+4.56)}$	$82.18_{(+5.51)}$	84.29	87.50

Table 4.4: Comparison to previous work: ge15 [Grave and Elhadad, 2015, Figure 4], zb15 [Zhang and Barzilay, 2015], zb_s15 [Zhang and Barzilay, 2015, semi-supervised with 50 annotated sentences], mph11 [McDonald *et al.*, 2011] and mx14 [Ma and Xia, 2014] on the Google universal treebank v2. The mph11 results are copied from [Ma and Xia, 2014, Table 4]. All results are reported on gold part of speech tags. The numbers in parentheses are absolute improvements over [Ma and Xia, 2014]. Sup (1st) is the supervised first-order dependency parser used by [Ma and Xia, 2014] and sup(ae) is the Yara arc-eager supervised parser [Rasooli and Tetreault, 2015].

full trees, has a performance of 75.88%, close to the 76.15% accuracy for the method of [Ma and Xia, 2014]. There are clear gains as we move from θ^1 to θ^4, on all languages. The average accuracy for θ^4 is 78.89%.

Results with Multiple Source Languages, using Concatenation Table 4.2 shows results using multiple source languages, using the concatenation method. In these experiments for a given target language we use all other languages in our data as source languages. The performance of θ^1 improves from an average of 75.88% for a single source language, to 79.76% for multiple languages. The performance of θ^4 gives an additional improvement to 81.23%.

Results with Multiple Source Languages, using Voting The final set of results in Table 4.2 are for multiple languages using the voting strategy. There are further improvements: model θ^1 has average accuracy of 80.95%, and model θ^4 has average accuracy of 82.18%.

Results with Automatic POS Tags We use our final θ^4 models to parse the treebank with automatic tags provided by the same POS tagger used for tagging the parallel data. Table 4.3 shows the results for the transfer methods and the supervised parsing models of McDonald *et al.* [2011] and Rasooli and Tetreault [2015]. The first-order supervised method of McDonald *et al.* [2005b] gives only a 1.7% average absolute improvement in accuracy over the voting method. For one language (Swedish), our method actually gives improved accuracy over the 1st order parser.

L	$en \rightarrow trg$							concat							voting						
	$\mathcal{P}_{80} \cup \mathcal{P}_{\geq 7}$				$\mathcal{P}_{100}$			$\mathcal{P}_{80} \cup \mathcal{P}_{\geq 7}$				$\mathcal{P}_{100}$			$\mathcal{P}_{80} \cup \mathcal{P}_{\geq 7}$				$\mathcal{P}_{100}$		
	sen#	dep#	len	acc.	sen#	len	acc.	sen#	dep#	len	acc.	sen#	len	acc.	sen#	dep#	len	acc.	sen#	dep#	acc.
de	$34k$	9.6	28.3	84.7	$18k$	6.8	85.8	$98k$	9.4	28.8	84.1	$51k$	6.3	88.0	$75k$	10.8	23.5	84.5	$47k$	8.2	91.4
es	$108k$	10.9	31.4	87.3	$20k$	7.4	89.4	$536k$	11.0	31.8	86.3	$89k$	7.5	89.8	$346k$	17.0	28.5	86.1	$109k$	12.1	89.2
fr	$70k$	10.1	32.8	85.8	$13k$	6.7	84.1	$342k$	10.5	33.0	87.5	$47k$	6.9	89.5	$303k$	14.9	29.9	87.4	$78k$	11.7	91.2
it	$57k$	10.0	31.2	84.4	$9k$	6.3	76.9	$434k$	11.1	31.3	84.7	$70k$	7.4	87.2	$301k$	15.2	28.5	84.5	$101k$	12.4	87.9
pt	$489k$	10.0	31.0	85.2	$10k$	6.0	84.0	$462k$	11.1	31.3	81.4	$77k$	7.3	85.4	$222k$	12.4	30.3	81.3	$39k$	8.8	85.8
sv	$81k$	10.4	25.8	83.1	$30k$	7.4	87.8	$255k$	9.5	23.6	84.6	$79k$	6.8	89.7	$211k$	12.2	25.2	84.2	$86k$	9.5	88.8
avg	$140k$	10.2	30.1	85.1	$17k$	6.8	84.7	$354k$	10.4	30.0	84.8	$69k$	7.0	88.3	$243k$	13.7	27.6	84.7	$77k$	10.4	89.0

Table 4.5: Table showing statistics on projected dependencies for the target languages, for the single-source, multi-source (concat) and multi-source (voting) methods. "sen#" is the number of sentences. "dep#" is the average number of dependencies per sentence. "len" is the average sentence length. "acc." is the percentage of projected dependencies that agree with the output from a supervised parser.

Comparison to Previous Results Table 4.4 gives a comparison of the accuracy on the six languages, using the single source and multiple source methods, to previous work. As shown in the table, our model outperforms all models: among them, the results of McDonald *et al.* [2011] and Ma and Xia [2014] are directly comparable to us because they use the same training and evaluation data. The recent work of Xiao and Guo [2015] uses the same parallel data but evaluates on CoNLL treebanks but their results are lower than Ma and Xia [2014]. The work of [Guo *et al.*, 2015] evaluates on the same data as ours but uses different parallel corpora. They only reported on three languages (German: 60.35, Span-

ish: 71.90 and French: 72.93) which are all far bellow our results. The work of Grave and Elhadad [2015] is the state-of-the-art fully unsupervised model with minimal linguistic prior knowledge. The model of Zhang and Barzilay [2015] does not use any parallel data but uses linguistic information across languages. Their semi-supervised model selectively samples 50 annotated sentences but our model outperforms their model.

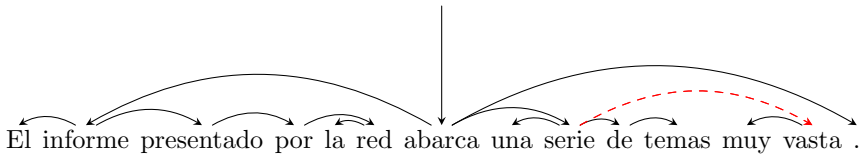
Compared to the results of McDonald *et al.* [2011] and Ma and Xia [2014] which are directly comparable, there are clear improvements across all languages; the highest accuracy, 82.18%, is a 5.51% absolute improvement over the average accuracy for Ma and Xia [2014].

4.4 Analysis

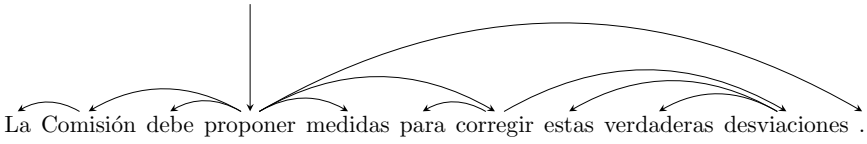
We conclude with some analysis of the accuracy of the projected dependencies for the different languages, for different definitions ($\mathcal{P}_{100}$, $\mathcal{P}_{80}$ etc.), and for different projection methods. Table 4.5 gives a summary of statistics for the various languages. Recall that German is used as the development language in our experiments; the other languages can be considered to be test languages. In all cases the accuracy reported is the percentage match to a supervised parser used to parse the same data.

There are some clear trends. The accuracy of the $\mathcal{P}_{100}$ datasets is high, with an average accuracy of 84.7% for the single source method, 88.3% for the concatenation method, and 89.0% for the voting method. The voting method not only increases accuracy over the single source method, but also increases the number of sentences (from an average 17k to 77k) and the average number of dependencies per sentence (from 6.8 to 10.4).

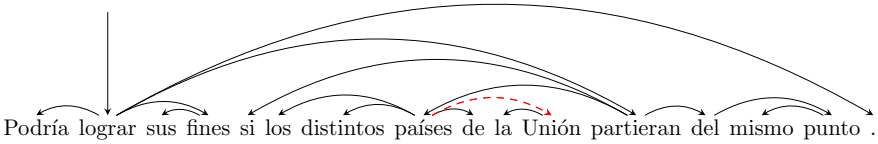
The accuracy of the $\mathcal{P}_{80} \cup \mathcal{P}_{\geq 7}$ datasets is slightly lower, with around 83-87% accu-



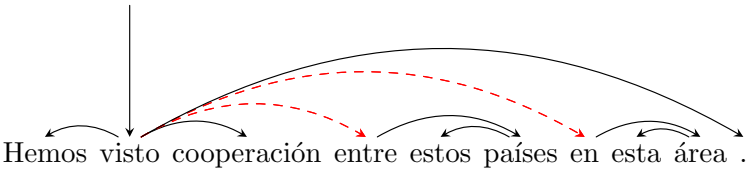
(a)



(b)



(c)



(d)

(e)

Figure 4.3: Randomly selected examples of Spanish dependency structures derived using the *voting* method. Dashed/red dependencies are mismatches with the output of a supervised Spanish parser; all other dependencies match the supervised parser. In these examples, 92.4% of dependencies match the supervised parser; this is close to the average match rate on Spanish of 89.2% for the voting method.

racy for the single source, concatenation and voting methods. The voting method gives a significant increase in the number of sentences—from an average of 140k to 243k. The average sentence length for this data is around 28 words, considerably longer than the

$\mathcal{P}_{100}$ data; the addition of longer sentences is very likely beneficial to the model. For the voting method the average number of dependencies is 13.7, giving an average density of 50% on these sentences.

The accuracy for the different languages, in particular for the voting data, is surprisingly uniform, with a range of 85.8-91.4% for the $\mathcal{P}_{100}$ data, and 81.3-87.4% for the $\mathcal{P}_{80} \cup \mathcal{P}_{\geq 7}$ data. The number of sentences for each language, the average length of those sentences, and average number of dependencies per sentence is also quite uniform, with the exception of German, which is a clear outlier. German has fewer sentences, and fewer dependencies per sentence: this may account for it having the lowest accuracy for our models. Future work should investigate why this is the case: one hypothesis is that German has quite different word order from the other languages (it is V2, and verb final), which may lead to a degradation in the quality of the alignments from GIZA++, or in the projection process.

Finally, figure 4.3 shows some randomly selected examples from the $\mathcal{P}_{100}$ data for Spanish, giving a qualitative feel for the data obtained using the voting method.

4.5 Conclusions

We have described a density-driven method for the induction of dependency parsers using parallel data and source-language parsers. The key ideas are a series of increasingly relaxed definitions of density, together with an iterative training procedure that makes use of these definitions. The method gives a significant gain over previous methods, with dependency accuracies approaching the level of fully supervised methods.

Chapter 5

Cross-Lingual Syntactic Transfer with Limited Resources

5.1 Introduction

Creating manually-annotated syntactic treebanks is an expensive and time consuming task. Recently there has been a great deal of interest in cross-lingual syntactic transfer, where a parsing model is trained for some language of interest, using only treebanks in other languages. There is a clear motivation for this in building parsing models for languages for which treebank data is unavailable.

In the previous chapter we showed that using parallel resources is beneficial for the transfer task. This chapter considers the problem of cross-lingual syntactic transfer with limited resources of monolingual and translation data. Specifically, we use the Bible corpus of Christodouloupoulos and Steedman [2014] as a source of translation data, and Wikipedia as a source of monolingual data. We deliberately limit ourselves to the use of Bible translation data because it is available for a very broad set of languages: the data from Christodouloupoulos and Steedman [2014] includes data from 100 languages. The Bible data contains a much smaller set of sentences (around 24,000) than other translation corpora, for example Europarl [Koehn, 2005], which has around 2 million sentences per language pair. This makes it a considerably more challenging corpus to work with.

Similarly, our choice of Wikipedia as the source of monolingual data is motivated by the availability of Wikipedia data in a very broad set of languages.

We introduce a set of simple but effective methods for syntactic transfer, as follows:

- We describe a method for deriving cross-lingual clusters, where words from different languages with a similar syntactic or semantic role are grouped in the same cluster. These clusters can then be used as features in a shift-reduce dependency parser.
- We describe a method for transfer of lexical information from the target language into source language treebanks, using word-to-word translation dictionaries derived from parallel corpora. Lexical features from the target language can then be integrated in parsing.
- We describe a method that integrates the above two approaches with the density-driven approach to annotation projection.

Experiments show that our model outperforms previous work on a set of European languages from the Google universal treebank [McDonald *et al.*, 2013]. We achieve 80.9% average unlabeled attachment score (UAS) on these languages; in comparison the work of Zhang and Barzilay [2015], Guo *et al.* [2016] and Ammar *et al.* [2016b] have a UAS of 75.4%, 76.3% and 77.8%, respectively. All of these previous works make use of the much larger Europarl [Koehn, 2005] corpus to derive lexical representations. When using Europarl data instead of the Bible, our approach gives 83.9% accuracy, a 1.7% absolute improvement over the density-driven approach. Finally, we conduct experiments on 38 datasets (26 languages) in the universal dependencies v1.3 [Nivre *et al.*, 2016]. Our method

has an average unlabeled dependency accuracy of 74.8% for these languages, more than 6% higher than the density-driven method. Thirteen datasets (10 languages) have accuracies higher than 80.0%.[1]

5.2 Background

This section gives a description of the underlying parsing models used in our experiments, the data sets used, and a baseline approach based on delexicalized parsing models.

5.2.1 The Parsing Model

We assume that the parsing model is a discriminative linear model, where given a sentence x, and a set of candidate parses $\mathcal{Y}(x)$, the output from the model is

$$y^*(x) = \arg\max_{y \in \mathcal{Y}(x)} \theta \cdot \phi(x, y)$$

where $\theta \in \mathbb{R}^d$ is a parameter vector, and $\phi(x, y)$ is a feature vector for the pair (x, y). In our experiments we use the shift-reduce dependency parser of Rasooli and Tetreault [2015], which is an extension of the approach in Zhang and Nivre [2011]. The parser is trained using the averaged structured Perceptron [Collins, 2002].

We assume that the feature vector $\phi(x, y)$ is the concatenation of three feature vectors:

- $\phi^{(p)}(x, y)$ is an unlexicalized set of features. Each such feature may depend on the part-of-speech (POS) tag of words in the sentence, but does not depend on the identity of individual words in the sentence.

[1] The parser code is available at `https://github.com/rasoolims/YaraParser/tree/transfer`.

- $\phi^{(c)}(x, y)$ is a set of cluster features. These features require access to a dictionary that maps each word in the sentence to an underlying cluster identity. Clusters may, for example, be learned using the Brown clustering algorithm [Brown *et al.*, 1992]. The features may make use of cluster identities in combination with POS tags.
- $\phi^{(l)}(x, y)$ is a set of lexicalized features. Each such feature may depend directly on word identities in the sentence. These features may also depend on part-of-speech tags or cluster information, in conjunction with lexical information.

Parsing Features

We used all features in Zhang and Nivre [2011], which describes features based on the word and part-of-speech at various positions on the stack and buffer of the transition system. In addition, we expand the Zhang and Nivre [2011] features to include clusters, as follows: whenever a feature tests the part-of-speech for a word in position 0 of the stack or buffer, we introduce features that replace the part-of-speech with the Brown clustering bit-string of length 4 and 6. Whenever a feature tests for the word identity at position 0 of the stack or buffer, we introduce a cluster feature that replaces the word with the full cluster feature. We take the cross product of all features corresponding to the choice of 4 or 6 length bit string for part-of-speech features.

5.2.2 Data Assumptions

Throughout this chapter we will assume that we have m source languages $\mathcal{L}_1 \ldots \mathcal{L}_m$, and a single target language $\mathcal{L}_{m+1}$. We assume the following data sources:

Source language treebanks. We have a treebank $\mathcal{T}_i$ for each language $i \in \{1 \dots m\}$.

Part-of-speech (POS) data. We have hand-annotated POS data for all languages $\mathcal{L}_1 \dots \mathcal{L}_{m+1}$. We assume that the data uses a universal POS set that is common across all languages.

Monolingual data. We have monolingual, raw text for each of the $(m+1)$ languages. We use $\mathcal{D}_i$ to refer to the monolingual data for the ith language.

Translation data. We have translation data for all language pairs. We use $\mathcal{B}_{i,j}$ to refer to translation data for the language pair (i, j) where $i, j \in \{1 \dots (m+1)\}$ and $i \neq j$.

In our main experiments we use the Google universal treebank [McDonald *et al.*, 2013] as our source language treebanks[2] (this treebank provides universal dependency relations and POS tags), Wikipedia data as our monolingual data, and the Bible from Christodouloupoulos and Steedman [2014] as the source of our translation data. In additional experiments we use the Europarl corpus as a source of translation data, in order to measure the impact of using the smaller Bible corpus.

5.2.3 A Baseline Approach: Delexicalized Parsers with Self-Training

Given the data assumption of a universal POS set, the feature vectors $\phi^{(p)}(x, y)$ can be shared across languages. A simple approach is then to simply train a delexicalized parser using treebanks $\mathcal{T}_1 \dots \mathcal{T}_m$, using the representation $\phi(x, y) = \phi^{(p)}(x, y)$ (see [McDonald *et al.*, 2013; Täckström *et al.*, 2013]).

[2]We also train our best performing model on the newly released universal treebank v1.3 [Nivre *et al.*, 2016]. See §5.4.3 for more details.

Feature	Description
82A	Order of subject and verb
83A	Order of object and verb
85A	Order of adposition and noun phrase
86A	Order of genitive and noun
87A	Order of adjective and noun
88A	Order of demonstrative and noun

Table 5.1: The six properties from the world atlas of language structures (WALS) [Dryer and Haspelmath, 2013] used to select the source languages for each target language in our experiments.

Our baseline approach makes use of a delexicalized parser, with two refinements:

WALS properties. We use the six properties from the World Atlas of Language Structures (WALS) [Dryer and Haspelmath, 2013] to select a subset of closely related languages for each target language. These properties are shown in Table 5.1. The model for a target language is trained on treebank data from languages where at least 4 out of 6 WALS properties are common between the source and target language.[3] This gives a slightly stronger baseline. Our experiments showed an improvement in average labeled dependency accuracy for the languages from 62.52% to 63.18%. Table 5.2 shows the set of source languages used for each target language. These source languages are used for all experiments in this chapter.

Self-training. We use self-training [McClosky *et al.*, 2006] to further improve parsing performance. Specifically, we first train a delexicalized model on treebanks $\mathcal{T}_1 \ldots \mathcal{T}_m$; then use the resulting model to parse a dataset $\mathcal{T}_{m+1}$ that includes target-language sentences which have POS tags but do not have dependency structures. We finally use the auto-

[3]There was no effort to optimize this choice; future work may consider more sophisticated sharing schemes.

Target	Sources
en	de, fr, pt, sv
de	en, fr, pt
es	fr, it, pt
fr	en, de, es, it, pt, sv
it	es, fr, pt
pt	en, de, es, fr, it, sv
sv	en, fr, pt

Table 5.2: The selected source languages for each target language in the Google universal treebank v2 [McDonald *et al.*, 2013]. A language is chosen as a source language if it has at least 4 out of 6 WALS properties in common with the target language.

matically parsed data $\mathcal{T}'_{m+1}$ as the treebank data and retrain the model. This last model is trained using all features (unlexicalized, clusters, and lexicalized). Self-training in this way gives an improvement in labeled accuracy from 63.18% to 63.91%.

5.2.4 Translation Dictionaries

Our only use of the translation data $\mathcal{B}_{i,j}$ for $i, j \in \{1 \ldots (m+1)\}$ is to construct a translation dictionary $t(w, i, j)$. Here i and j are two languages, w is a word in language $\mathcal{L}_i$, and the output $w' = t(w, i, j)$ is a word in language $\mathcal{L}_j$ corresponding to the most frequent translation of w into this language.

We define the function $t(w, i, j)$ as follows: We first run the GIZA++ alignment process [Och and Ney, 2003] on the data $\mathcal{B}_{i,j}$. We then keep intersected alignments between sentences in the two languages. Finally, for each word w in $\mathcal{L}_i$, we define $w' = t(w, i, j)$ to be the target language word most frequently aligned to w in the aligned data. If a word w is never seen aligned to a target language word w', we define $t(w, i, j) = \texttt{NULL}$.

Inputs: 1) Monolingual texts $\mathcal{D}_i$ for $i = 1 \ldots (m+1)$; 2) a function $t(w, i, j)$ that translates a word $w \in \mathcal{L}_i$ to $w' \in \mathcal{L}_j$; and 3) a parameter α such that $0 < \alpha < 1$.

Algorithm:

$\mathcal{D} = \{\}$
for i = 1 to $m + 1$ **do**
  for each sentence $s \in \mathcal{D}_i$ **do**
    for $p = 1$ to $|s|$ **do**
      Sample $\bar{a} \sim [0, 1)$
      if $\bar{a} \geq \alpha$ **then**
        continue
      Sample $j \sim unif\{1, ..., m+1\} \backslash \{i\}$
      $w' = t(s_p, i, j)$
      if $w' \neq$ `NULL` **then**
        Set $s_p = w'$
  $\mathcal{D} = \mathcal{D} \cup \{s\}$
Use the algorithm of Stratos *et al.* [2015] on $\mathcal{D}$ to learn a clustering $\mathcal{C}$.

Output: The clustering $\mathcal{C}$.

Figure 5.1: An algorithm for learning a cross-lingual clustering. In our experiments we used the parameter value $\alpha = 0.3$.

5.3 Our Approach

We now describe an approach that gives significant improvements over the baseline. §5.3.1 describes a method for deriving cross-lingual clusters, allowing us to add cluster features $\phi^{(c)}(x, y)$ to the model. §5.3.2 describes a method for adding lexical features $\phi^{(l)}(x, y)$ to the model. §5.3.3 describes a method for integrating the approach with the density-driven approach. Finally, §7.4 describes experiments. We show that each of the above steps leads to improvements in accuracy.

5.3.1 Learning Cross-Lingual Clusters

We now describe a method for learning cross-lingual clusters. This follows previous work on cross-lingual clustering algorithms [Täckström *et al.*, 2012]. A *cross-lingual* hierarchi-

cal clustering is a function $C(w, l)$ where the clusters are shared across the $(m + 1)$ languages of interest. That is, the word w can be from any of the $(m+1)$ languages. Ideally, a cross-lingual clustering should put words across different languages which have a similar syntactic and/or semantic role in the same cluster. There is a clear motivation for cross-lingual clustering in the parsing context. We can use the cluster-based features $\phi^{(c)}(x, y)$ on the source language treebanks $\mathcal{T}_1 \ldots \mathcal{T}_m$, and these features will now generalize beyond these treebanks to the target language $\mathcal{L}_{m+1}$.

We learn a cross-lingual clustering by leveraging the monolingual data sets $\mathcal{D}_1 \ldots \mathcal{D}_{m+1}$, together with the translation dictionaries $t(w, i, j)$ learned from the translation data. Figure 5.1 shows the algorithm that learns a cross-lingual clustering. The algorithm first prepares a multilingual corpus, as follows: for each sentence s in the monolingual data $\mathcal{D}_i$, for each word in s, with probability α, we replace the word with its translation into some randomly chosen language. Once this data is created, we can easily obtain a cross-lingual clustering. Figure 5.1 shows the complete algorithm. The intuition behind this method is that by creating the cross-lingual data in this way, we bias the clustering algorithm towards putting words that are translations of each other in the same cluster.

5.3.2 Treebank Lexicalization

We now describe how to introduce lexical representations $\phi^{(l)}(x, y)$ to the model. Our approach is simple: we take the treebank data $\mathcal{T}_1 \ldots \mathcal{T}_m$ for the m source languages, together with the translation lexicons $t(w, i, m + 1)$. For any word w in the source treebank data,

Inputs: 1) Source treebanks $\mathcal{T}_i$ for $i = 1...m$; 2) function $t(w, i, m+1)$ that translates a word $w \in \mathcal{L}_i$ to $w' \in \mathcal{L}_{m+1}$.

Algorithm:

for $i = 1$ to m **do**
  for sentence $s \in \mathcal{T}_i$ **do**
    for $j = 1$ to $|s|$ **do**
      $w' = t(s_j, i, m+1)$
      $original\_word(s_j) = s_j$
      $s_j = w'$ ▷ $s_j = \texttt{NULL}$ iff $w' = \texttt{NULL}$

Outputs: $\cup_{i=1}^{m}$ modified $\mathcal{T}_i$

Figure 5.2: Algorithm for lexicalizing the source treebanks with target words.

we can look up its translation $t(w, i, m+1)$ in the lexicon, and add this translated form to the underlying sentence. Features can now consider lexical identities derived in this way. In many cases the resulting translation will be the NULL word, leading to the absence of lexical features. However, the representations $\phi^{(p)}(x, y)$ and $\phi^{(c)}(x, y)$ still apply in this case, so the model is robust to some words having a NULL translation. Figure 5.2 shows the pseudo-code for the lexicalization approach.

5.3.3 Integration with the Density-Driven Projection Method

In this section we describe a method for integrating our approach with the cross-lingual transfer method, which makes use of density-driven projections.

In annotation projection methods [Hwa *et al.*, 2005; McDonald *et al.*, 2011], it is assumed that we have translation data $\mathcal{B}_{i,j}$ for a source and target language, and that we have a dependency parser in the source language $\mathcal{L}_i$. The translation data consists of pairs (e, f) where e is a source language sentence, and f is a target language sentence. A method such as GIZA++ is used to derive an alignment between the words in e and f,

for each sentence pair; the source language parser is used to parse e. Each dependency in e is then potentially transferred through the alignments to create a dependency in the target sentence f. Once dependencies have been transferred in this way, a dependency parser can be trained on the dependencies in the target language.

The density-driven approach makes use of various definitions of "density" of the projected dependencies. For example, $\mathcal{P}_{100}$ is the set of projected structures where the projected dependencies form a full projective parse tree for the sentence; $\mathcal{P}_{80}$ is the set of projected structures where at least 80% of the words in the projected structure are a modifier in some dependency. An iterative training process is used, where the parsing algorithm is first trained on the set $\mathcal{T}_{100}$ of complete structures, and where progressively less dense structures are introduced in learning.

We integrate our approach with the density-driven approach as follows: consider the treebanks $\mathcal{T}_1 \ldots \mathcal{T}_m$ created using the lexicalization method of §5.3.2. We add all trees in these treebanks to the set $\mathcal{P}_{100}$ of full trees used to initialize the density-driven method. In addition we make use of the representations $\phi^{(p)}$, $\phi^{(c)}$ and $\phi^{(l)}$, throughout the learning process.

5.4 Experiments

This section first describes the experimental settings, then reports results.

5.4.1 Data and Tools

Data In the first set of experiments, we consider 7 European languages studied in several pieces of previous work [Ma and Xia, 2014; Zhang and Barzilay, 2015; Guo *et al.*, 2016;

Lang.	en	de	es	fr	it	pt	sv
#Sen.	31.8	20.0	13.6	13.6	10.1	6.1	3.9
#Token	750.5	408.2	402.3	372.1	311.1	169.3	60.6
#Type	3.8	6.1	2.7	2.4	2.1	1.6	1.3

Table 5.3: Sizes of the monolingual datasets for each of our languages. All numbers are in millions.

Ammar *et al.*, 2016a; Lacroix *et al.*, 2016]. More specifically, we use the 7 European languages in the Google universal treebank (v.2; standard data) [McDonald *et al.*, 2013]. As in previous work, gold part-of-speech tags are used for evaluation. We use the concatenation of the treebank training sentences, Wikipedia data and the Bible monolingual sentences as our monolingual raw text. Table 5.3 shows statistics for the monolingual data. We use the Bible from Christodouloupoulos and Steedman [2014], which includes data for 100 languages, as the source of translations. We also conduct experiments with the Europarl data (both with the original set and a subset of it with the same size as the Bible) to study the effects of translation data size and domain shift. The statistics for translation data is shown in Table 5.4.

In a second set of experiments, we run experiments on 38 datasets (26 languages) in the more recent Universal Dependencies v1.3 corpus [Nivre *et al.*, 2016]. The full set of languages we use is listed in Table 5.9.[4] We use the Bible as the translation data, and Wikipedia as the monolingual text. The standard training, development and test set splits are used in all experiments. The development sets are used for analysis, given in §5.5 of this chapter.

[4]We excluded languages that are not completely present in the Bible of Christodouloupoulos and Steedman [2014] (Ancient Greek, Basque, Catalan, Galician, Gothic, Irish, Kazakh, Latvian, Old Church Slavonic, and Tamil). We also excluded Arabic, Hebrew, Japanese and Chinese, as these languages have tokenization and/or morphological complexity that goes beyond the scope of this chapter. Future work should consider these languages.

Data	Lang.	en	de	es	fr	it	pt	sv
Bible	tokens	1.5M	665K	657K	732K	613K	670K	696K
	types	16K	20K	27K	22K	29K	29K	23K
EU-S	tokens	718K	686K	753K	799K	717K	739K	645K
	types	22K	41K	31K	27K	30K	32K	39K
Europarl	tokens	56M	50M	57M	62M	55M	56M	46M
	types	133K	400K	195K	153K	188K	200K	366K

Table 5.4: Statistics for the Bible, sampled Europarl (EU-S) and Europarl datasets. Each individual Bible text file from Christodouloupoulos and Steedman [2014] consists of 24720 sentences, except for English datasets, where two translations into English are available, giving double the amount of data. Each text file from the sampled Europarl datasets consists of 25K sentences and Europarl has approximately 2 million sentences per language pair.

Brown Clustering Algorithm We use the off-the-shelf Brown clustering tool[5] [Liang, 2005] to train monolingual Brown clusters with 500 clusters. The monolingual Brown clusters are used as features over lexicalized values created in $\phi^{(l)}$, and in self-training experiments. We train our cross-lingual clustering with the off-the-shelf-tool[6] from Stratos *et al.* [2015]. We set the window size to 2 with a cluster size of 500.[7]

Parsing Model We use the k-beam arc-eager dependency parser of Rasooli and Tetreault [2015], which is similar to the model of Zhang and Nivre [2011]. We modify the parser such that it can use both monolingual and cross-lingual word cluster features. The parser is trained using the the maximum violation update strategy [Huang *et al.*, 2012]. We use three epochs of training for all experiments. We use the DEPENDABLE Tool [Choi *et al.*, 2015] to calculate significance tests on several of the comparisons (details are given in the captions to tables 5.5, 5.6, and 5.9).

[5] `https://github.com/percyliang/brown-cluster`

[6] `https://github.com/karlstratos/singular`

[7] Usually the original Brown clusters are better features for parsing but their training procedure does not scale well to large datasets. Therefore we use the more efficient algorithm from Stratos *et al.* [2015] on the larger cross-lingual datasets to obtain word clusters.

L	Baseline		Our method using the Bible					
			§5.3.1		§5.3.2		§5.3.3	
	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS
en	58.2	65.5	65.0	72.3	66.3	74.0	**70.8**	**76.5**
de	49.7	59.1	51.6	59.7	54.9	62.6	**65.2**	**72.8**
es	68.3	77.2	73.1	79.6	76.6	81.9	**76.7**	**82.1**
fr	67.3	77.7	69.5	79.9	74.4	81.9	**75.8**	**82.2**
it	69.7	79.4	71.6	80.0	74.7	82.8	**76.1**	**83.3**
pt	71.5	77.5	76.9	81.5	81.0	84.4	**81.3**	**84.7**
sv	62.6	74.2	63.5	75.1	68.2	78.7	**71.2**	**80.3**
avg	63.9	72.9	67.3	75.5	70.9	78.1	**73.9**	**80.3**

Table 5.5: Performance of different models in this chapter; first the baseline model, then models trained using the methods described in sections §5.3.1–5.3.3. All results make use of the Bible as a source of translation data. All differences in UAS and LAS are statistically significant with $p < 0.001$ using McNemar's test, with the exception of "de" UAS/LAS Baseline vs. 3.1 (i.e., 49.7 vs 51.6 UAS and 59.1 vs 59.7 LAS are not significant differences).

Word alignment We use the intersected alignments from GIZA++ [Och and Ney, 2003] on translation data. We exclude sentences in translation data with more than 100 words.

5.4.2 Results on the Google Treebank

Table 5.5 shows the dependency parsing accuracy for the baseline delexicalized approach, and for models which add 1) cross-lingual clusters (§5.3.1); 2) lexical features (§5.3.2); and 3) integration with the density-driven method. Each of these three steps gives significant improvements in performance. The final LAS/UAS of 73.9/80.3% is several percentage points higher than the baseline accuracy of 63.9/72.9%.

Comparison to the Density-Driven Approach using Europarl Data Table 5.6 shows accuracies for the density-driven, first using Europarl data and second using the Bible alone (with no cross-lingual clusters or lexicalization). The Bible data is considerably smaller than Europarl (around 100 times smaller), and it can be seen that results using the Bible

Lang.	Bible				Europarl-Sample				Europarl			
	Density		This Chapter		Density		This Chapter		Density		This Chapter	
	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS
en	59.1	66.4	70.8	76.5	64.3	72.8	70.2	76.2	68.4	76.3	71.1	77.5
de	60.2	69.5	65.2	72.8	61.6	72.0	64.9	73.0	73.0	79.7	75.6	82.1
es	70.3	76.8	76.7	82.1	72.0	78.3	76.0	81.5	74.6	80.9	76.6	82.6
fr	69.9	76.9	75.8	82.2	71.9	79.0	75.7	82.5	76.3	82.7	77.4	83.9
it	71.1	78.5	76.1	83.3	73.2	80.4	76.2	82.9	77.0	83.7	77.4	84.4
pt	72.1	76.4	81.3	84.7	75.3	79.7	81.61	84.8	77.3	82.1	82.1	85.6
sv	66.5	76.3	71.2	80.3	71.9	80.6	73.5	81.6	75.6	84.1	76.9	84.5
avg	67.0	75.7	73.9	80.3	70.0	77.6	74.0	80.4	74.6	81.3	76.7	82.9

Table 5.6: Results for our method using different sources of translation data. "Density" refers to the density-driven approach; "Our model" gives results using the methods described in sections 5.3.1–5.3.3 of this chapter. The "Bible" experiments use the Bible data of Christodouloupoulos and Steedman [2014]. The "Europarl" experiments use the Europarl data of Koehn [2005]. The "Europarl-Sample" experiments use 25K randomly chosen sentences from Europarl; this gives a similar number of sentences to the Bible data. All differences in LAS and UAS in this table between the density and "our model" settings (i.e., for the Bible, Europarl-Sample and Europarl settings) are found to be statistically significant according to McNemar's sign test.

Lang.	MX14	LA16	ZB15		GCY16		AMB16		DD		Our model				Supervised	
											Bible		Europarl			
	UAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS		
en	–	–	59.8	70.5	–	–	–	–	68.4	76.3	70.8	76.5	**71.1**	**77.5**	92.0	93.8
de	74.3	76.0	54.1	62.5	55.9	65.0	57.1	65.2	73.0	79.7	65.2	72.8	**75.6**	**82.1**	79.4	85.3
es	75.5	78.9	68.3	78.0	73.0	79.0	74.6	80.2	74.6	80.9	**76.7**	82.1	76.0	**82.6**	82.3	86.7
fr	76.5	80.8	68.8	78.9	71.0	77.6	73.9	80.6	76.3	82.7	75.8	82.2	**77.4**	**83.9**	81.7	86.3
it	77.7	79.4	69.4	79.3	71.2	78.4	72.5	80.7	77.0	83.7	76.1	83.3	**77.4**	**84.4**	86.1	88.8
pt	76.6	–	72.5	78.6	78.6	81.8	77.0	81.2	77.3	82.1	81.3	84.7	**82.1**	**85.6**	87.6	89.4
sv	79.3	83.0	62.5	75.0	69.5	78.2	68.1	79.0	75.6	84.1	71.2	80.3	**76.9**	**84.5**	84.1	88.1
$avg_{\backslash en}$	76.7	–	65.9	75.4	69.3	76.3	70.5	77.8	75.6	82.2	74.4	80.9	**77.7**	**83.9**	83.5	87.4

Table 5.7: Comparison of our work using the Bible and Europarl data, with previous work: MX14 [Ma and Xia, 2014], LA16 [Lacroix *et al.*, 2016], ZB15 [Zhang and Barzilay, 2015], GCY16 [Guo *et al.*, 2016], AMB 16 [Ammar *et al.*, 2016b], and DD (the density-driven approach). "Supervised" refers to the performance of the parser trained on fully gold standard data in a supervised fashion (i.e. the practical upper-bound of our model). "$avg_{\backslash en}$" refers to the average accuracy for all datasets except English.

are several percentage points lower than the results for Europarl (75.7% UAS vs. 81.3% UAS). Integrating cluster-based and lexicalized features described in the current chapter with the density-driven approach closes much of this gap in performance (80.3% UAS).

Thus we have demonstrated that we can get close to the performance of the Europarl-based models using only the Bible as a source of translation data. Using our approach on the full Europarl data gives an average UAS of 82.9%, an improvement from the 81.3% UAS of the density-driven approach.

Table 5.6 also shows results when we use a random subset of the Europarl data, in which the number of sentences (25,000) is chosen to give a very similar size to the Bible. It can be seen that accuracies using the Bible vs. the Europarl-Sample are very similar (80.3% vs. 80.4% UAS), suggesting that the size of the translation corpus is much more important than the genre.

Comparison to Other Previous Work Table 5.7 compares the accuracy of our method to the following related work: 1) Ma and Xia [2014], who describe an annotation projection method based on entropy regularization; 2) Lacroix *et al.* [2016], who describe an annotation projection method based on training on partial trees with dynamic oracles; 3) Zhang and Barzilay [2015], who describe a method that learns cross-lingual embeddings and bilingual dictionaries from Europarl data, and uses these features in a discriminative parsing model; 4) Guo *et al.* [2016], who describe a method that learns cross-lingual embeddings from Europarl data and uses a shift-reduce neural parser with these representations; 5) Ammar *et al.* [2016b][8], who use the same embeddings as Guo *et al.* [2016], within an LSTM-based parser; and 6) The density-driven approach on the Europarl data. Our method gives significant improvements over the first three models, in spite of using the Bible translation data rather than Europarl. When using the Europarl data, our

[8]This work was later published under a different title [Ammar *et al.*, 2016a] without including UAS results.

Lang.	Density-driven		Our model (§5.3.3)			
			Bible		Europarl	
	LAS	UAS	LAS	UAS	LAS	UAS
en	66.2	74.4	67.8	74.4	**68.0**	**75.1**
de	71.6	78.8	61.9	70.3	**73.6**	**80.8**
es	72.3	79.2	73.8	79.9	**74.2**	**80.7**
fr	73.5	80.8	72.6	79.9	**75.0**	**82.3**
it	74.9	82.0	74.0	81.7	**75.3**	**82.6**
pt	75.4	80.7	79.2	83.3	**80.4**	**84.4**
sv	73.4	82.0	67.3	77.2	**73.7**	**82.2**
avg	72.5	79.7	70.9	78.1	**74.3**	**81.2**

Table 5.8: The final results based on automatic part of speech tags.

method improves the state-of-the-art model of the density-driven approach.

Performance with Automatic POS Tags For completeness, Table 5.8 gives results for our method with automatic part-of-speech tags. The tags are obtained using the model of Collins [2002][9] trained on the training part of the treebank dataset. Future work should study approaches that transfer POS tags in addition to dependencies.

5.4.3 Results on the Universal Dependencies v1.3

Table 5.9 gives results on 38 datasets (26 languages) from the newly released universal dependencies corpus [Nivre *et al.*, 2016]. Given the number of treebanks and to speed up training, we pick source languages that have at least 5 out of 6 common WALS properties with each target language. Our experiments are carried out using the Bible as our translation data. As shown in Table 5.9, our method consistently outperforms the density-driven method and for many languages the accuracy of our method gets close to the accuracy

[9] `https://github.com/rasoolims/SemiSupervisedPosTagger`

Dataset	Density-driven		Our model		Supervised	
	LAS	UAS	LAS	UAS	LAS	UAS
it	74.3	81.3	79.8	86.1	88.4	90.7
sl	68.2	75.9	78.6	84.1	86.3	89.1
es	69.1	77.5	76.3	84.1	83.5	86.9
bg	66.2	79.5	72.0	83.6	85.5	90.5
pt	66.7	75.8	74.8	83.4	83.0	86.7
es-ancora	68.9	77.5	74.6	83.1	86.5	89.4
fr	72.0	77.9	76.6	82.6	84.5	87.1
sv-lines	67.5	76.7	73.3	82.4	81.0	85.4
pt-br	68.3	75.2	76.2	82.0	87.8	89.7
sv	65.9	75.7	71.7	81.3	83.6	87.7
no	71.7	78.8	74.3	81.2	88.0	90.5
pl	65.4	77.6	70.1	81.0	85.1	90.3
hr	55.8	70.2	65.9	80.9	76.2	85.1
cs-cac	61.1	70.3	69.0	78.5	82.4	87.6
da	63.1	72.8	68.3	77.8	80.8	84.3
en-lines	67.0	75.9	68.6	77.3	80.7	84.6
cs	59.0	68.1	67.2	76.4	84.5	88.7
id	38.0	55.7	57.8	76.0	79.8	85.1
de	61.3	72.8	64.9	75.7	80.2	85.8
ru-syntagrus	56.0	70.7	61.6	75.3	82.0	87.8
ru	56.7	64.8	65.4	74.8	71.9	77.7
cs-cltt	57.5	65.4	65.6	74.7	77.1	81.4
ro	54.6	67.4	60.7	74.6	78.2	85.3
la	54.5	71.6	55.7	72.8	43.1	52.5
nl-lassysmall	51.5	62.6	61.9	71.7	76.5	80.6
el	53.7	66.7	59.6	71.0	79.1	83.1
et	48.9	65.6	56.9	70.9	75.9	82.9
hi	34.4	50.6	49.9	69.9	89.4	92.9
hu	26.1	48.9	55.0	69.9	69.5	79.4
en	59.7	68.1	61.8	69.0	85.3	88.1
fi-ftb	50.3	63.2	56.5	67.5	73.3	79.7
fi	49.8	60.8	57.3	66.4	73.4	78.2
la-ittb	44.1	55.4	51.8	62.8	76.2	80.9
nl	40.6	49.4	50.1	62.0	70.1	75.0
la-proiel	43.6	60.3	45.0	61.3	64.9	72.9
sl-sst	42.4	59.2	47.6	60.6	63.4	70.4
fa	44.4	53.2	46.5	56.0	84.1	87.5
tr	05.3	18.5	32.7	51.9	65.6	78.8
Average	56.7	68.1	64.0	74.8	78.9	83.8

Table 5.9: Results for the density driven method and ours using the Bible data on the universal dependencies v1.3 [Nivre *et al.*, 2016]. The table is sorted by the performance of our method. The last major columns shows the performance of the supervised parser. The abbreviations are as follows: bg (Bulgarian), cs (Czech), da (Danish), de (German), el (Greek), en (English), es (Spanish), et (Estonian), fa (Persian (Farsi)), fi (Finnish), fr (French), hi (Hindi), hr (Croatian), hu (Hungarian), id (Indonesian), it (Italian), la (Latin), nl (Dutch), no (Norwegian), pl (Polish), pt (Portuguese), ro (Romanian), ru (Russian), sl (Slovenian), sv (Swedish), and tr (Turkish). All differences in LAS and UAS in this table were found to be statistically significant according to McNemar's sign test with $p < 0.001$.

of the supervised parser. In all the languages, our method is significantly better than the density-driven method using the McNemar's test with $p < 0.001$.

Accuracy on some languages (e.g., Persian (fa) and Turkish (tr)) is low, suggesting that future work should consider more powerful techniques for these languages. There are two important facts to note. First, the number of fully projected trees in some languages is so low such that the density-driven approach cannot start with a good initialization to fill in partial dependencies. For example Turkish has only one full tree with only six words, Persian with 25 trees, and Dutch with 28 trees. Second, we observe very low accuracies in supervised parsing for some languages in which the number of training sentences is very low (for example, Latin has only 1326 projective trees in the training data).

5.5 Analysis

We conclude with some analysis of the accuracy of the method on different dependency types, across the different languages. Table 5.10 shows precision and recall on different dependency types in English (using the Google treebank). The improvements in accuracy when moving from the delexicalized model to the Bible or Europarl model apply quite uniformly across all dependency types, with all dependency labels showing an improvement.

Table 5.11 shows the dependency accuracy sorted by part-of-speech tag of the modifier in the dependency. We break the results into three groups: G1 languages, where UAS is at least 80% overall; G2 languages, where UAS is between 70% and 80%; and G3 languages, where UAS is less than 70%. There are some quite significant differences in accuracy depending on the POS of the modifier word. In the G1 languages, for example, ADP, DET,

dependency	freq	Delexicalized		Bible		Europarl	
		prec./rec.	f1	prec./rec.	f1	prec./rec.	f1
adpmod	10.6	57.2/62.7	59.8	67.1/71.8	69.4	70.3/73.8	72.0
adpobj	10.6	65.5/69.1	67.2	75.3/77.4	76.3	75.9/79.2	77.6
det	9.5	72.5/75.6	74.0	84.3/86.3	85.3	86.6/89.8	88.2
compmod	9.1	83.7/ 59.9	69.8	87.3/70.2	77.8	89.0/73.0	80.2
nsubj	8.0	69.7/60.0	64.5	82.1/77.5	79.7	83.0/78.1	80.5
amod	7.0	76.9/72.3	74.5	83.0/78.7	80.8	80.9/77.9	79.4
ROOT	4.8	69.3/70.4	69.8	85.0/85.1	85.0	83.8/85.8	84.8
num	4.6	67.8/55.3	60.9	70.7/55.2	62.0	75.0/63.0	68.5
dobj	4.5	60.8/80.3	69.2	64.0/84.9	73.0	68.4/86.6	76.5
advmod	4.1	65.9/61.9	63.8	72.7/68.1	70.3	69.6/68.8	69.2
aux	3.5	76.6/93.9	84.4	90.2/95.9	93.0	89.6/96.4	92.9
cc	2.9	67.6/61.7	64.5	73.1/73.1	73.1	73.1/73.3	73.2
conj	2.8	46.3/56.1	50.7	45.6/62.9	52.9	48.1/62.8	54.5
dep	2.0	90.5/25.8	40.1	99.2/33.8	50.4	92.0/34.4	50.1
poss	2.0	72.1/30.6	43.0	77.9/45.8	57.7	78.2/42.1	54.7
ccomp	1.6	76.2/28.4	41.3	88.0/61.3	72.3	82.3/69.1	75.1
adp	1.2	20.0/0.5	0.9	92.7/42.1	57.9	91.7/23.3	37.1
nmod	1.2	60.7/48.1	53.7	56.3/47.1	51.3	52.6/46.2	49.2
xcomp	1.2	66.6/48.6	56.2	85.1/65.3	73.9	78.3/71.0	74.5
mark	1.1	37.8/24.6	29.8	73.8/50.3	59.8	62.8/53.8	57.9
advcl	0.8	23.6/22.3	22.9	38.7/38.8	38.8	38.0/42.9	40.3
appos	0.8	8.5/43.0	14.3	20.4/61.0	30.6	26.4/61.7	37.0
auxpass	0.8	88.9/91.4	90.1	96.8/97.1	97.0	98.6/98.6	98.6
rcmod	0.8	38.2/33.3	35.6	46.8/54.6	50.4	52.7/55.0	53.8
nsubjpass	0.7	73.2/64.9	68.8	87.6/77.0	82.0	85.5/75.8	80.3
acomp	0.6	86.8/92.5	89.6	83.3/93.5	88.1	91.0/93.9	92.4
adpcomp	0.6	42.0/70.2	52.5	47.9/61.5	53.9	55.4/47.1	50.9
partmod	0.6	20.2/36.0	25.8	36.7/49.1	42.0	31.0/40.7	35.2
attr	0.5	67.7/86.4	75.9	76.5/92.1	83.6	72.6/92.7	81.4
neg	0.5	74.7/85.0	79.6	93.3/91.0	92.1	92.6/89.8	91.2
prt	0.3	27.4/92.2	42.2	32.4/96.6	48.5	31.9/97.4	48.1
infmod	0.2	30.7/72.4	43.2	38.4/64.4	48.1	42.6/63.2	50.9
expl	0.1	84.8/87.5	86.2	93.8/93.8	93.8	91.2/96.9	93.9
iobj	0.1	51.7/78.9	62.5	88.9/84.2	86.5	36.4/84.2	50.8
mwe	0.1	0.0/0.0	0.0	5.3/2.1	3.0	11.1/10.4	10.8
parataxis	0.1	5.6/19.6	8.7	17.3/47.1	25.3	14.6/45.1	22.0
cop	0.0	0.0/0.0	0.0	0.0/0.0	0.0	0.0/0.0	0.0
csubj	0.0	12.8/33.3	18.5	22.2/26.7	24.2	25.0/46.7	32.6
csubjpass	0.0	100.0/100.0	100.0	100.0/100.0	100.0	50.0/100.0	66.7
rel	0.0	100.0/6.3	11.8	90.9/62.5	74.1	66.7/37.5	48.0

Table 5.10: Precision, recall and f-score of different dependency relations on the English development data of the Google universal treebank. The major columns show the dependency labels ("dep."), frequency ("freq."), the baseline delexicalized model ("delex"), and our method using the Bible and Europarl ("EU") as translation data. The rows are sorted by frequency.

POS	$\mathcal{G}1$		$\mathcal{G}2$		$\mathcal{G}3$	
	freq%	acc.	freq%	acc.	freq%	acc.
NOUN	22.0	77.6	30.0	71.2	25.3	58.0
ADP	16.9	92.3	10.9	92.3	11.2	90.6
DET	11.9	96.4	3.0	92.4	3.6	86.6
VERB	11.7	74.5	13.5	66.1	17.1	52.2
PROPN	8.1	79.0	4.7	65.2	6.8	49.5
ADJ	8.0	88.5	12.7	86.9	8.4	73.6
PRON	5.4	87.7	5.9	82.2	7.6	71.1
ADV	4.3	76.0	6.6	70.9	5.6	61.9
CONJ	3.6	71.8	4.7	63.0	4.2	60.4
AUX	2.7	91.5	1.7	88.9	3.0	70.6
NUM	2.2	79.5	2.3	68.4	2.0	75.7
SCONJ	1.8	80.5	1.9	77.2	2.6	65.0
PART	0.9	80.2	1.8	64.3	1.9	45.0
X	0.2	52.3	0.1	40.5	0.6	36.9
SYM	0.1	64.3	0.1	40.9	0.1	45.5
INTJ	0.1	78.5	0.0	51.7	0.3	60.2

Table 5.11: Accuracy of unlabeled dependencies by POS of the modifier word, for three groups of languages for the universal dependencies experiments in Table 5.9: G1 (languages with UAS $\geq$ 80), G2 (languages with 70 $\leq$ UAS $<$ 80), G3 (languages with UAS $<$ 70). The rows are sorted by frequency in the G1 languages.

ADJ, PRON and AUX all have over 85% accuracy; in contrast NOUN, VERB, PROPN, ADV all have accuracy that is less than 80%. A very similar pattern is seen for the G2 languages, with ADP, DET, ADJ, and AUX again having greater than 85% accuracy, but NOUN, VERB, PROPN and ADV having lower accuracies. These results suggest that difficulty varies quite significantly depending on the modifier POS, and different languages show the same patterns of difficulty with respect to the modifier POS.

Table 5.12 shows accuracy sorted by the POS tag of the *head* word of the dependency. By far the most frequent head POS tags are NOUN, VERB, and PROPN (accounting for 85% of all dependencies). The table also shows that for all language groups G1, G2, and G3, the f1 scores for NOUN, VERB and PROPN are generally higher than the f1 scores for

POS	$\mathcal{G}1$				$\mathcal{G}2$				$\mathcal{G}3$			
	freq%	prec.	rec.	f1	freq%	prec.	rec.	f1	freq%	prec.	rec.	f1
NOUN	43.9	85.4	88.6	87.0	43.5	77.3	81.2	79.2	34.5	67.1	71.0	69.0
VERB	32.0	83.5	83.6	83.6	35.4	74.9	77.9	76.4	41.3	63.8	66.5	65.1
PROPN	9.1	84.0	84.0	84.0	4.1	67.6	63.2	65.3	6.4	57.2	54.8	56.0
ADJ	4.5	76.2	72.4	74.3	5.7	75.7	56.0	64.4	5.8	64.9	49.1	55.9
PRON	1.4	79.3	68.3	73.4	1.4	81.5	61.4	70.0	2.2	65.2	49.1	56.0
NUM	1.2	77.2	72.4	74.7	1.0	52.0	41.8	46.3	0.7	62.5	54.7	58.3
ADV	1.0	54.0	39.0	45.3	1.5	56.5	27.2	36.7	1.2	44.1	25.8	32.6
ADP	0.6	39.8	6.5	11.2	0.3	25.0	0.9	1.7	0.3	40.5	8.3	13.8
SYM	0.3	79.0	81.1	80.1	0.1	41.5	66.3	51.0	0.1	55.3	52.2	53.7
DET	0.3	36.3	22.6	27.8	0.1	60.6	30.6	40.7	0.1	67.6	25.3	36.8
AUX	0.2	35.7	3.7	6.6	0.0	17.2	6.7	9.6	0.8	33.3	2.2	4.2
X	0.1	52.4	52.2	52.3	0.1	42.5	41.6	42.1	0.4	39.7	42.7	41.1
SCONJ	0.1	36.8	10.0	15.7	0.1	45.7	5.8	10.3	0.1	30.0	13.5	18.7
PART	0.1	26.7	3.0	5.4	0.1	15.9	4.3	6.8	0.1	26.7	36.8	30.9
CONJ	0.1	47.8	6.5	11.4	0.1	3.3	0.9	1.4	0.1	51.7	10.2	17.0
INTJ	0.0	52.4	47.8	50.0	0.0	20.0	7.1	10.5	0.1	44.2	43.0	43.6

Table 5.12: Precision, recall and f-score of unlabeled dependency attachment for different POS tags *as head* for three groups of languages for the universal dependencies experiments in Table 5.9: G1 (languages with UAS ≥ 80), G2 (languages with $70 \leq$ UAS < 80), G3 (languages with UAS < 70). The rows are sorted by frequency in the G1 languages.

other head POS tags.

Finally, Table 5.13 shows precision and recall for different dependency labels for the G1, G2 and G3 languages. We again see quite large differences in accuracy between different dependency labels. The G1 language dependencies, with the most frequent label `nmod`, has an F-score of 75.2. In contrast, the second most frequent label, `case`, has 93.7 F-score. Other frequent labels with low accuracy in the G1 languages are `advmod`, `conj`, and `cc`.

Dep.	$\mathcal{G}1$				$\mathcal{G}2$				$\mathcal{G}3$			
	freq%	prec.	rec.	f1	freq%	prec.	rec.	f1	freq%	prec.	rec.	f1
nmod	15.8	74.0	76.3	75.2	16.4	67.3	72.2	69.7	17.3	56.9	57.6	57.3
case	15.3	92.6	94.7	93.7	10.7	92.4	93.5	93.0	10.7	90.2	90.2	90.2
det	11.8	96.5	96.4	96.4	3.5	91.8	91.9	91.9	3.8	79.1	86.4	82.6
nsubj	6.5	85.3	86.8	86.0	7.5	75.5	73.5	74.5	7.8	61.0	63.2	62.1
amod	6.4	92.9	94.0	93.5	10.8	90.1	90.9	90.5	5.3	75.7	82.9	79.1
dobj	5.3	93.0	90.8	91.9	7.1	84.3	81.8	83.0	5.7	71.9	72.6	72.3
root	5.3	84.8	85.2	85.0	6.8	77.5	77.9	77.7	7.9	64.9	65.7	65.3
advmod	4.1	73.4	72.2	72.8	7.1	68.1	69.3	68.7	5.3	54.8	58.7	56.7
conj	4.0	60.4	68.1	64.0	5.8	50.2	56.6	53.2	4.2	41.3	48.1	44.5
cc	3.4	71.2	71.2	71.2	4.5	63.5	63.3	63.4	3.9	60.6	61.6	61.1
mark	3.3	85.1	87.0	86.0	2.2	76.2	79.6	77.9	3.4	70.9	71	71
acl	2.4	65.9	61.6	63.7	1.7	49.7	51.3	50.5	2.0	32.6	28.7	30.5
aux	2.2	91.5	93.6	92.5	1.2	86.8	91.1	88.9	2.2	66.4	78.2	71.8
name	1.9	86.5	86.2	86.4	1.3	75.3	72.1	73.6	0.8	27.8	45.1	34.4
cop	1.6	73.1	74.5	73.8	1.3	67.7	52.5	59.1	2.1	50.8	51.2	51
nummod	1.4	83.8	86.0	84.9	1.6	73.9	77.6	75.7	1.4	79.2	81.7	80.5
advcl	1.3	60.1	59.8	60.0	1.3	57.4	48.8	52.7	2.0	42.6	38.1	40.2
appos	1.3	73.9	64.9	69.1	0.8	51.2	48.9	50.0	0.5	31.3	32.1	31.7
mwe	0.9	57.7	15.6	24.6	0.5	66.2	15.1	24.6	0.3	31.9	15.6	20.9
xcomp	0.8	82.9	74.6	78.6	1.2	76.2	73.4	74.8	1.0	40.7	62.9	49.5
ccomp	0.8	72.8	70.8	71.8	0.6	63.1	64.1	63.6	1.2	42.8	40.3	41.5
neg	0.7	89.5	88.1	88.8	0.7	81.2	82.1	81.6	1.1	73.6	72	72.8
iobj	0.7	98.7	91.1	94.7	0.5	96.3	71.0	81.7	1.1	97.1	67.1	79.3
expl	0.6	90.9	84.7	87.7	0.7	87.3	86.8	87.1	0.1	62.5	45	52.3
auxpass	0.5	95.7	96.5	96.1	0.7	98.3	93.5	95.8	1.2	92.3	49.8	64.7
nsubjpass	0.5	94.6	89.9	92.2	0.7	96.1	85.0	90.2	0.6	94.4	67.2	78.5
parataxis	0.4	56.0	32.4	41.1	0.9	52.2	36.8	43.2	0.4	30.4	33.2	31.7
compound	0.4	74.2	66.2	69.9	0.6	72.5	63.6	67.8	4.4	84.7	51.6	64.1
csubj	0.2	77.0	52.5	62.4	0.3	88.1	57.3	69.4	0.2	45.9	31.3	37.2
dep	0.1	70.4	52.4	60.1	0.6	91.2	38.5	54.2	0.5	17.7	16.2	16.9
discourse	0.1	75.6	58.5	66.0	0.1	53.3	60.0	56.5	0.7	77.1	48.4	59.4
foreign	0.0	62.2	69.7	65.7	0.1	98.4	60.7	75.1	0.1	30.9	19.3	23.8
goeswith	0.0	35.7	29.4	32.3	0.1	75.0	19.6	31.1	0.0	26.1	16.7	20.3
csubjpass	0.0	100.0	73.9	85.0	0.0	93.3	71.2	80.8	0.1	87.5	19.7	32.2
list	0.0	–	–	–	0.0	77.0	45.6	57.3	0.1	71.4	18.5	29.4
remnant	0.0	90.0	25.7	40.0	0.0	27.3	10.2	14.8	0.1	92.3	11.8	20.9
reparandum	0.0	–	–	–	0.0	–	–	–	0.1	100.0	34.6	51.4
vocative	0.0	55.6	31.3	40.0	0.0	57.4	52.9	55.1	0.1	84.5	58.6	69.2
dislocated	0.0	88.9	30.8	45.7	0.0	54.5	60.0	57.1	0.0	92.0	48.9	63.9

Table 5.13: Precision, recall and f-score for different dependency labels for three groups of languages for the universal dependencies experiments in Table 5.9: G1 (languages with UAS $\geq$ 80), G2 (languages with 70 $\leq$ UAS $<$ 80), G3 (languages with UAS $<$ 70). The rows are sorted by frequency in the G1 languages.

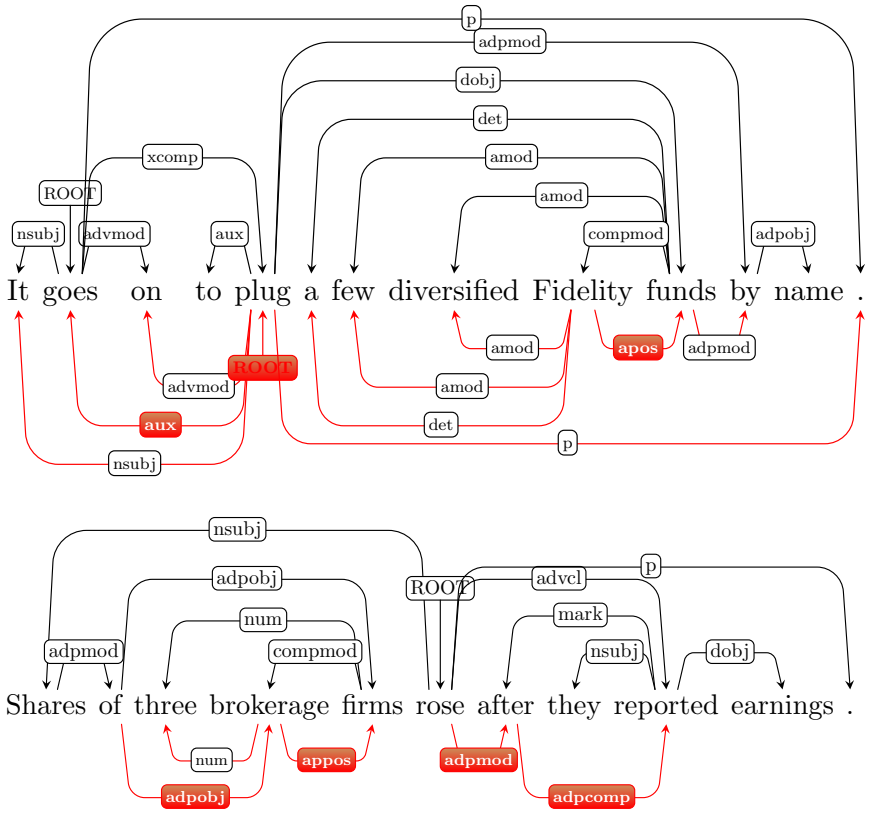
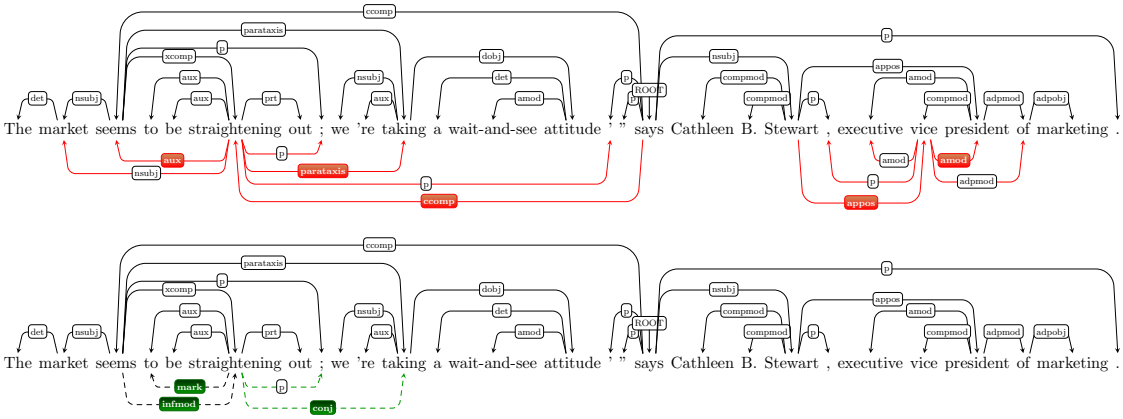


Figure 5.3: Two English sentences for which our method (either using the Bible or Europarl as a source of translation data), correctly recovers the full tree, but the baseline method gives an incorrect parse. The bottom edges show the incorrect dependency predictions made by the baseline parser: colored dependency edges show both labeled and unlabeled attachment errors while the white-colored dependencies show correct label assignment but wrong head selection.

Qualitative Analysis We find interesting examples for which our method predicts the correct tree, including some long sentences (Figures 5.4a and 5.4b). Except for very short sentences with basic grammatical structures, the baseline parser completely fails to recover the correct structure. In Figure 5.3, two examples are depicted for which the baseline parser has a very low accuracy while our method is fully accurate. In the first example, the baseline parser attaches the root node to a wrong verb ("plug" instead of "goes") and subsequently attaches incorrect dependents. In the first example in Figure 5.3, the baseline parser cannot find the head of the noun phrase ("diversified Fidelity funds"). Because

(a)



(b)

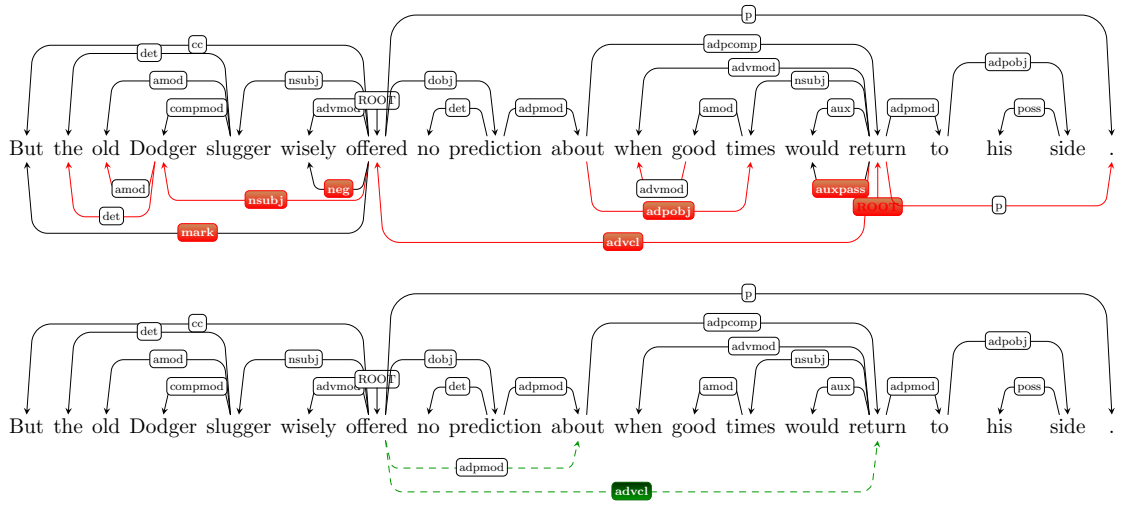


Figure 5.4: Two English sentences from the development data for which our method with the Europarl data correctly recovers the full tree, but the baseline model and the Bible-based model have some errors. The correct dependency parse is shown above each sentence. Incorrect dependencies from the baseline model are shown with solid red edges (top figure in each case) and incorrect dependencies from the Bible model are shown with dashed green edges (bottom figure in each case).

of the lack of lexical features, the baseline parser attaches the word “after” as a prepositional phrase instead of assigning it as a dependent for the adverbial clause (“they reported earnings”).

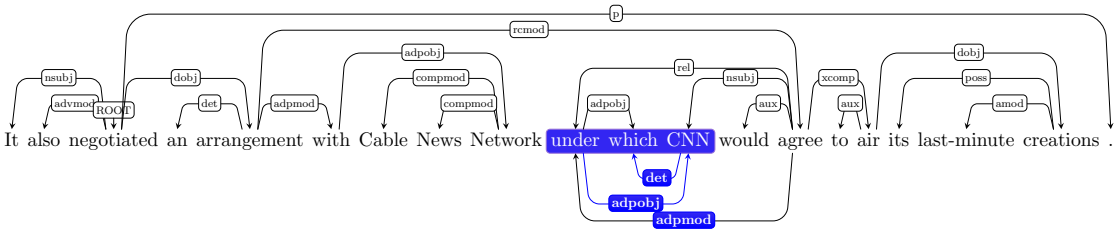


Figure 5.5: An example English sentence where the supervised parser could correctly recover the full tree but our model using Europarl has some incorrect dependency predictions (shown at the bottom of the tree).

Figure 5.4a shows two examples in which the parser trained using the Europarl data is the only model to parse the sentences perfectly. As expected, the baseline parser does a poor job in parsing these sentences. Surprisingly however, the parser from our approach using the Bible data is able to predict the majority of dependencies correctly. In the first example, it has two labeled errors and one punctuation attachment error[10] and just one attachment error: it attaches "taking" as a conjunction for "straightening" instead of attaching it as a "parataxis" for the word "seems". In the second example, there is a prepositional attachment error for the word "about" and consequently another attachment error because "about" is not the dependent of "prediction" in that sentence. In general, we can see that the parser trained with the Bible data is able to recover correct dependencies, largely beating the baseline.

Figure 5.5 shows an example in which the fully supervised parser is completely accurate but our best parser makes some errors. In this specific example, the word "which" has a determiner POS tag and is wrongly attached as a determiner to the next noun, while it should be a prepositional object. To compensate for that, the parser wrongly attaches "CNN" to "under" as a prepositional object. There is also a labeled attachment error for

[10]Punctuation is not included in evaluation.

the word “under”.

From the above examples, we can conclude that having more translation data helps distinguish between correct and incorrect attachments. Having more accurate projected trees help improve the syntactic ordering, and having more translation data gives rise to the quality of word clusters and lexical features.

5.6 Conclusions

We have described a method for cross-lingual syntactic transfer that is effective in a scenario where a large amount of translation data is not available. We have introduced a simple, direct method for deriving cross-lingual clusters, and for transferring lexical information across treebanks for different languages. Experiments with this method show that the method gives improved performance over previous work that makes use of Europarl, a much larger translation corpus.

Chapter 6

Low-Resource Syntactic Transfer with Unsupervised Source Reordering

6.1 Introduction

There has recently been a great deal of interest in cross-lingual transfer of dependency parsers, for which a parser is trained for a target language of interest using the treebanks in other languages. By using transfer methods, we can eliminate the need for the expensive and time-consuming task of treebank annotation for low-resource languages. In previous chapters, we have shown promising results on a set of European languages but it fails to give a promising performance in non-European languages, due to the lack of large parallel datasets for annotation projection, and significant word order differences with European languages for direct model transfer. This chapter considers the problem of dependency parser transfer in a scenario where a large parallel data is not available. Inspired by Chapter 5, we use the Bible data as our parallel data in addition to available monolingual raw text from Wikipedia and gold-standard treebanks $\mathcal{T}_i$—for $i = 1 \dots k$—in k source languages. The main goal of this work is to reorder gold-standard source treebanks $\mathcal{T}_i$ to make those treebanks syntactically more similar to the target language of interest. We use two different approaches for source treebank reordering: 1) reordering

based on dominant dependency directions according to the projected dependencies, 2) learning a classifier on the alignment data. We show that an ensemble of these methods with the baseline method leads to higher performance for the majority of datasets in our experiments.

The main contributions of this work are as follows:

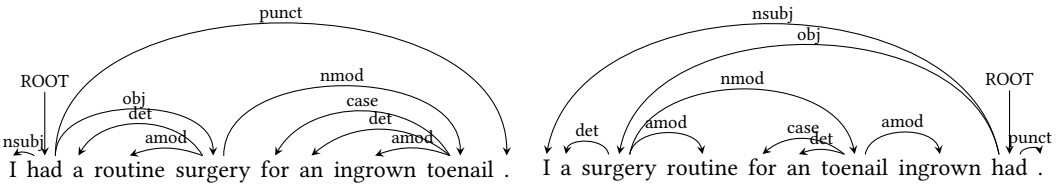
- We propose two different syntactic reordering methods based on the projected dependencies. The first model is based on the dominant dependency direction in the target language according to the projected dependencies. The second model learns a reordering classifier from the small set of aligned sentences in the Bible parallel data.
- We run an extensive set of experiments on 68 treebanks for 38 languages. We show that by just using the Bible data, we are able to achieve significant improvements in non-European languages while maintaining a high accuracy in European languages.
- We show that our transfer model outperforms a supervised model for the cases in which the gold-standard treebank is very small. This indicates the strength of our model when the language is truly low-resource.

Our final model improves over two strong baselines, two deep parsing models inspired by the non-neural state-of-the-art models of Chapters 4 and 5. Our final results improve the performance on non-European languages by an average UAS absolute improvement of 3.3% and LAS absolute improvement of 2.4%.

6.2 Background

Our transfer approach is inspired by the state-of-the-art transfer method in Chapter 5. That model trains on the concatenation of projected dependencies $\mathcal{P}$ and all of the source treebanks $\mathcal{T}_1 \ldots \mathcal{T}_k$. It applies the following techniques to achieve a highly accurate parser: 1) It uses cross-lingual word clusters as additional features for their shift-reduce parser, 2) It translates each source word to the target language using the dictionaries extracted from the Bible word alignments; though there are many cases for which a translation is not available for word. As a consequence, the treebank becomes a partially translated data, 3) It gradually adds the projected dependencies to the training data according to the density of projections using the self-training approach in the density-driven model in Chapter 4.

We make the following changes to the model in Chapter 5: 1) Instead of using a perceptron-based shift-reduce parser, we use our reimplementation of the state-of-the-art neural biaffine graph-based parser of Dozat and Manning [2016]; 2) Since we use a neural parser, instead of using cross-lingual word clusters, we use cross-lingual word embeddings that are trained using Word2Vec [Mikolov *et al.*, 2013a] with exactly the same pseudo-corpus approach of Chapter 5; and 3) Our graph-based parser is able to directly train on partial trees, thus we do not need to apply self-training in order to be able to train on projected dependencies; instead we directly train on the set of projected dependencies for which at least 80% of words have projected dependencies or there is a span of length $l \geq 5$ such that all words in that span achieve a projected dependency. This is the same as the definition of dense structures $\mathcal{P}_{80} \cup \mathcal{P}_{\geq 5}$ in Chapter 4.



(a) Original tree. (b) Persian-specific reordered tree.

Figure 6.1: An example of a gold-standard English tree that is reordered to look similar to the Persian syntactic order.

6.3 Approach

Training directly on a treebank from different languages might lead to a wrong model due to the differences in syntactic ordering between the source and target languages. This problem is not really harmful when the target language is a European language, since many of the gold-standard treebanks are from the European language family. Our experiments show that if one aims to train a transfer model for a non-European language, the model fails to be as accurate as that of a European language.

In general, for a head h that has c modifiers $m_1 \dots m_c$, we decide to put each of the dependents m_i on the left or right of the head h. After placing them the correct side of the head, the order in the original source sentence is preserved. Figure 6.1 shows a real example of an English tree that is reordered for the sake of Persian as our target language of interest. Here we see that we have a verb-ending sentence such that the nominal modifiers come after the noun. If one aims to translate this English sentence word by word, the reordered sentence gives a very good translation without any change in the sentence.

As mentioned earlier, we use two different approaches for source treebank reorder-

ing: 1) reordering based on dominant dependency directions according to the projected dependencies, 2) learning a classifier on the alignment data. Now we describe each of the mentioned methods.

6.3.1 Model 1: Reordering Based on Dominant Dependency Direction

We extract dominant dependency directions according to the projected dependencies $\mathcal{P}$ from the alignment data, and use the information for reordering source treebanks. Let the tuple $\langle i, m, h, r \rangle$ show the dependency of the m'th word in the i'th projected sentence for which the h'th word is the parent with the dependency label r. $\langle i, m, \texttt{NULL}, \texttt{NULL} \rangle$ shows an unknown dependency for the m'th: this occurs in projected dependency data for which some of the words in the target language do not achieve a projected dependency. We use the notation $h(i, m)$ to show the head index of the m'th word in the i'th sentence and $r(i, m)$ to show its dependency label.

Definition 6.1 *Dependency direction: $d(i, m)$ shows the dependency direction of the m'th modifier word in the i'th sentence:*

$$d(i,m) = \begin{cases} 1 & \texttt{if } h > m \\ -1 & \texttt{otherwise} \end{cases}$$

Definition 6.2 *Dependency direction proportion: Dependency direction proportion of each dependency label l with direction $d \in \{-1, 1\}$ is defined as:*

$$\alpha^{(\mathcal{P})}(l, d) = \frac{\sum_{i=1}^{|\mathcal{P}|} \sum_{m=1}^{|\mathcal{P}^{(i)}|} \mathbb{I}(r(i,m) = l \;\&\; d(i,m) = d)}{\sum_{i=1}^{|\mathcal{P}|} \sum_{m=1}^{|\mathcal{P}^{(i)}|} \mathbb{I}(r(i,m) = l)}$$

Definition 6.3 *Dominant dependency direction: For each dependency label l, we show the dominant dependency direction $\lambda^{(\mathcal{P})}(l) = d$ if $\alpha^{(\mathcal{P})}(l, d) > 0.75$. In cases where there is no dominant dependency direction, $\lambda^{(\mathcal{P})}(l) = 0$.*

We consider the following dependency labels for extracting dominant dependency direction information: nsubj, obj, iobj, csubj, ccomp, xcomp, obl, vocative, expl, dislocated, advcl, advmod, aux, cop, nmod, appos, nummod, acl, amod. We ignore most of the function word dependencies (such as "mark"), and other non-core dependencies such as conjunction.

Reordering condition Given a set of projections $\mathcal{P}$, we can calculate the dominant dependency direction information for the projections $\lambda^{(\mathcal{P})}$. Similar to the projected dependencies, we extract *supervised* dominant dependency directions from the gold-standard source treebank $\mathcal{D}$: $\lambda^{(\mathcal{D})}$. When we encounter a gold-standard dependency relation $\langle i, m, h, r \rangle$ in a source treebank $\mathcal{D}$, we change the direction if the following condition holds:

$$\lambda^{(\mathcal{D})}(r) \neq \lambda^{(\mathcal{P})}(r) \texttt{ and } \lambda^{(\mathcal{P})}(r) = -d(i, m)$$

In other words, if the source and target languages do not have the same dominant dependency direction for r and the dominant direction of the target language is the reverse of the current direction, we change the direction of that dependency.

6.3.2 Model 2: Reordering Classifier

We now describe our approach for learning a reordering classifier for a target language using the alignment data. Our method has two steps; the first step prepares the training data from the automatically aligned parallel data, and the second step learns a classifier from the training data.

Preparing Training Data from Alignments

The goal of this step is to reorder source language sentences in the parallel data via the guidance from automatic word alignments. The parallel data is the concatenation of parallel data from all source languages translated to the target language. We use the reordered data as training data for the classifier.

Given a parallel data set $(e^{(i)}, f^{(i)})$ for $i = 1 \dots n$ that contains pairs of source and target sentences $e^{(i)}$ and $f^{(i)}$, we extract one-to-one word alignments $a^{(i)} = a_1^{(i)} \dots a_{s_i}^{(i)}$ where s_i is the number of tokens in the source sentence $e^{(i)}$. In cases where $a_j^{(i)}$ is not `NULL`[1], $1 \leq a_j^{(i)} \leq t_i$ shows the index of the target sentence for which t_i is the length of the target sentence. By applying a simple heuristic based on alignment information $a^{(i)}$, we can create a new mapping $\mu^{(i)} = \mu_1^{(i)} \dots \mu_{s_i}^{(i)}$ that maps each index $1 \leq j \leq s_i$ in the original sentence to a unique index $1 \leq \mu_j^{(i)} \leq s_i$ in the reordered sentence. We assume that we have a supervised parser for the source language; thus we can obtain dependency information for each index $\langle i, m, h, r \rangle$. The decision about the direction of

[1] A `NULL` value indicates a missing alignment.

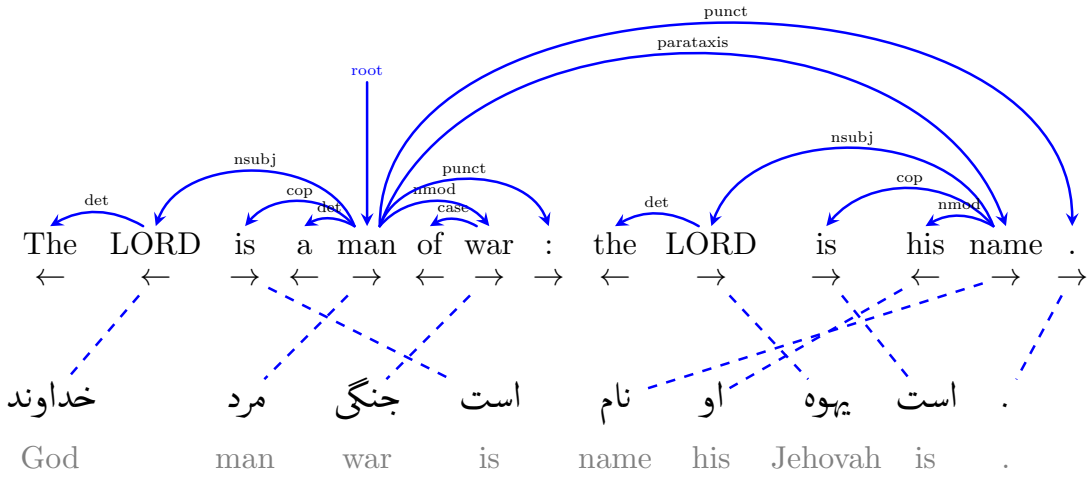


Figure 6.2: A reordering example from the Bible for English-Persian language pair. The Persian words are written from left to right for the ease of presentation. The arrows bellow the English words show the new dependency direction with respect to the word alignments to the Persian side. The reordered sentence would be "The LORD a man of war is : his name the LORD is .".

each dependency can be made based on the following condition:

$$d^*(i,m) = \begin{cases} 1 & \texttt{if } \mu_h^{(i)} > \mu_m^{(i)} \\ -1 & \texttt{otherwise} \end{cases}$$

In other words, we decide about the new order of a dependency according the mapping $\mu^{(i)}$.

Figure 6.2 shows an example for the data preparation step. As shown in the figure, the new directions for the English words are decided according to the Persian alignments.

Classifier

The reordering classifier uses the recurrent representation of the input sentence and decides about the new direction of each dependency. For a source sentence $e^{(i)} = e_1^{(i)} \dots e_{s_i}^{(i)}$ that belongs to a source language $\mathcal{L}$, we first obtain its recurrent representation $\eta^{(i)} = \eta_1^{(i)} \dots \eta_{s_i}^{(i)}$ by running a deep (3 layers) bi-directional LSTM [Hochreiter and Schmidhuber, 1997], where $\eta_j^{(i)} \in \mathbb{R}^{d_h}$. For every dependency tuple $\langle i, m, h, r \rangle$, we use a multi-layer Perceptron (MLP) to decide about the new order $dir \in \{-1, 1\}$ of the m'th word with respect to its head h:

$$p(dir|i, m, h, r) = \texttt{softmax}(W\phi(i, m, h, r))$$

where $W \in \mathbb{R}^{2 \times d_\phi}$ and $\phi(i, m, h, r) \in \mathbb{R}^{d_\phi}$ is as follows:

$$\phi(i, m, h, r) = \texttt{relu}(Hq(i, m, h, r) + B)$$

where `relu` is the rectified linear unit activation [Nair and Hinton, 2010], $H \in \mathbb{R}^{d_\phi \times d_q}$, $B \in \mathbb{R}^{d_\phi}$, and $q(i, m, h, r) \in \mathbb{R}^{d_q}$ is as follows:

$$q(i, m, h, r) = [\eta_m^{(i)}; \eta_h^{(i)}; R[r]; \Lambda[\mathbb{I}(h > m)]; L[\mathcal{L}]]$$

where $\eta_m^{(i)}$ and $\eta_h^{(i)}$ are the recurrent representations for the modifier and head words respectively, R is the dependency relation embedding dictionary that embeds every dependency relation to a $\mathbb{R}^{d_r}$ vector, Λ is the direction embedding for the original position

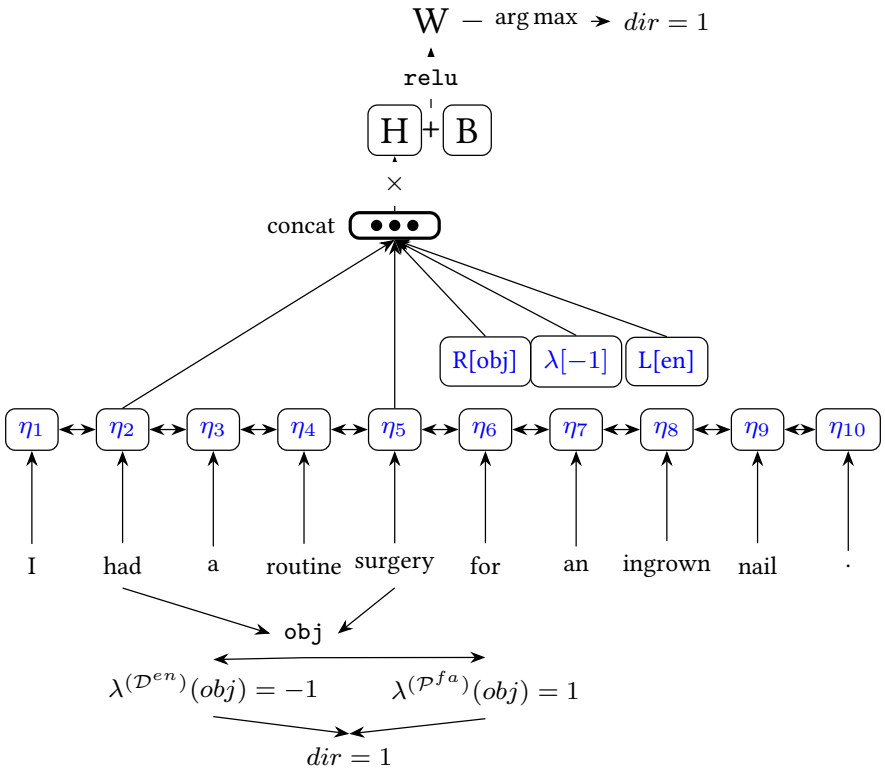


Figure 6.3: Two different approaches for reordering the dependency order for the example in Figure 6.1. The reordering classifier is shown on top, for the dependency relation between the words "had" and "surgery" with an "obj" relation. At the bottom, the reordering model based on dominant dependency direction information is shown.

of the head with respect to its head and embeds each direction to a 2-dimensional vector, and L is the language embedding dictionary that embeds the source language id $\mathcal{L}$ to a $\mathbb{R}^{d_L}$ vector.

Input Embedding Layer The input to the recurrent layer is the concatenation of the following vectors:

- Original source word w: we use the sum of two vectors for the original source word. The first vector is the pretrained cross-lingual word embedding $C[w] \in \mathbb{R}^{d_w}$ that is fixed during training, and the second is a randomly initialized word embedding vector $E[w] \in \mathbb{R}^{d_w}$.

- Translation of the source word w to the target language ($t(w)$): This translation word comes from the extracted dictionary from the Bible alignments. We use the same fixed and updateable word embedding dictionary to embed this translation to a vector. In cases where a translation does not exist, we set $t(w) = w$.
- Part-of-speech (POS) tag p: we embed the POS tag to a randomly initialized POS embedding vector $T[p] \in \mathbb{R}^{d_p}$.

Figure 6.3 shows a graphical depiction of the two reordering models that we use in this work.

6.4 Experiments

Datasets and Tools We use 68 datasets from 38 languages in the Universal Dependencies corpus version 2.0 [Nivre *et al.*, 2017]. The languages are Arabic (ar), Bulgarian (bg), Coptic (cop), Czech (cs), Danish (da), German (de), Greek (el), English (en), Spanish (es), Estonian (et), Basque (eu), Persian (fa), Finnish (fi), French (fr), Hebrew (he), Hindi (hi), Croatian (hr), Hungarian (hu), Indonesian (id), Italian (it), Japanese (ja), Korean (ko), Latin (la), Lithuanian (lt), Latvian (lv), Dutch (nl), Norwegian (no), Polish (pl), Portuguese (pt), Romanian (ro), Russian (ru), Slovak (sk), Slovenian (sl), Swedish (sv), Turkish (tr), Ukrainian (uk), Vietnamese (vi), and Chinese (zh).

We use the Bible data from Christodouloupoulos and Steedman [2014] for the 38 languages. We extract word alignments using Giza++ [Och and Ney, 2003]. Following Chapter 4, we obtain intersected alignment and apply POS consistency to filter potentially incorrect alignments. We use the Wikipedia dump data to extract monolingual data for the

languages in order to train monolingual embeddings. We follow the method in Chapter 5 to use the extracted dictionaries from the Bible and monolingual text from Wikipedia to create cross-lingual word embeddings. We use the UDPipe pretrained models [Straka and Straková, 2017] to tokenize Wikipedia, and a reimplementation of the Perceptron tagger of Collins [2002] to achieve automatic POS tags.

Supervised Parsing Models We trained our supervised models on the union of all datasets in a language for obtaining the supervised parsers. It is worth noting that there are two major changes that we make to the neural parser of Dozat and Manning [2016] in our implementation using the Dynet library [Neubig *et al.*, 2017]: first, we add a one-layer character BiLSTM for representing the character information for each word. The final character representation is obtained by concatenating the forward representation of the last character and the backward representation of the first character. The concatenated vector is summed with the randomly initialized as well as fixed pre-trained cross-lingual word embedding vectors. Second, inspired by Weiss *et al.* [2015], we maintain the moving average parameters to obtain more robust parameters at decoding time.

We excluded the following languages from the set of source languages for *annotation projection* due to their low supervised accuracy: Estonian, Hungarian, Korean, Latin, Lithuanian, Latvian, Turkish, Ukrainian, Vietnamese, and Chinese.

6.4.1 Baseline Transfer Models

We use two baseline models: 1) Annotation projection: This model only trains on the projected dependencies. 2) Annotation projection + direct transfer: Inspired by the state-

of-the-art method in Chapter 5, we train on a concatenation of source treebanks from all 37 languages (i.e. excluding the target language), and the projected dependencies in the target language. To speed up training, we sample at most thousand sentences from each treebank, comprising a training data of about 37K sentences.

6.4.2 Reordering Ensemble Model

We noticed that our reordering models perform better in non-European languages, and perform slightly worse in European languages. We use the following ensemble model to make use of all of the three models (annotation projection + direct transfer, and the two reordering models), to make sure that we always obtain an accurate parser.

The ensemble model is as follows: given three output trees for the i'th sentence $\langle i_j, m, h_j, r_j \rangle$ for $j = 1, 2, 3$ in the target language $\mathcal{L}$, where the first tuple ($j = 1$) belongs to the baseline model, the second ($j = 2$) and third ($j = 3$) belong to the two reordering models, we weight each dependency edge with respect to the following conditions:

$$\omega(m, h, r) = z(m, h, r) \cdot \sum_{j=1}^{3} c(j, \mathcal{L}) \cdot \mathbb{I}(\langle i_j, m, h, r \rangle)$$

where $c(j, \mathcal{L})$ is a coefficient that puts more weight on the first or the other two outputs depending on the target language family:

$$c(j, \mathcal{L}) = \begin{cases} 2 & \texttt{if } j = 1 \text{ \& } \mathcal{L} \texttt{ is European} \\ 2 & \texttt{if } j > 1 \text{ \& } \mathcal{L} \texttt{ is not European} \\ 1 & \texttt{otherwise} \end{cases}$$

and $z(m, h, r)$ is a simple weighting depending on the dominant order information:

$$z(m,h,r) = \begin{cases} 1 & \text{if } direction(\langle m,h \rangle) = -\lambda^{(\mathcal{P})}(r) \\ 3 & \text{if } direction(\langle m,h \rangle) = \lambda^{(\mathcal{P})}(r) \\ 2 & \text{otherwise } (\lambda^{(\mathcal{P})}(r) = 0) \end{cases}$$

We run the Eisner first-order graph-based algorithm [Eisner, 1996] on top of the edge weights ω to extract the best possible tree.

6.4.3 Parameters

We run all of the transfer models with 4000 mini-batches, in which each mini-batches contains approximately 5000 tokens. We follow the same parameters as in Dozat and Manning [2016] and use a dimension of 100 for character embeddings. For the reordering classifier, we use the Adam algorithm [Kingma and Ba, 2014] with default parameters to optimize the log-likelihood objective. We filter the alignment data to keep only those sentences for which at least half of the source words have an alignment. We randomly choose 1% of the reordering data as our heldout data. Table 6.1 shows the parameter values that we use in the reordering classifier.

6.4.4 Results

Table 6.2 shows the results on the Universal Dependencies corpus [Nivre *et al.*, 2017]. As shown in the table, the algorithm based on the dominant dependency direction improves the accuracy on most of the non-European languages and performs slightly worse than

Variable	Notation	Size
Word embedding	d_w	100
POS embedding	d_p	100
Bi-LSTM	d_h	400
Dependency relation embedding	d_r	50
Language ID embedding	d_L	50
Hidden layer	d_ϕ	200
Number of BiLSTM layers	–	3
Mini-batch size (tokens)	–	~ 1000

Table 6.1: Parameter values in the reordering classifier model.

the baseline model in the European languages. The ensemble model, besides its simplicity, improves over the baseline in most of the languages, leading to an average UAS improvement of 0.9 for all languages and 3.3 for non-European languages. This improvement is very significant in many of the non-European languages; for example, from an LAS of 37.6 to 52.7 in Coptic, from a UAS of 44.9 to 53.7 in Basque[2], from a UAS of 40.6 to 47.0 in Chinese. Our model also outperforms the supervised models in Ukrainian and Latvian. That is an interesting indicator that for cases that the training data is very small for a language (37 sentences for Ukrainian, and 153 sentences for Latvian), our transfer approach outperforms the supervised model. We believe that developing a transfer model gives a higher accuracy than a supervised model from just a few number of sentences.

6.5 Analysis

In this section, we briefly describe our analysis based on the results in the ensemble model and the baseline. Table 6.3 shows the size of projected dependencies that we use in our experiments. As shown in the table, for some languages such as Coptic, the number

[2] Although Basque is geographically in the Europe, it is regarded as an isolated non-European language.

$\mathcal{L}$	Data	Baselines				Reordering								Supervised	
		Projection		Direct+Proj		Dominant		Classifier		Ensemble		**Difference**			
		UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
cop	cop	2.0	0.4	58.5	37.6	69.1	52.7	65.5	50.9	69.6	52.7	11.1	15.1	86.9	80.1
eu	eu	39.5	22.0	44.9	29.0	53.7	34.0	48.6	32.2	53.7	34.4	8.8	5.4	81.9	75.9
zh	zh	23.6	10.8	40.6	17.8	47.3	25.4	45.4	23.5	47.0	25.6	6.4	7.8	81.1	74.8
vi	vi	44.6	26.8	51.2	33.6	55.3	34.5	50.4	34.2	55.1	34.5	4.0	0.9	66.2	56.7
tr	tr_pud	44.7	19.9	46.6	24.5	50.3	26.7	42.6	22.0	49.9	26.3	3.4	1.8	56.7	31.7
fa	fa	54.4	46.2	61.8	53.0	64.3	54.7	63.0	53.4	65.1	55.4	3.3	2.4	87.8	83.6
ar	ar_pud	60.3	44.2	65.2	50.5	68.2	52.0	66.5	51.4	68.3	52.3	3.2	1.8	71.9	58.8
id	id	59.9	42.8	72.1	56.0	73.6	56.5	72.9	56.8	74.6	56.7	2.5	0.6	84.8	77.4
tr	tr	44.6	23.9	46.6	29.3	48.9	30.6	44.9	26.6	49.0	30.0	2.4	0.7	64.2	52.5
he	he	63.1	46.9	70.4	55.4	72.4	54.9	71.6	55.7	72.7	55.4	2.3	0.0	88.2	82.4
ar	ar	49.5	36.8	58.9	46.8	60.8	48.3	59.2	46.9	61.2	48.8	2.3	2.0	85.6	78.9
ja	ja	54.8	38.9	65.2	46.5	65.9	46.8	64.1	44.8	66.6	46.8	1.4	0.3	94.5	92.7
ja	ja_pud	58.6	44.1	66.8	51.5	67.4	51.5	64.7	48.4	67.9	51.9	1.1	0.4	94.7	93.5
ko	ko	34.3	17.3	43.0	24.8	43.5	23.8	43.6	26.4	44.1	24.7	1.1	-0.2	76.2	69.9
hi	hi_pud	53.4	43.3	58.2	47.6	58.3	47.5	58.8	48.5	58.9	48.2	0.6	0.6	70.2	55.6
lt	lt	60.6	42.5	66.6	49.5	63.7	46.8	64.6	46.0	67.2	49.9	0.6	0.4	54.8	40.0
cs	cs_cac	33.9	14.8	76.2	66.9	76.3	66.7	75.2	65.8	76.7	67.4	0.5	0.6	92.1	88.3
cs	cs_cltt	13.7	5.1	69.4	59.7	69.7	59.5	66.6	57.8	70.0	60.3	0.5	0.6	88.9	84.9
fr	fr_partut	81.6	75.2	84.3	77.8	84.9	78.4	84.4	78.1	84.8	78.4	0.5	0.5	90.0	85.1
hr	hr	70.6	59.9	79.4	69.9	79.3	69.5	77.9	67.7	79.9	70.1	0.5	0.2	86.8	80.4
el	el	62.3	47.2	75.9	63.9	75.4	63.1	74.7	62.5	76.4	64.1	0.4	0.2	88.0	84.4
ru	ru_pud	75.7	65.8	81.1	72.2	80.9	72.2	79.9	70.7	81.5	72.7	0.4	0.5	86.5	74.1
de	de	71.4	62.3	75.4	67.1	75.6	67.1	75.5	66.4	75.8	67.3	0.4	0.2	85.9	81.2
fr	fr	80.2	72.9	83.0	75.9	82.9	75.9	83.3	75.9	83.4	76.2	0.4	0.3	90.4	86.9
cs	cs	33.9	14.5	74.6	65.3	74.1	64.4	73.0	63.7	75.0	65.8	0.4	0.5	92.5	89.1
fi	fi_pud	64.1	52.5	67.2	55.0	66.8	55.0	67.3	55.1	67.5	55.5	0.4	0.5	81.6	74.5
nl	nl	59.2	48.2	68.5	55.2	69.6	55.9	68.3	54.4	68.8	55.4	0.4	0.1	83.5	76.6
ru	ru	68.9	59.4	75.1	63.9	75.4	64.1	74.5	63.4	75.5	64.3	0.4	0.4	85.7	77.9
la	la_ittb	56.4	42.5	63.0	49.2	63.2	49.5	62.4	48.7	63.3	49.7	0.4	0.4	89.5	86.5
no	no_nynorsk	72.5	62.9	76.4	68.1	76.5	68.0	76.1	67.3	76.8	68.4	0.3	0.3	91.3	88.8
uk	uk	55.1	36.9	64.3	46.1	64.5	45.7	61.7	42.2	64.6	45.9	0.3	-0.2	43.3	22.1
bg	bg	80.4	69.4	83.8	73.8	84.0	73.8	83.1	73.0	84.1	73.9	0.3	0.1	90.9	86.0
en	en_lines	75.6	66.5	77.8	69.0	78.9	69.9	77.0	68.2	78.1	69.2	0.3	0.3	85.8	80.5
fi	fi_ftb	63.9	46.5	66.0	48.3	65.8	47.6	65.7	48.1	66.3	48.4	0.3	0.1	81.1	74.4
ru	ru_syntagrus	69.4	57.5	73.9	62.2	73.8	61.8	73.2	61.2	74.2	62.3	0.3	0.1	91.3	88.3
fi	fi	60.6	48.7	64.6	51.9	63.5	51.2	63.7	51.1	64.8	52.0	0.2	0.1	80.9	73.5
hu	hu	58.3	41.1	67.8	49.0	67.8	48.9	65.8	47.4	68.0	49.1	0.2	0.1	78.2	69.8
cs	cs_pud	35.7	16.6	77.5	69.3	76.7	67.6	76.2	67.7	77.7	69.4	0.2	0.2	89.9	84.4
nl	nl_lassysmall	61.8	52.1	73.9	63.4	73.8	62.8	73.0	61.9	74.0	63.3	0.2	0.0	91.3	87.3
sl	sl_sst	58.4	44.1	61.7	47.7	61.6	47.7	61.6	47.4	61.9	48.0	0.2	0.3	70.6	63.6
en	en_pud	73.5	65.5	75.9	69.3	77.1	69.9	74.5	67.7	76.0	69.4	0.2	0.2	88.3	84.2
de	de_pud	74.1	65.3	77.8	68.9	77.7	68.5	76.9	67.4	78.0	68.8	0.1	0.0	85.9	79.0
pl	pl	77.6	64.7	79.9	67.9	79.7	67.5	79.5	67.2	80.1	68.0	0.1	0.1	89.4	83.3
sv	sv_lines	77.2	67.7	81.1	71.6	80.7	71.1	80.1	70.4	81.3	71.7	0.1	0.1	86.9	81.5
en	en	70.1	61.6	72.8	64.6	73.5	65.2	71.6	63.5	72.9	64.8	0.1	0.3	88.2	84.8
es	es	78.5	68.0	83.1	73.8	83.2	73.8	82.3	72.8	83.2	73.9	0.1	0.1	89.3	83.9
sv	sv	75.3	67.0	79.0	70.9	78.8	70.9	78.2	70.0	79.1	71.0	0.1	0.1	86.7	82.3
en	en_partut	72.0	65.3	77.4	71.1	78.0	71.1	76.3	69.9	77.5	71.2	0.1	0.1	88.4	83.0
sv	sv_pud	75.9	67.4	80.5	72.1	80.2	72.0	79.2	71.0	80.6	72.1	0.1	0.0	84.0	77.6
it	it	81.3	74.4	85.0	79.0	85.4	79.5	84.4	78.1	85.1	79.1	0.1	0.0	92.1	89.5
lv	lv	59.0	43.6	63.3	47.2	62.1	45.6	60.7	44.7	63.3	47.0	0.1	-0.1	71.3	61.2
ro	ro	72.8	59.0	76.8	64.2	76.2	63.7	75.3	63.2	76.8	64.3	0.1	0.1	89.6	83.5
et	et	63.1	40.8	66.7	46.0	65.6	45.8	65.5	45.2	66.7	46.1	0.1	0.2	71.6	60.7
pt	pt	62.6	50.7	84.1	76.9	83.7	76.6	83.4	76.2	84.2	77.1	0.0	0.2	90.6	85.6
pt	pt_br	60.6	47.7	81.3	71.2	80.8	70.8	80.8	70.4	81.4	71.3	0.0	0.2	91.6	89.0
no	no_bokmaal	78.0	70.5	80.5	73.2	80.6	73.4	79.7	72.1	80.5	73.2	0.0	0.0	92.1	89.7
fr	fr_pud	81.0	72.8	83.7	75.7	84.2	76.2	83.3	75.2	83.7	75.7	0.0	0.0	89.1	83.8
es	es_pud	81.3	70.9	84.3	75.6	84.6	76.0	83.6	74.6	84.3	75.7	0.0	0.1	89.1	80.8
it	it_pud	83.8	76.0	87.3	81.3	87.5	81.3	86.5	79.9	87.3	81.2	0.0	-0.1	91.9	88.4
fr	fr_sequoia	79.1	73.0	82.2	76.4	81.6	75.8	81.9	76.0	82.2	76.4	0.0	0.0	90.4	86.7
la	la	49.2	33.6	53.9	36.2	51.3	33.3	54.0	35.5	53.9	35.4	0.0	-0.8	67.2	54.5
sl	sl	76.4	67.6	82.1	74.2	81.3	73.0	81.3	73.3	82.0	74.2	-0.1	0.0	88.9	85.4
es	es_ancora	77.7	66.2	82.4	72.7	82.0	72.2	81.4	71.3	82.3	72.5	-0.1	-0.3	91.1	87.0
da	da	70.7	61.7	75.7	67.4	75.3	66.7	74.6	66.2	75.6	67.2	-0.1	-0.2	83.1	79.3
pt	pt_pud	63.5	51.8	82.7	75.8	82.5	75.8	82.0	74.8	82.6	75.7	-0.2	-0.1	86.4	78.5
la	la_proiel	59.2	46.2	61.5	47.4	60.9	47.1	60.2	46.0	61.3	47.2	-0.2	-0.2	80.9	75.4
sk	sk	73.6	63.8	78.7	71.0	78.0	69.8	77.1	68.7	78.5	70.7	-0.2	-0.3	83.5	77.9
hi	hi	58.7	47.2	63.7	50.0	62.3	49.0	62.6	49.3	62.7	49.4	-1.0	-0.6	94.2	90.4
Avg	All	62.0	49.7	71.2	59.3	71.7	59.6	70.6	58.7	**72.1**	**60.0**	0.9	0.7	83.9	77.3
Avg	Non-EU	46.6	32.0	57.1	40.9	60.1	43.1	57.8	41.9	**60.4**	**43.3**	3.3	2.4	80.3	72.2

Table 6.2: Dependency parsing results, in terms of unlabeled attachment accuracy (UAS) and labeled attachment accuracy (LAS) after ignoring punctuations, on the Universal Dependencies v2 [Nivre *et al.*, 2017] test sets using supervised part-of-speech tags. The results are sorted by their "difference" between the ensemble model and the baseline. The rows for non-European languages are highlighted with cyan. The rows that are highlighted by pink are the ones that the transfer model outperforms the supervised model.

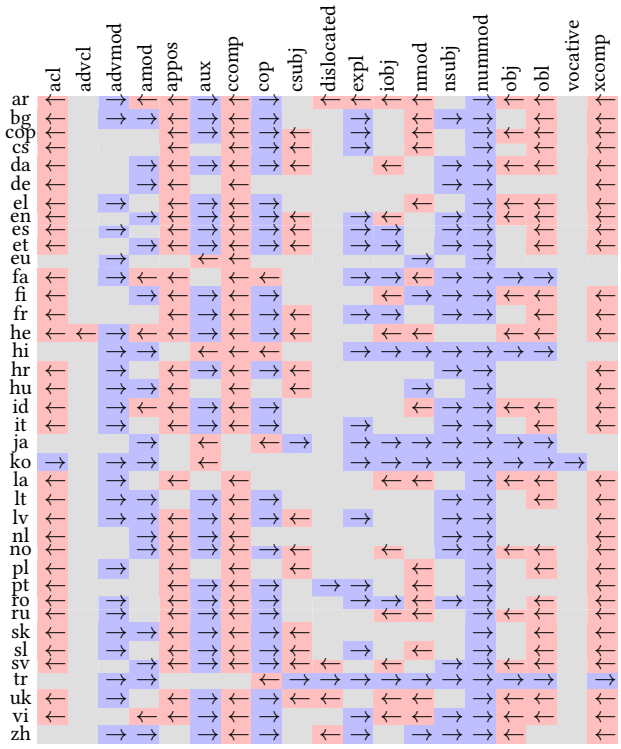


Figure 6.4: A graphical depiction of dominant dependency direction for different languages (the gray color shows no dominant order) based on projected dependencies in the Bible data.

of dense projected dependencies is too small such that the parser gives a worse learned model than a random baseline.

The dominant dependency direction model generally performs better than the classifier. Our manual investigation shows that the classifier kept many of the dependency directions unchanged, while the dominant dependency direction model changed the direction. Therefore, the dominant direction model gives a higher recall with the expense of loosing precision. The training data for the reordering classifier is very noisy due to wrong alignments. We believe that the dominant direction model, besides its simplicity, is a more robust classifier for reordering, though the classifier is helpful in an ensemble setting. Figure 6.4 shows the dominant dependency directions of different dependency

relations for the languages in our experiments. As shown in the figure, the dominant dependency direction for many of the relations is not necessarily the same such as in “aux”, “advcl”, and “obj”.

Tables 6.4 and 6.5 show the differences in parsing f-score of dependency relations, and part-of-speech as head. As we see in table 6.4, for noun modifiers (“nmod”) we see more consistent improvements, while in other relations such as “nsubj”, there is less consistency. Table 6.5 shows that we are able to improve the head dependency relation for the three most important head POS tags in the dependency grammar. We see that this improvement is consistent for non-European languages.

6.6 Conclusion

We have described a cross-lingual dependency transfer method that takes into account the problem of word order differences between the source and target languages. We have shown that applying projection-driven reordering improves the accuracy of non-European languages while maintaining the high accuracies in European languages.

L.	Sen.	Len.	Density
ar	985	19.1	0.58
bg	11546	27.7	0.75
cop	2	2.0	1.00
cs	138	25.0	0.41
da	9955	23.9	0.73
de	15181	28.0	0.75
el	3428	27.7	0.64
en	44576	28.7	0.85
es	20811	26.8	0.80
et	881	22.7	0.59
eu	171	20.9	0.62
fa	3688	27.4	0.61
fi	10240	25.1	0.74
fr	20316	28.1	0.82
he	1623	21.3	0.63
hi	2539	30.0	0.54
hr	9363	25.4	0.73
hu	1165	29.2	0.57
id	1557	27.4	0.53
it	16599	27.4	0.77
ja	604	33.5	0.48
ko	447	15.2	0.61
la	10442	19.9	0.77
lt	3945	22.5	0.68
lv	2774	22.2	0.76
nl	3155	30.7	0.55
no	20807	27.1	0.83
pl	10875	25.9	0.75
pt	140	32.6	0.49
ro	11443	28.7	0.72
ru	13853	24.8	0.77
sk	16269	25.6	0.78
sl	14139	26.6	0.76
sv	18373	27.7	0.79
tr	310	20.7	0.52
uk	463	23.6	0.63
vi	773	31.5	0.53
zh	181	33.0	0.46
Avg.	7994	25	0.7

Table 6.3: Sizes of the projected dependencies in the Bible data. The second column shows the number of sentences, the third column shows the average sentence length, and the last column shows the proportion of projected dependencies with respect to the number of words in the sentences.

lang	data	acl	advmod	amod	conj	nmod	nsubj	obj	obl	root
ar	ar	33.6/35.4	66.0/66.3	80.4/84.0	41.5/42.7	60.8/63.8	54.9/41.5	54.6/69.8	48.8/52.6	29.3/30.2
ar	ar_pud	37.1/42.3	62.3/61.0	74.1/80.9	48.2/49.2	66.0/69.1	61.3/58.6	69.8/78.7	51.1/55.1	63.5/67.4
bg	bg	61.7/62.9	77.0/77.6	92.1/92.3	67.8/67.4	75.4/76.5	84.9/84.5	92.8/92.9	79.4/79.3	82.3/81.9
cop	cop	12.1/28.6	64.5/63.5	50.0/80.0	32.1/22.2	44.0/52.9	82.2/83.2	66.7/64.5	56.3/67.8	60.6/60.6
cs	cs	49.7/50.2	68.9/69.1	89.4/89.8	57.1/57.6	63.6/65.4	71.3/72.1	78.2/78.2	70.5/70.8	79.8/80.0
cs	cs_cac	52.5/53.5	66.7/65.6	89.1/89.5	61.9/61.9	66.1/67.9	72.6/74.0	79.9/79.9	72.2/72.8	79.6/80.1
cs	cs_cltt	49.6/49.3	44.9/45.5	89.4/89.8	46.7/46.2	60.0/63.2	70.3/71.5	76.8/76.9	57.5/59.2	76.7/76.1
cs	cs_pud	53.2/53.9	71.4/70.8	89.5/89.9	64.4/64.9	62.1/63.8	76.4/76.9	83.3/83.6	72.5/72.6	83.5/83.2
da	da	51.9/52.9	70.8/69.6	84.5/84.9	69.1/68.8	62.6/62.0	83.7/83.9	85.5/86.0	60.4/61.0	75.0/73.5
de	de	41.8/39.4	63.9/64.3	85.7/87.2	62.0/62.5	64.8/65.5	76.2/76.7	76.8/77.8	67.5/69.3	79.3/79.2
de	de_pud	52.6/51.1	64.9/66.0	86.5/87.5	61.8/62.9	70.8/70.3	81.1/82.2	80.8/81.5	68.5/68.7	83.0/82.8
el	el	40.0/41.9	73.1/73.1	83.7/85.3	55.8/55.6	59.0/60.8	71.0/71.9	85.4/85.0	67.5/68.2	77.9/77.4
en	en	49.2/50.3	77.2/77.0	76.8/77.8	64.4/64.5	54.2/54.3	80.7/80.6	80.5/81.1	64.6/65.2	80.9/80.7
en	en_lines	59.9/61.4	72.2/72.5	83.4/83.4	63.8/64.8	64.5/64.9	83.1/83.4	85.3/86.2	74.9/75.3	81.8/82.0
en	en_partut	51.6/52.9	72.9/73.6	84.1/85.7	60.1/60.3	58.7/58.7	82.0/81.5	80.7/81.5	70.9/71.6	87.0/86.7
en	en_pud	56.3/58.1	74.7/74.8	79.9/80.9	64.7/65.0	65.7/65.9	81.8/82.3	82.3/83.1	70.0/70.0	86.9/86.6
es	es	59.7/60.4	76.6/77.3	88.3/88.8	60.6/61.5	77.8/78.2	80.8/81.3	91.1/91.4	78.2/78.4	82.2/80.5
es	es_ancora	58.0/57.2	72.5/71.2	88.7/88.8	64.2/64.1	79.0/79.3	86.2/86.1	86.9/87.0	74.2/74.2	86.4/85.2
es	es_pud	57.9/58.7	72.6/70.6	91.0/90.9	68.8/69.2	81.0/81.1	85.7/85.8	93.5/93.3	74.1/74.9	84.9/85.5
et	et	48.5/47.3	69.3/68.9	73.6/74.5	54.1/54.7	63.6/62.9	75.5/75.1	76.6/76.8	44.5/42.6	73.1/73.7
eu	eu	2.6/4.9	56.3/57.2	64.3/62.7	36.9/40.9	19.6/49.1	51.5/61.9	44.9/60.3	13.1/26.5	54.0/58.7
fa	fa	39.0/46.3	40.9/45.8	79.3/81.7	42.2/44.6	64.1/66.8	45.4/52.5	58.2/67.0	55.7/64.3	59.2/63.3
fi	fi	46.1/44.1	61.8/62.0	78.7/80.0	60.1/60.8	54.1/56.3	70.7/71.8	70.1/68.9	58.9/57.3	72.9/72.6
fi	fi_ftb	38.5/37.0	62.2/62.8	77.4/78.1	56.4/56.9	42.3/43.8	72.3/72.5	73.4/72.3	75.3/76.1	72.0/72.5
fi	fi_pud	50.2/50.5	65.4/65.9	79.2/80.0	63.5/64.6	58.7/59.7	72.7/73.3	70.4/70.8	64.4/63.8	76.9/77.8
fr	fr	53.8/54.7	75.0/75.6	88.5/88.5	62.3/62.2	76.5/77.2	85.8/85.8	89.4/90.6	69.3/70.6	81.0/81.3
fr	fr_partut	57.3/61.1	77.8/78.3	87.8/87.4	61.9/62.2	76.2/77.0	84.2/84.4	90.9/91.2	73.1/74.3	84.0/85.4
fr	fr_pud	66.3/66.5	74.2/73.8	89.9/90.1	67.3/67.5	77.6/77.6	83.2/83.5	92.7/92.8	72.8/73.3	84.6/83.9
fr	fr_sequoia	53.2/52.7	71.6/72.1	90.5/90.4	60.8/60.7	77.0/76.9	82.9/81.9	91.6/91.8	70.9/71.1	84.8/85.4
he	he	40.7/41.6	53.1/54.9	77.1/78.8	48.2/51.1	72.1/75.3	60.2/65.3	76.8/77.2	69.2/71.1	63.3/63.6
hi	hi	43.5/44.6	58.0/50.4	75.4/75.6	40.5/38.3	65.7/65.4	47.3/49.3	70.3/69.8	50.8/53.4	68.8/68.2
hi	hi_pud	28.1/28.1	50.6/52.3	75.6/75.5	39.5/40.6	64.8/65.7	49.5/50.8	59.4/61.2	48.9/51.2	56.5/58.9
hr	hr	56.5/56.5	71.2/71.7	90.4/90.5	64.1/65.4	76.5/77.1	78.9/79.5	87.3/88.0	73.7/74.5	79.3/81.5
hu	hu	32.8/30.2	69.6/69.8	82.4/83.3	52.1/52.5	44.4/49.0	71.5/72.5	79.0/78.8	72.9/73.9	69.3/69.0
id	id	45.6/45.9	77.9/79.9	59.2/74.0	56.1/57.5	73.3/77.8	80.0/82.2	81.1/85.3	73.4/77.4	82.9/83.7
it	it	65.4/64.4	74.3/76.1	87.3/88.2	61.2/60.4	80.5/80.5	80.1/78.6	91.8/92.6	75.2/76.4	80.1/79.3
it	it_pud	65.8/65.8	78.2/78.6	89.6/89.7	67.3/66.8	82.5/82.7	87.7/87.2	93.1/93.4	77.6/77.8	87.5/86.9
ja	ja	19.6/21.3	66.2/66.1	67.4/63.3	2.1/1.2	72.7/73.7	51.6/51.9	80.1/86.3	64.9/67.5	35.8/36.1
ja	ja_pud	23.9/27.9	58.5/60.8	70.1/66.9	1.2/1.4	68.4/68.7	50.7/53.0	82.8/86.6	62.6/64.2	41.2/38.4
ko	ko	39.4/43.3	58.9/60.4	71.9/77.7	15.8/19.3	24.4/25.5	40.0/38.1	67.2/69.1	58.1/59.0	66.3/65.1
la	la	28.6/27.6	56.3/56.9	34.6/34.5	41.5/40.6	34.6/36.5	63.8/65.3	63.5/61.2	67.4/63.8	70.6/70.3
la	la_ittb	38.0/38.4	65.2/65.3	69.4/69.4	51.1/50.4	49.0/51.0	60.0/61.3	68.9/68.0	59.3/60.6	67.6/67.6
la	la_proiel	43.1/43.9	63.5/63.8	62.2/62.5	52.2/52.1	66.0/65.1	65.6/66.7	73.7/73.1	68.1/67.3	70.0/69.3
lt	lt	33.3/33.3	77.2/76.7	72.2/71.8	57.0/59.4	78.4/80.8	69.5/69.3	67.1/64.3	58.0/56.8	67.3/63.6
lv	lv	37.1/36.0	73.8/75.0	65.7/67.1	46.9/48.2	47.9/47.0	70.2/71.1	69.4/69.2	58.3/57.1	72.6/72.5
nl	nl	43.9/45.3	60.3/60.7	84.7/86.7	65.9/66.3	65.8/65.0	68.2/68.4	57.4/58.7	59.5/60.8	67.9/67.5
nl	nl_lassysmall	45.9/45.4	73.0/74.5	81.5/81.6	69.0/68.4	73.9/73.3	76.2/77.1	80.6/81.0	69.2/68.7	83.1/84.1
no	no_bokmaal	62.9/63.7	72.4/72.3	89.2/89.8	66.8/67.1	69.5/69.6	84.2/84.1	89.0/88.9	72.6/72.6	81.3/81.4
no	no_nynorsk	56.8/58.3	62.4/62.6	87.3/88.3	66.0/66.8	66.7/67.5	78.1/78.5	83.9/83.7	64.5/65.1	77.0/77.3
pl	pl	58.0/58.2	78.1/78.4	79.8/80.7	68.9/69.6	68.7/69.7	75.3/74.8	83.2/83.1	78.8/79.2	86.5/86.0
pt	pt	55.2/55.7	72.4/73.0	88.6/88.1	67.3/66.8	80.3/80.4	80.9/81.9	88.2/88.2	76.3/75.8	80.3/79.5
pt	pt_br	55.7/56.9	73.0/73.2	84.8/85.1	59.4/59.5	68.1/67.7	78.2/78.4	91.1/90.8	84.7/85.6	75.0/74.2
pt	pt_pud	56.9/56.2	73.1/72.9	88.4/88.8	63.6/63.9	80.6/79.9	83.3/82.8	89.1/89.0	72.2/71.2	83.0/82.6
ro	ro	55.8/55.1	64.9/65.6	80.3/80.7	58.3/58.9	72.2/72.9	76.5/76.8	89.2/89.5	74.6/74.9	81.6/81.1
ru	ru	44.0/42.4	65.2/67.0	85.8/86.5	56.0/56.3	71.0/71.6	69.3/69.1	82.6/83.8	77.5/77.3	76.4/75.6
ru	ru_pud	55.9/56.2	69.9/70.4	90.3/91.1	68.5/68.4	76.3/77.8	84.6/84.9	90.0/91.1	76.2/76.7	84.4/84.1
ru	ru_syntagrus	56.4/56.4	65.1/65.8	83.9/84.7	57.2/57.1	69.4/70.5	74.7/74.6	87.6/88.4	75.0/75.3	76.0/75.7
sk	sk	61.3/60.5	76.6/79.1	86.7/86.5	68.6/68.4	66.1/65.6	74.6/74.7	83.1/83.2	78.0/76.5	87.0/86.8
sl	sl	76.4/77.1	68.4/69.1	89.0/89.1	72.8/73.0	75.7/75.7	78.7/78.8	88.2/87.9	78.2/78.3	82.5/82.3
sl	sl_sst	52.7/57.8	58.4/58.5	82.1/82.6	46.3/45.8	55.1/57.2	64.8/64.8	76.6/75.8	64.1/64.4	67.9/67.6
sv	sv	68.7/68.4	68.9/68.7	90.8/91.1	68.8/69.6	68.0/67.9	85.3/85.1	87.6/88.5	68.3/69.1	82.4/82.6
sv	sv_lines	69.8/69.3	73.8/73.9	89.4/90.1	66.3/66.6	67.4/67.1	86.7/86.6	90.2/90.5	75.1/75.6	84.5/85.0
sv	sv_pud	62.4/61.7	75.0/75.0	90.6/91.1	68.8/69.4	69.4/69.1	85.0/85.5	87.7/88.3	72.5/72.2	85.4/85.8
tr	tr	30.6/16.8	49.6/46.6	64.6/66.2	29.5/31.5	38.1/35.4	55.4/54.5	52.2/59.7	45.0/49.6	49.2/58.7
tr	tr_pud	32.0/22.6	34.2/38.5	65.2/67.5	29.5/35.4	58.4/62.2	39.8/41.6	61.0/64.7	56.0/61.5	42.3/56.1
uk	uk	39.9/40.6	70.6/71.0	72.0/72.4	52.9/52.8	37.0/41.5	67.4/68.2	69.2/68.9	61.2/60.6	70.6/69.9
vi	vi	55.3/56.5	65.1/64.1	39.6/59.3	35.6/38.1	57.3/63.0	50.0/52.5	57.6/62.6	46.6/51.4	55.8/57.5
zh	zh	23.9/19.5	66.4/70.8	63.8/70.7	25.4/29.9	38.7/53.9	32.4/39.9	39.3/41.0	47.6/50.7	26.8/33.5

Table 6.4: Unlabeled attachment f-score of some of the frequent dependency relations for the baseline and the reordering ensemble model. We show the the two numbers separated by slash. We show the green color for improvement and the red color for worse result in the ensemble model; the darkness of the color indicates the level of difference.

lang	data	ADJ	NOUN	VERB
ar	ar	40.4/46.7	70.6/72.5	55.3/58.8
ar	ar_pud	32.3/39.7	73.2/75.9	67.2/70.1
bg	bg	70.6/71.1	85.8/86.2	86.2/86.5
cop	cop	0.0/0.0	63.4/75.7	64.6/76.4
cs	cs	64.8/64.9	77.9/78.5	76.5/76.7
cs	cs_cac	66.0/65.7	79.7/80.5	77.3/77.6
cs	cs_cltt	55.9/56.5	76.9/77.7	68.3/68.9
cs	cs_pud	71.2/70.9	79.4/80	80.2/80.3
da	da	70.9/71.2	79.5/79.3	79.5/79.6
de	de	65.7/66.7	81.3/81.5	75.8/76.3
de	de_pud	61.3/62.4	81.5/81.5	81.0/81.2
el	el	64.3/64.8	79.8/80.5	75.6/75.8
en	en	77.7/78.8	70.6/70.4	81.0/81.3
en	en_lines	74.4/74.7	78.3/78.5	82.2/82.8
en	en_partut	71.9/72.1	76.6/76.7	82.6/82.7
en	en_pud	69.5/70.6	75.4/75.5	81.2/81.6
es	es	75.6/74.6	88.0/88.4	80.6/80.9
es	es_ancora	71.3/71.4	87.4/87.4	83.0/82.9
es	es_pud	66.5/66.3	89.0/89.1	83.2/83.2
et	et	59.5/59.6	59.6/59.5	75.4/75.5
eu	eu	31.1/35.4	37.6/47.9	52.4/61.2
fa	fa	46.2/51.6	68.7/70.7	53.7/59.7
fi	fi	65.8/66.3	61.8/62.5	70.5/70.5
fi	fi_ftb	64.7/65.5	64.7/65.1	69.2/69.5
fi	fi_pud	58.1/59.4	63.8/64.1	74.6/74.8
fr	fr	74.1/74.6	87.3/87.5	81.9/82.7
fr	fr_partut	72.2/72.9	88.4/88.8	83.1/83.8
fr	fr_pud	71.3/71.1	88.7/88.8	81.0/81.1
fr	fr_sequoia	72.0/72.0	86.5/86.6	82.2/82
he	he	64.7/69.1	75.6/77.8	68.1/70.6
hi	hi	22.3/23.5	75.9/74.9	57.5/57.9
hi	hi_pud	48.1/49.3	67.8/67.9	56.6/58.7
hr	hr	72.3/71.8	82.2/82.4	83.1/83.8
hu	hu	42.5/43.3	71.8/72.5	73.6/73.7
id	id	63.2/67.3	70.7/74.5	78.0/79.7
it	it	61.4/63.3	89.1/89.1	85.2/85.4
it	it_pud	71.7/72.0	90.7/90.7	87.1/87.2
ja	ja	52.8/59.5	73.1/74.6	65.1/66.5
ja	ja_pud	60.4/65.4	71.5/72.6	66.7/68.3
ko	ko	55.7/52.9	23.5/24.3	52.4/54.3
la	la	35.1/35.6	43.8/44.4	58.8/58.5
la	la_ittb	57.9/57.4	65.5/66.5	63.5/63.6
la	la_proiel	55.2/55.4	61.8/61.6	64.3/64.1
lt	lt	54.0/57.1	70.8/72.2	69.7/69.7
lv	lv	58.7/60.2	57.0/57.2	70.3/70.6
nl	nl	57.7/61.3	81.9/81.7	66.4/67.2
nl	nl_lassysmall	46.4/47.6	79.8/80	75.4/75.3
no	no_bokmaal	76.0/75.9	83.4/83.4	84.2/84.4
no	no_nynorsk	69.7/70.5	81.4/81.8	79.6/79.9
pl	pl	66.1/67.2	79.2/79.4	85.2/85.2
pt	pt	72.1/73.3	88.7/88.8	82.9/82.7
pt	pt_br	39.5/39.5	88.2/88.2	77.8/77.7
pt	pt_pud	61.4/60.5	89.0/88.8	81.2/81.2
ro	ro	55.6/56.4	79.3/79.5	80.3/80.3
ru	ru	52.3/53.1	77.9/78.5	79.8/80
ru	ru_pud	64.4/64.5	83.1/83.7	81.9/82.4
ru	ru_syntagrus	57.3/56.9	78.7/79.2	74.3/74.6
sk	sk	69.5/69.5	80.5/80.3	84.0/83.5
sl	sl	73.6/72.6	83.3/83.4	85.2/85.2
sl	sl_sst	60.6/61.6	69.4/69.6	67.4/67.1
sv	sv	77.0/76.9	82.8/82.9	81.1/81.4
sv	sv_lines	78.7/78.9	85.1/85.1	83.1/83.3
sv	sv_pud	77.2/77.3	83.4/83.4	83.8/84.1
tr	tr	42.3/46.8	49.4/47.9	48.0/51.5
tr	tr_pud	43.2/46.7	50.0/52	49.5/53.5
uk	uk	49.3/48.8	64.1/64.6	71.9/72.3
vi	vi	31.5/35.7	50.6/56.5	55.1/58.3
zh	zh	47.7/52.1	47.5/56.4	43.0/45.7

Table 6.5: Unlabeled attachment f-score of POS tags as heads for the baseline and the reordering ensemble model. We show the the two numbers separated by slash, the green color for improvement, and the red color for worse result in the ensemble model; the darkness of the color indicates the level of difference.

Part II

Cross-Lingual Transfer of Sentiment Analysis Systems

Chapter 7

Cross-Lingual Sentiment Transfer with Limited Resources

7.1 Introduction

Online discussion forums and social media are often used to express subjective views. They are frequently exploited to express political viewpoints or to post reviews, but they can also be used in serious situations such as disasters where people post about their experiences. In such cases, negative sentiment can be an indication of continued problems while positive sentiment can be an indication that the situation has been happily resolved. Sentiment analysis involves assigning a sentiment label, usually positive, negative, or neutral, to a given text. While there has been quite a bit of work on identifying sentiment in English (e.g. Socher *et al.* [2013], Zhang *et al.* [2016]), and some in languages such as Chinese (e.g. Wan [2008]), Arabic (e.g. Abdul-Mageed and Diab [2011]) and Spanish (e.g. Brooke *et al.* [2009]), there has been much less work in identifying sentiment for low-resource languages where sentiment-labeled data or even machine translation systems do not exist. Yet emergency situations requiring humanitarian assistance can occur in areas where such languages are spoken; for example, the remote mountainous Xinjiang Autonomous Region in China where the Uyghur language is spoken was the site of a major earthquake. When such disasters occur, the ability to quickly develop sys-

tems for recognizing the sentiment posted by locals in their native languages can help humanitarian organizations quickly identify and respond to the areas of greatest need.

Most previous work in cross-lingual sentiment analysis (e.g. Duh *et al.* [2011], Balahur and Turchi [2014], Salameh *et al.* [2015], Zhou *et al.* [2016a], and Zhou *et al.* [2016b]) has typically assumed the availability of a full machine translation system, manually created lexicons, or in-domain parallel corpora. In the event of the occurrence of an incident situation, such resources may not be available for the target language. Furthermore, previous work has focused on a small number of languages and has solely used English as the source language. However, if labeled training data is available in languages in the same family as the target language, it would be useful to utilize such data from multiple source language families to build better sentiment transfer models.

In this chapter, our goal is to create robust sentiment analysis systems for languages or settings where no labeled sentiment training data exists and where machine translation capabilities are minimal. We create cross-lingual sentiment systems for multiple languages, leveraging parallel data from different genres. We consider the standard source-to-target cross-lingual transfer scenario as well as a multilingual scenario in which multiple source languages are used to transfer information to a target language. Our target languages come from a range of Indo-European, Turkic, Afro-Asiatic, Uralic, and Sino-Tibetan language families.

We consider two main approaches for transferring sentiment: *annotation projection* and *direct transfer*. We apply the classical annotation projection in a new setting with a large number of different sources to project supervised labels from the source languages to the target language. Our direct transfer approach directly trains a sentiment analysis

system on the labeled sentences of the source language(s) and applies the trained model directly on the target language. Our goal in this chapter is to show through experimentation the value of these two approaches and the impact of the availability of different resources.

The main contributions of this chapter are the following:

- We introduce a novel direct transfer method using deep learning with a partial lexicalization strategy. The approach relies on translating some input words into the target languages while keeping the structure of the source language. Therefore, our model does not rely on having fully translated text.[1]
- We introduce a simple but effective annotation projection strategy which uses a state of the art sentiment system that is easily and quickly applied to any new language. Our projection model works particularly well for non-Indo-European languages even when in-domain data is not available.
- We systematically experiment with different methods based on direct transfer and projection to develop sentiment analysis systems in the absence of rich resources. Our experiments on a set of 16 different languages show that we can create a robust model. Our experiments study the effect of parallel data from different genres, single and multi-source transfer, and manual dictionaries. Our use of the Bible and Quran for parallel data simulate a low resource scenario for languages where more resources do exist.

Our experiments show that both our approaches benefit from having multiple source

[1]The source code for the direct transfer model is available here: `https://github.com/rasoolims/senti-lstm`

languages and in some cases the performance of the transfer method is close to the supervised model. We show that the direct transfer approach does particularly well with out-of-domain data and when multiple source languages are available, while projection is more suited for in-domain data and for non-European language families. An ensemble of the two approaches results in further performance boosts in the multi-source setting. Our error analysis treating English as a low-resource language concludes that our transfer model makes very reasonable errors on tweets with out-of-vocabulary words.

7.2 Annotation Projection

Annotation projection requires only a simple, fast and easily transferable model. For this we choose to use Naive Bayes Logistic Regression with word embedding features (NBLR + POSwemb) [Yu *et al.*, 2017] to train systems for both *source-side* labeled data and *target-side* projected data. This is an extension of Wang and Manning [2012]. The advantage of this model is that it can be quickly trained whenever a new language is introduced, which makes it potentially more efficient for the low-resource scenario of developing a system within one day as training time is less than that of a deep learning model.

The model uses two types of features: sparse and dense. The sparse features include ngrams up to length 3. For each ngram t in sentence x, the model counts the number of times that the ngram occurs with each possible label l in the training data and stores it as $f^l(t)$. It defines two vectors p_l and q_l for each label l, where each ngram t in sentence x has the following values:

$$\begin{cases} p_l(t) = \alpha + f^l(t) \\ \\ q_l(t) = \alpha + \sum_{y \neq l} f^y(t) \end{cases}$$

where α is a smoothing constant. Finally the following log-count ratio is defined:

$$r_l = \log \left(\frac{p_l / ||p_l||_1}{q_l / ||q_l||_1} \right)$$

where $||p_l||_1$ and $||q_l||_1$ are the L-1 norm values of the corresponding vectors. The value $r_l(t)$ for any ngram t not appearing in sentence x is zero, leading to a very sparse vector. Hence, as opposed to having a sparse count vector (as in traditional machine learning methods), we show the features by the concatenation of its log-count ratios:

$$r(x) = [r_{negative}; r_{neutral}; r_{positive}]$$

where ; shows the concatenation operator. Thus, if the number of seen ngrams in the training data is m, the feature vector r will be a sparse vector in $\mathbb{R}^{3m}$.[2]

For the dense feature representation, we group words according to their part-of-speech (POS) tags in the set P = {NOUN, VERB, ADJECTIVE} and average their pre-trained word embedding vectors. The concatenation of the three averaged word vectors and the sparse vector $r(x)$ is the final input feature to the logistic regression classifier.

[2]In the case of using English as the supervised source language, we also append additional positive and negative indicator features as additional unigrams and calculate their log ratio counts. These indicators are extracted from the sentiment lexicons of Wilson *et al.* [2005] and Hu and Liu [2004].

7.3 Direct Transfer

The main problem with direct transfer is that most of the common features do not generalize beyond each source language: for example, lexical features in one language are unlikely to appear in other languages. We apply the following techniques to address this problem:

- **Word representation features**: We train cross-lingual word representations such that words with similar meanings in different languages have similar representations. We use the method in chapter5 to train cross-lingual word embeddings and word clusters. As a reminder, chapter 5 uses cross-lingual dictionaries to apply random code-switching on monolingual texts (e.g. Wikipedia) in all source and target languages. This leads to a concatenation of monolingual texts in different languages for which each sentence has words from different languages. The code-switched text is trained to derive word clusters and embeddings.
- **Lexicalization with code-switching**: This approach allows us to enhance the direct transfer model by using lexicalized features directly from the target language while at the same time preserving the source language structure. It assumes that either a parallel corpus or bilingual dictionary is available.

  We use a simple deterministic partial translation strategy inspired from chapter 5. First, we use the parallel corpora to create source-target word alignments using GIZA++ [Och and Ney, 2003]. The word alignments are used to create bilingual dictionaries. During training, the bilingual dictionaries are used to translate words from the labeled source training data to the target language. We translate all words

that have an entry in the dictionaries. Therefore, since many words do not exist in the bilingual dictionaries, the resulting training data looks like code-switched text. The following sentence is an example English sentence partially translated to German via the dictionary extracted from Quran and Bible (the underlined words are not translated): **post:*weggingst*** **a:*ein*** **photo** **to:*zu*** **facebook**.

7.3.1 Deep Learning Model

Direct transfer requires a powerful model that can fully benefit from cross-lingual word representations and utilize the context and structure of the input text in order to determine sentiment labels. For this we choose to model direct transfer using long short-term memory (LSTM) networks [Hochreiter and Schmidhuber, 1997]. The input to our model is a sequence of n words in the sentence $x = \{x_1, x_2, \cdots, x_n\}$. We assume the lexicalization step is applied on the words in the training data if a dictionary exists.

Embedding Layer We use the following features for every word in a sentence. For all the following features, if a source training word can be translated into the target language and its translation has an entry in the word embeddings dictionary, we use that as the feature; otherwise we use the source word feature as the input feature:

- A *fixed* pre-trained cross-lingual word embedding $x_{ce} \in \mathbb{R}^{d_{ce}}$ extracted from the method in §5.1.
- A randomly initialized word embedding $x_e \in \mathbb{R}^{d_e}$ for every word in the sentence.
- The cross-lingual word cluster embedding $x_{cc} \in \mathbb{R}^{d_{cc}}$ to represent the word cluster identity of each word.

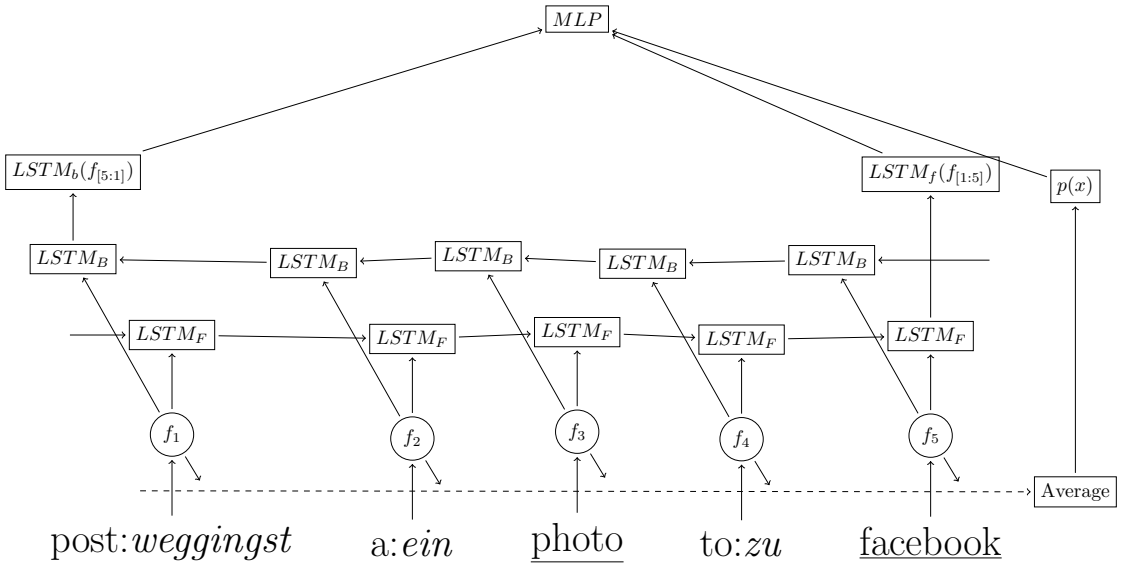


Figure 7.1: A graphical depiction of the neural network model in our direct transfer approach. This is an example of an English tweet translated to German. The words are shown at the bottom with their translation appearing after colon (":") for which underlined words are not translated. The circles (f_i) show the embedding layer, the LSTMs are shown on top of the embedding layer and the average layer on the right side of the words. All the intermediate layers are concatenated and fed to a multi-layer perceptron (MLP). More details are given in §7.3.

- In the case of single source transfer from English, we also use the fixed two-dimensional Sentiwordnet [Baccianella *et al.*, 2010] score $x_{sw} \in \mathbb{R}^2$ that represents the likelihood of a word being positive or negative. We simply translate Sentiwordnet lexicon to the target language by using the translation dictionaries.

Intermediate Layer The input for every word x_i to the intermediate layer is the concatenation of the embedding features: $f \in \mathbb{R}^{d_f} = [x^i_{ce}; x^i_e; x^i_{cc}; x^i_{sw}]$. The intermediate layer is composed of the following structures:

- **Recurrent layer**: We use a Bidirectional LSTM (BiLSTM) for representing the sequence of words in the sentence: a forward pass $LSTM_f(f_{[1:n]}) \in \mathbb{R}^{d_{rec} \times d_f}$ gives

the final representation of the sentence by looking from the beginning to the end, and the backward pass $LSTM_b(f_{[n:1]}) \in \mathbb{R}^{d_{rec} \times d_f}$ looks from the end to the beginning. Then the output of the two LSTMs are concatenated as $r(x) \in \mathbb{R}^{2 \cdot d_{rec}}$.

- **Average layer**: The average layer $p(x) \in \mathbb{R}^{d_f}$ is the average over all the input features f for all words in the sentence. This layer represents the bag-of-words information of the sentence without taking the sequence information into account.

Output Layer The two intermediate layers $r(x)$ and $p(x)$ are concatenated and fed to a hidden layer $H \in \mathbb{R}^{d_h \times (2 \cdot d_{rec} + d_f)}$ activated by rectified linear units (ReLU) [Nair and Hinton, 2010]:

$$H(x) = ReLU(H([r(x); p(x)]))$$

Finally the hidden layer output is fed to the output softmax layer. We use the log-likelihood objective function with the Adam optimizer [Kingma and Ba, 2014] to learn the model parameters.

7.4 Experimental Settings

We have conducted a diverse set of experiments in order to evaluate our methods and determine the impact of the availability of different resources. We experiment with two scenarios:

- *Parallel data*: In this setting, we experiment with both the annotation projection and direct transfer approaches. In order to determine the effect of domain and genre,

we use different types of parallel corpora: religious text, contemporary political text and relatively in-domain and in-genre data from the Linguistic Data Consortium.

- *Manual translation dictionaries*: Instead of automatically creating dictionaries from word alignments, we use the manual translation dictionaries extracted from Wiktionary[3] and use them in the direct transfer model. The Wiktionary entries are noisy and for some languages, the coverage of the lexicon is very low.

7.4.1 Datasets, Tools and Settings

Labeled sentiment data We downloaded tweets labeled with sentiment for 12 languages from Mozetič *et al.* [2016][4] as well as SentiPers data [Hosseini *et al.*, 2018], a set of digital products reviews for Persian, as our source languages. All labeled data is annotated for positive, negative, and neutral. The evaluation languages are Arabic (ar), Bulgarian (bg), German (de), English (en), Spanish (es), Croatian (hr), Hungarian (hu), Polish (pl), Portuguese (pt), Russian (ru), Slovak (sk), Slovene (sl), Swedish (sv), Uyghur (ug) and Chinese (zh). We use 80% of the data as training, 10% for development and 10% of the data for testing. We use all development data sets to train the supervised models. We set the number of training epochs for the transfer model by looking at the performance on the Persian development data. As labeled training data for Uyghur, Chinese and Arabic is either unavailable or much smaller (in the case of Arabic) than that of Mozetič *et al.* [2016]'s Twitter data, we have used these three languages only as target languages. For evaluation data, we employed a native informant for Uyghur and Chinese to annotate a small num-

[3] `https://www.wiktionary.org/`

[4] Not all tweets from Mozetič *et al.* [2016] are available anymore (some of them were deleted).

ber of sentences, and for Arabic, we used the newly released evaluation data of SemEval 2017 Task 4[5]. Table 7.1 shows the data sizes.

Dataset	Train			Test		
	#sen	#tok	#types	#sen	#tok	#types
Arabic	–	–	–	6100	115401	19474
Bulgarian	23739	313003	52922	2958	38685	12545
Chinese	–	–	–	487	10243	3213
Croatian	56212	726037	88966	7025	90544	23827
English	46623	601284	39294	5828	75147	10857
German	63669	822605	78353	7961	102822	19907
Hungarian	36167	427951	83931	4520	53588	17750
Persian	15000	331431	13840	3027	67926	6886
Polish	116105	1455273	136089	14517	182194	37018
Portuguese	62989	712852	41982	7872	89553	12440
Russian	44757	522218	84463	5594	64779	18823
Slovak	40470	576611	82281	5058	71131	20751
Slovene	74238	1106211	108735	9277	137371	29899
Spanish	137106	2007046	91419	17133	249690	27942
Swedish	32600	494139	42446	4074	61423	11698
Uyghur	–	–	–	346	6805	3560

Table 7.1: Training and evaluation sizes for different languages. We ignored the hash tags, URLs and name mentions in the data.

Parallel data We use the following parallel corpora:

- *Bible and Quran*: The motivation for experimenting with religious data is that it is a lot more likely to be available for many languages, even very low-resource languages. We use the Quran and Bible translations as out-of-domain parallel datasets, from a very different genre than our test data, and thus create a low resource scenario for languages that have larger resources available. We use the corpus of Christodouloupoulos and Steedman [2014] for the Bible dataset and the Tanzil

[5] `http://alt.qcri.org/semeval2017/task4/`

translations[6] for the Quran. This dataset has multiple translations for some languages.[7] When using the annotation projection approach, in cases where we have more than one Quran English translation for a target-language sentence, we run the model on all translations and apply majority voting to get the most frequent label.

- *Europarl* : We use the Europarl data [Koehn, 2005] as contemporary political text. We restricted ourselves to those sentences that are translated to all of the 10 languages (Bulgarian, German, English, Spanish, Hungarian, Polish, Portuguese, Slovak, Slovene, and Swedish). That comprised a total of 294738 sentences for all languages.
- *Linguistic Data Consortium (LDC) parallel data*: The LDC packages are produced under the Low Resource Languages for Emergent Incidents (LORELEI) program and consist of a combination of the following genres: news, discussion forums, and social networks such as Twitter. We use seven English to target parallel translations from the LDC data: Chinese (16440 sentences)[8], Persian (57087 sentences)[9], Hungarian (157931 sentences)[10], Arabic (49446 sentences), Russian (193967 sentences)[11],

[6] `http://tanzil.net/trans/`

[7] We excluded a subset of the translations from Russian and English that are interpretations as opposed to translations. For Russian, we use the Krachkovsky, Kuliev, Osmanov, Porokhova, and Sablukov translations and for English, we use the Ahmedali, Arberry, Daryabadi, Itani, Mubarakpuri, Pickthall, Qarai, Qaribullah, Sahih, Sarwar, Shakir, Wahiduddin, and Yusufali translations.

[8] LDC2016E30_LORELEI_Mandarin

[9] LDC2016E93_LORELEI_Farsi

[10] LDC2016E99_LORELEI_Hungarian

[11] LDC2016E95_LORELEI_Russian

Spanish (345940 sentences)[12], and Uyghur (99272 sentences)[13]. Since our evaluation set consists of Twitter data except for Persian, LDC has the most similar genres to our evaluation datasets.

Automatic dictionaries We use GIZA++ [Och and Ney, 2003] to create automatic word alignments between source and target words from the parallel sentence-aligned corpora. After getting the intersected alignments from the two alignment directions, we choose the most frequent translation for each word in order to prune alignment noise.

POS tagging and tokenization OpenNLP is used to split sentences. To tokenize words, we use the Stanford Chinese segmenter [Chang *et al.*, 2008], Madamira Arabic tokenizer [Pasha *et al.*, 2014][14], Hazm Persian tokenizer[15], European tokenizers in the Europarl package, and OpenNLP[16] for all other languages. We use the POS information in the Universal Dependencies[17] to train POS taggers for the NBLR+POSwemb (annotation projection) model.

Embeddings and Brown clusters We use the Wikipedia dump data set to train the monolingual and cross-lingual word embeddings and Brown clusters. We use the Word2vec tool[18] with its default setting and dimension of 300 for all of our embeddings

[12]LDC2016E97_LORELEI_Spanish

[13]LDC2016E57_LORELEI_IL3_Incident_Language_Pack_for_Year_1_Eval

[14]Madamira is used in low-resource mode with the form-based ATB_BWFORM tokenization scheme.

[15]`https://github.com/sobhe/hazm`

[16]`https://opennlp.apache.org/`

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

[18]`https://code.google.com/archive/p/word2vec/`

(monolingual and cross-lingual), except for the comparable embeddings where we used a window size of 10. We trained monolingual and cross-lingual word clusters with the method of Stratos *et al.* [2014] with 500 clusters.

Model settings We set the following parameters for the NBLR +POSwemb model: $\alpha = 1$, $C = 0.01$ or 0.05. We use the LIBLINEAR package [Fan *et al.*, 2008] implemented in the sklearn software[19]. The Dynet library [Neubig *et al.*, 2017] with its default settings is used for the neural network. We use 7 epochs for the single-source transfer and 2 epochs for the multi-source transfer with concatenation. The following parameters are used: $d_{ce} = 300$, $d_e = 400$, $d_{cc} = 50$, $d_{rec} = 400$, $d_h = 400$, and batch size of 10K.

All our models were tuned only when supervised data was assumed available or projected. When we applied our direct transfer models to the new language, we did not do any additional tuning on the target language, except in the case of Persian, which we used as a development language.

7.4.2 Baseline Approaches

We use two simple baselines. The first translates the Sentiwordnet lexicon [Baccianella *et al.*, 2010] to each target language by using the dictionaries extracted from the Bible and Quran parallel data. It decides to give positive/negative sentiment, if the average positive/negative score is at least 0.1 higher than the average negative/positive score. Otherwise it assigns the neutral label. The second baseline assigns all sentences *neutral* majority label of the English training data.

[19] `http://scikit-learn.org/stable/`

7.5 Results and Discussion

This section reports and discusses our results showing the performance of the annotation projection and direct transfer models under the availability of different resources: in-domain parallel corpora (LDC), contemporary European political text (EP), and out-of-domain parallel corpora (BQ), which is available for every language. For direct transfer, we additionally have comparable data (CP) and Wiktionary (WK). Models in all languages apply three-way sentiment classification (positive, negative, and neutral). Since accuracy can be misleading if the majority label from the source language data is dominantly predicted, we use F1 score macro-averaged over the three classes as our evaluation metric. Significance is computed using the bootstrap significance method [Berg-Kirkpatrick *et al.*, 2012].

7.5.1 English to Target Transfer

In the first set of experiments, we conduct single-source transfer by using only English as the source language. The results for projection and direct transfer are depicted in Table 7.2 for difference resources. The single-source transfer approaches perform significantly better *($p < 0.05$)* than the *Neut* majority baseline for all languages and significantly better than the *Senti* baseline in all cases except with Ugyhur, where our evaluation set is quite small, and in some languages with CP and WK, where the transfer is dependent on the quality of the Wikipedia resources.

We see some trends appearing in terms of the impact of different resources. First, and unsurprisingly, using in-domain data (LDC) enables significant increases in performance

Language	Baselines		Projection			Direct					Sup.
	$Neut$	$Senti$	BQ	EP	LDC	CP	WK	BQ	EP	LDC	
Arabic	18.6	25.6	29.8	–	37.2	21.1	31.0	**37.3**	–	30.0	–
Bulgarian	22.4	30.6	29.4	38.7	–	28.6	**45.3**‡	33.0	43.5	–	54.5
Chinese	19.7	35.6	39.8	–	52.7	32.7	**66.8**‡	30.3	–	56.3	–
Croatian	12.8	19.8	24.2	–	–	13.9	–	**30.8**‡	–	–	61.6
German	23.9	32.0	41.0	47.3	–	37.7	**51.0**‡	43.5	45.4	–	59.9
Hungarian	16.5	29.2	29.9	38.1	**47.0**‡	22.4	40.8	41.1	31.8	42.3	60.4
Persian	17.9	25.3	26.5	–	33.1	20.7	31.7	**40.1**‡	–	26.5	67.8
Polish	13.8	26.0	27.9	38.8	–	30.2	32.9	41.7	**43.3**‡	–	64.5
Portuguese	17.3	22.9	36.4	**39.3**	–	22.2	35.4	38.6	**39.3**	–	51.1
Russian	20.0	29.5	36.1	–	48.0	25.3	43.8	44.8	–	**48.1**	69.2
Slovak	11.9	19.2	23.3	30.0	–	24.6	**36.6**‡	22.6	20.4	–	70.1
Slovene	20.6	28.1	31.8	**44.6**‡	–	25.3	32.1	32.2	40.1	–	58.6
Spanish	19.3	25.3	36.0	41.8	**42.7**	27.7	37.7	42.6	39.4	42.2	45.4
Swedish	16.1	22.7	35.0	44.6	–	31.1	40.4	39.1	**49.0**‡	–	62.5
Uyghur	23.6	36.7	37.4	–	**38.6**	25.7	28.0	30.0	–	37.5	–
Average	18.3	27.2	32.3	40.4	42.8	25.9	39.5	36.5	39.1	41.1	60.5

Table 7.2: F1 macro-averaged scores of different methods in the single-source (English to target) transfer setting. "$Neut$" predicts the majority label of the source data (English) and "$Senti$" refers to the baseline Sentiwordnet method. The "sup." column refers to the supervised LSTM model. "CP" refers to the use of embeddings extracted from comparable corpora, "WK" refers to the full direct model using the Wiktionary lexicon. "BQ", "EP" and "LDC" refer to the different parallel datasets used in our experiments: Bible and Quran as "BQ", Europarl as "EP" and the LDC in-domain parallel datasets. ‡ indicates that the best result is significantly better than the second-best (and also other) numbers.

for most languages. For projection, using LDC outperforms both BQ and EP in all cases. For direct transfer, LDC outperforms BQ and EP in all cases except for Persian, Arabic, and Spanish (where both closely approach the supervised bound). Interestingly, BQ actually ends up being a better choice for Persian and Arabic because the BQ corpus is larger or has more unique Quran translations than LDC parallel translations, leading to a dictionary with higher coverage. The quality of direct transfer is affected by the coverage of the dictionary that was extracted, particularly for languages which have morphologically complex words.

Second, when comparing automatic vs. manual (Wiktionary) dictionaries, the result depends heavily on the quality of Wiktionary for that language. Wiktionary significantly outperforms BQ in 4 out of 14 languages, such as Chinese, where the manual dictionary is of exceptionally high quality; in the remaining cases, BQ is either better or the two are indistinguishable. The quality of dictionary is an important factor in direct transfer because both the cross-lingual embeddings and code-switching are the result of those dictionaries. Wiktionary is better for languages such as German and Chinese but less so in Arabic and Persian, where many of the Wiktionary entries occur in their uninflected forms, but we have large BQ corpora for both Arabic and Persian and so better BQ performance.

In comparing projection with direct transfer, direct transfer does significantly better than projection in the low resource BQ setting. This applies to all cases except Chinese and Uyghur and the results are indistinguishable for two other languages, Slovene and Swedish. Thus, for low resource settings without in-domain data, we can conclude that direct transfer is a better than projection; this makes sense because it relies on dictionaries and embeddings rather than projecting sentences which are not relevant for sentiment analysis. On the other hand, with EP, projection does better in five out of nine cases and with LDC, projection does better than direct transfer in five out of seven cases. These results imply that if we do have high-quality in-domain parallel data, projecting annotations from English seems to be preferable.

Our results for English-Chinese are comparable in magnitude to those of previous work, e.g Meng *et al.* [2012]; note we do 3-way classification instead of 2-way.

Language	BQ		EP		LDC	
	-cs	+cs	-cs	+cs	-cs	+cs
Arabic	26.4	37.3	–	–	36.7	30.0
Bulgarian	24.7	33.0	24.6	43.5	–	–
Chinese	16.5	30.3	–	–	55.9	56.3
Croatian	18.5	30.8	–	–	–	–
German	36.0	43.5	40.5	45.4	–	–
Hungarian	30.2	41.1	28.2	31.8	40.6	42.3
Persian	18.4	40.1	–	–	34.9	26.5
Polish	23.6	41.7	24.6	43.3	–	–
Portuguese	20.2	38.6	26.5	39.3	–	–
Russian	24.1	44.8	–	–	43.4	48.1
Slovak	15.1	22.6	17.9	20.4	–	–
Slovene	35.2	32.2	31.8	40.1	–	–
Spanish	27.0	42.6	27.3	39.4	40.9	42.2
Swedish	23.5	39.1	29.4	49.0	–	–
Uyghur	16.1	30.0	–	–	25.4	37.5
Average	23.7	36.5	27.9	39.1	39.7	41.1

Table 7.3: Experimental results on the English to target direct transfer with and without code-switching ("-cs", "+cs") on the labeled data. "BQ" is the Bible and Quran parallel data, "EP" is the Europarl data and "LDC" is the LDC parallel data. The underlined numbers are not significantly better than the all-neutral baseline.

Effect of partial lexicalization Table 7.3 shows the results with and without applying target-language lexicalization. The results with BQ and EP shows that the process of translating lexical items (code-switching) is essential to improving performance. This effect is not as pronounced or as consistent when using LDC, where the quality of bilingual embeddings is already high enough that it suffices for training the direct model without much additional lexicalization.

7.5.2 Multi-Source Transfer

In this set of experiments, we use training datasets from all languages (except that of the target language) as source languages, with English now included as a target low-resource

language. For projection, as described in §7.2, we applied majority voting on sentences aligned to translations in more than one language. For direct transfer, we experimented with the two techniques described in §7.3: (1) Concatenation, where we concatenate all the training sets from different languages and use that as the training data for a single model, and (2) Ensemble, where we train a separate direct model for each source language and then use the most frequent label assignment on the test data. We found that the concatenation approach performs best on average, so we show those results for the direct transfer model. Finally, we apply an ensemble of the projection and direct transfer approaches: 1) projection, 2) concatenation, and 3) single-source ensembles. If the results are tied, we back up to concatenation, projection and the English label respectively. Table 7.4 shows the results for different settings.

Comparing the single-source with the multi-source experiments, we see that having multiple source languages results in a clear improvement for both projection and direct transfer approaches, with significant improvements observed in most languages. The exceptions here are Chinese, where the single-source English-Chinese Wiktionary is especially good, Portuguese, where single-source is the same or slightly better, and Uyghur, where the English projection model with LDC gives better results than using multiple source languages; but this result is not statistically significant. In addition, direct transfer now almost always outperforms projection (10 out of 16 languages for BQ, 9 out of 10 languages for EP). For Arabic, Chinese, and Uyghur, projection is better overall. We note that in these three languages, word order and syntactic structure can be quite different from the source and this may result in lower F-scores for direct transfer. [20]

[20]We note that the best scoring system for Arabic on the SemEval 2017 test set had a macro-average

Language	Projection		Direct		Ensemble		Sup.
	BQ	EP	BQ	EP	BQ	EP	
Arabic	**45.8**‡	–	26.6	–	38.8	–	–
Bulgarian	38.4	42.7	41.6	48.4	43.4	**49.1**	54.5
Chinese	**42.0**	–	31.1	–	38.1	–	–
Croatian	42.5	–	**43.4**	–	**43.4**	–	61.6
English	45.8	42.1	50.6	**54.0**	51.9	53.9	65.3
German	47.5	43.2	51.3	50.3	50.0	**54.7**‡	59.9
Hungarian	33.8	48.5	50.7	**53.8**‡	51.0	51.6	60.4
Persian	37.2	–	**43.7**‡	–	37.2	–	67.8
Polish	41.7	49.3	38.3	**54.6**‡	45.1	52.1	64.5
Portuguese	38.5	34.0	34.3	37.8	**38.9**	37.9	51.1
Russian	41.6	–	46.2	–	**48.3**‡	–	69.2
Slovak	37.9	45.7	48.0	**48.2**‡	37.8	46.7	70.1
Slovene	39.6	44.7	42.3	46.5	45.6	**48.8**‡	58.6
Spanish	33.7	40.9	42.9	43.5	44.0	**45.2**‡	45.4
Swedish	45.0	47.7	44.6	46.5	43.9	**49.8**‡	62.5
Uyghur	**33.4**	–	27.5	–	29.0	–	–
Average	40.3	43.9	41.4	48.4	42.9	49.0	60.8

Table 7.4: F1 macro-average scores with different models (Projection, Direct, Ensemble) using different multi-parallel resources: Bible and Quran (BQ) and Europarl (EP) in the multi-source experiments. ‡ indicates that the best number is significantly better than the second-best number.

Language	BQ	EP
Arabic	39.1 (Slovak)	–
Bulgarian	44.3 (Swedish)	44.8 (Swedish)
Chinese	37.6 (Swedish)	–
Croatian	40.4 (Slovak)	–
English	51.7 (Swedish)	49.0 (German)
German	46.5 (Swedish)	49.6 (Bulgarian)
Hungarian	44.4 (Russian)	48.8 (Polish)
Persian	42.6 (English)	–
Polish	42.2 (Bulgarian)	50.7 (Hungarian)
Portuguese	39.1 (Croatian)	39.5 (Swedish)
Russian	44.8 (English)	–
Slovak	42.8 (Swedish)	48.7 (Polish)
Slovene	45.5 (Croatian)	45.7 (Bulgarian)
Spanish	42.6 (English)	41.0 (German)
Swedish	46.7 (German)	49.0 (English)
Uyghur	31.5 (German)	–

Table 7.5: F1 macro-average scores for best source language in the direct transfer model with Bible and Quran (BQ) and Europarl (EP) parallel corpora.

The ensemble of approaches also enables some gains over the direct transfer model, particularly for the European languages, where the direct model is most effective. In fact, we see that for some of these languages, especially Spanish, the performance of the transfer method approaches that of the supervised model: this in particular is very interesting because the supervised models use gold labeled in-domain data with careful tuning while the transfer model is blindly trained on noisy or crafted datasets from other languages.

Effect of Language Families We notice that European languages (e.g German, English, Swedish, Spanish, Hungarian) tend to benefit a lot from the direct transfer model and in particular the multi-source direct transfer and ensemble models. On the other hand, languages like Arabic, Chinese, Uyghur benefit less from the direct transfer model and have larger gains with projection. The non-Indo-European families are less syntactically and semantically similar to the Indo-European source language families and are more likely to incur changes in structure and word ordering when moving from train to test. We note that Hungarian, which is a Uralic language (also not Indo-European) does not follow this pattern. This may be because it has some similarities with European and notably the Balto-Slavic languages.

To investigate whether certain languages transfer sentiment better from each other, we ran our direct transfer model separately using each source language and determined for each target language the source language with highest performance. The results are shown in Table 7.5 with BQ and EP.

For Arabic, Chinese, and Uyghur, which are the most different from the other lan-

recall and accuracy of 58 when training with supervised Arabic data [Rosenthal *et al.*, 2017].

guages, Slovak, Swedish and German are the best source languages. For the European languages, we notice some interesting trends; for example, the Germanic families (English, Swedish and German) transfer well from each other and additionally are good source languages in general. Slovene transfers well from its Southern Balto-Slavic siblings (Croatian and Bulgarian), and Slovak gets its best performance (48.7 with EP) with Polish (its Western Slavic sibling) as a source language. The Balto-Slavic languages (Slovak, Polish, Croatian, Slovene, Bulgarian, and Russian) generally transfer well to each other and to Hungarian. Surprisingly the Romance families (Portuguese and Spanish) are not the best source languages for each other. There are of course other factors involved, such as the quality and size of the training data in the different languages; more extensive experiments would be required to do a full language family analysis.

7.6 Error Analysis

In order to better understand how the transfer model is doing and what kind of errors it makes, we conducted an error analysis where we considered English to be a target low-resource language. We sampled 100 errors of the best performing English models for direct and projection, where at least one of the two models makes an error.

In our sample, the supervised model agrees with the gold label 63% of the time. Considering the 100 error samples, in 34% of cases, the direct model agrees with the gold label, in 26% of cases, the projection model agrees with the gold label, and in the remaining 40% of cases, both models make an error. Since the direct model performs better than projection when English is a target language, we analyzed the 66 samples from this model and compared them with the predictions of the supervised model in order to understand

whether the errors come from the model itself or from the sentiment transfer.

Generally, we found that the source of sentiment errors comes from the following reasons: a key sentiment indicator was missed (e.g., "love," "excited", "bored"), there were misleading sentiment words (e.g., "super" in context of "getting up super early", "handsome" in context of a question), the tweet contained mispelled/rare words (e.g., "bff,", "bae", "puta"), inference was required (e.g., "i need to seriously come raid your closet" is positive without containing positive words), the correct answer was not clear or not easily determined for a human annotator (e.g., "a mother's job is forever") , or the gold label was clearly wrong (e.g "thanks for joining us tonight! we kept it as spoiler free as possible!" has a neutral instead of positive gold label). There are thus many tweets in this error sample where the sentiment is not clear cut.

The samples analyzing the transfer are divided into four groups with examples provided for each:

1. In the first group (48.5% of cases), the supervised model makes a correct prediction, but the transfer model results in an error. Looking at examples in this group, we found that this often occurs when the English target data contains rare, mispelled, or informal language words which are unlikely to have been learned using either cross-lingual representations or word alignments from parallel corpora.

    - *"**fck na** ! ! marshall ! bear nation hopes your **aight** ! ! !"* (negative, transfer predicts positive)
    - *"eagles might get **doored** tonight :'("* (negative, transfer predicts positive)

2. In the second group (26% of cases), the supervised model and the transfer model

make the same error and thus the cause for the error likely comes from the model rather than the transfer. We determined that 6 of these cases have an incorrect gold label, and 11 result from errors of the supervised model where the answer was unclear, key sentiment was missed, or inference was required.

- *"don 't let anyone discourage you from following your dreams ! it was one of the best decisions i made because it changed my lif ..."* (gold negative [wrong], transfer predicts positive)
- *"can 't wait to be an uncle again a wee boy this time , surely his names got to be jack if no , at least make it his middle name"* (gold positive [requires inference], transfer predicts neutral)

3. In the third group (16.6% of cases), the supervised and transfer model make different kinds of errors and thus the source of the error is likely from both the model itself and the transfer. We determined that three of these cases have an incorrect gold label, and the remaining eight are an error of the supervised model where the answer was unclear, key sentiment was missed, inference was required, or the sentence contained misleading sentiment words.

    - *"mount gambier that was rad and sweaty as hell , just one show left on the tour for us tomorrow in adelaide "* (gold positive [misleading sentiment], supervised neutral, transfer negative)

4. In the fourth group (9% of cases), the gold and supervised models agree, but the transfer model, which was trained on different data, actually makes a better prediction.

- *"this photo taken on 9th september with high quality one of my bday gifts from my friend thank you brother"*(gold and supervised predict neutral, transfer predicts positive)

A large number of errors are clearly because of the transfer, but there are also several cases where even the supervised model makes the same error as the transfer model. The errors the transfer models make are reasonable because of the difficult nature of the Twitter data and the vocabulary it learns from out of domain (BQ or EP) parallel data.

7.7 Conclusion

We have developed and experimented with two transfer methods for sentiment analysis in a diverse set of scenarios. We explored the impact of the availability of different resources including multiple gold labeled datasets in rich-resource languages, parallel data of different genres and domains, and manually and automatically generated dictionaries.

Our experiments show that using multiple source languages yields the best results for most target languages. Naturally, we saw that having similar genre and domain as the training data produces better results than out-of-domain and dissimilar genres; however, with our partial lexicalization strategy, out-of-domain parallel corpora still prove to be an effective low-resource setting.

We have shown that both projection and direct transfer are effective approaches for transferring sentiment in these low-resource scenarios. Direct transfer is the best performer with multiple source languages from related language families, and when high-quality parallel data is not available. Projection is the best performer when parallel in-

domain data is available and with target languages which differ structurally from the source languages. We also found that the ensemble of projection and direct transfer approaches can lead to higher and more robust performance for several of the European languages.

Chapter 8

Conclusion

> *"Therefore the language of mutual understanding is different indeed: to be one in heart is better than to be one in language."*
>
> — Masnvai, Jalal ad-Din Mohammad Rumi (1207-1273).

Transfer methods have several attractive qualities. They address the lack of annotated data in low-resource languages. They are usually fast to develop, and their accuracy, compared to an unsupervised learning method, is much closer to a supervised model.

In this thesis, we have proposed several methods to make use of transfer methods for dependency parsing and sentiment analysis. Our empirical results show a substantial improvement over previously state-of-the-art results, reducing the performance gap to supervised models. We have shown that in the presence of a large amount of translation data, we are able to obtain a highly accurate transfer model that its performance is very close to that of a supervised model. Moreover, we have shown that when a large amount of translation data is not available, we can still leverage annotated data in other languages via using the cross-lingual word representations and syntactic reordering. A summary of the contributions of this thesis is as follows:

- Developing accurate dependency parsers using annotation projection on a large amount of translation data.
- Developing a method for learning cross-lingual word representations, including

word clusters and word embeddings. We show that using those cross-lingual representations works well in dependency parsing and sentiment analysis.

- Developing an accurate method for dependency parsing without the availability of a large amount of translation data. We show that a combination of a direct transfer model and annotation projection along with using cross-lingual word representations and partial lexicalization can substantially improve the accuracy of a transfer method. Our empirical results on a large amount of parallel data have shown further improvements on dependency parsing over our annotation projection method.
- Developing unsupervised syntactic reordering models that help reduce errors in languages from different families in a direct transfer model.
- Conducting a diverse set of experiments on sentiment analysis transfer both for in-domain and out-of-domain data, small translation data and large translation data. We have developed models using annotation projection and direct transfer as well as an ensemble of them.

We believe that these contributions are important to natural language processing. Our methods are straightforward to use for new languages for which annotated data is not available. Furthermore, one can study language families and origins of languages by using the transfer methods as indicators of language similarity measures. One other interesting aspect of our work is the study of syntactic dependency order across languages without having any direct supervision. We believe that there are several potential avenues for improvement over reordering models.

8.1 Future Work

There are several things that are appealing to us as a potential subject for future work.

- **Cross-lingual representation of sentences**: Many of the methods that we proposed in chapters 5-7 are based on the assumption that cross-lingual word embeddings are perfect in representing similar concepts and function words across different languages. This assumption is very ambitious: if it were the case, we would have models with higher accuracies. There has been a great deal of research on developing cross-lingual word embeddings [Ruder, 2017], however, we do not believe that the current methods are good enough for transferring linguistic annotations across languages from different language families. We think that relying on word-level cross-lingual representations is not technically reasonable for transfer methods; instead, developing sentence-level cross-lingual representations can be beneficial. Hence, two semantically similar sentences in two languages should obtain similar vector-based representations. It is worth noting that, when we use a mixture of direct transfer and annotation projection with neural networks, we implicitly develop a cross-lingual sentence representation but it is not clear to what extent this representation can be improved in a pre-training step.
- **Holistic transfer of many natural language processing tasks**: We only explored two tasks in natural language processing, namely dependency parsing and sentiment analysis. One interesting subject for research would be to develop a holistic model that works well on different natural language processing tasks.

- **Applying transfer models on downstream tasks**: Tasks such as dependency parsing are not the ultimate goal for natural language processing: we should be able to apply a dependency parser to downstream applications. One important example is machine translation. Recent work [Katz-Brown *et al.*, 2011b; Lerner and Petrov, 2013] has used supervised parsing models for improving machine translation. The key question is can we use a transfer method to improve machine translation systems. One other example is cross-lingual question answering system, where the knowledge base is a union of different knowledge bases from different languages.
- **Transfer between closely-related languages**: Transferring information between closely related languages such as the Arabic dialects, Turkic families, etc., is an appealing subject to pursue. We think that transferring between related languages might be easier than languages that are not similar but they still have their own problems such as false friends, dialectal variations of the same words, and slight changes in syntactic order.

Each of the above ideas for future work can be interesting on their own right. We believe that transfer methods are not sufficiently explored in the natural language processing community; thus more research is required to make these methods practically more useful.

Bibliography

[Abdul-Mageed and Diab, 2011] Muhammad Abdul-Mageed and Mona T. Diab. Subjectivity and sentiment annotation of modern standard Arabic newswire. In *Proceedings of the 5th Linguistic Annotation Workshop*, pages 110–118. Association for Computational Linguistics, 2011.

[Agarwal *et al.*, 2011] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. Sentiment analysis of twitter data. In *Proceedings of the workshop on languages in social media*, pages 30–38. Association for Computational Linguistics, 2011.

[Agić *et al.*, 2016] 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*, 4:301–312, 2016.

[Ammar *et al.*, 2016a] Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444, 2016.

[Ammar *et al.*, 2016b] Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. One parser, many languages. *arXiv preprint arXiv:1602.01595v1*, 2016.

[Ammar *et al.*, 2016c] Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*, 2016.

[Angeli *et al.*, 2015] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. Leveraging linguistic structure for open domain information extraction. 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)*, pages 344–354, Beijing, China, July 2015. Association for Computational Linguistics.

[Baccianella *et al.*, 2010] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204, 2010.

[Balahur and Turchi, 2014] Alexandra Balahur and Marco Turchi. Comparative experiments using supervised learning and machine translation for multilingual sentiment analysis. *Computer Speech & Language*, 28(1):56–75, 2014.

[Baldwin *et al.*, 2010] Timothy Baldwin, Jonathan Pool, and Susan M. Colowick. Panlex and LEXTRACT: Translating all words of all languages of the world. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 37–40. Association for Computational Linguistics, 2010.

[Bansal *et al.*, 2014] Mohit Bansal, Kevin Gimpel, and Karen Livescu. Tailoring continuous word representations for dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 809–815, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[Berg-Kirkpatrick and Klein, 2010] 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.

[Berg-Kirkpatrick *et al.*, 2012] Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. An empirical investigation of statistical significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005. Association for Computational Linguistics, 2012.

[Bhutani *et al.*, 2016] Nikita Bhutani, H. V. Jagadish, and Dragomir Radev. Nested propositions in open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 55–64, Austin, Texas, November 2016. Association for Computational Linguistics.

[Björkelund *et al.*, 2009] Anders Björkelund, Love Hafdell, and Pierre Nugues. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, pages 43–48, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[Blunsom and Cohn, 2010] Phil Blunsom and Trevor Cohn. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of the 2010 Con-*

ference on Empirical Methods in Natural Language Processing, pages 1204–1213, Cambridge, MA, October 2010. Association for Computational Linguistics.

[Bohnet, 2010] Bernd Bohnet. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd international conference on computational linguistics*, pages 89–97. Association for Computational Linguistics, 2010.

[Bottou, 2010] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[Brody, 2010] Samuel Brody. It depends on the translation: Unsupervised dependency parsing via word alignment. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1214–1222, Cambridge, MA, October 2010. Association for Computational Linguistics.

[Brooke *et al.*, 2009] Julian Brooke, Milan Tofiloski, and Maite Taboada. Cross-linguistic sentiment analysis: From English to Spanish. In *RANLP*, pages 50–54, 2009.

[Brown *et al.*, 1992] Peter F. Brown, Peter V. Desouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

[Brown *et al.*, 1993] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

[Cai *et al.*, 2017] Jiong Cai, Yong Jiang, and Kewei Tu. CRF autoencoder for unsupervised dependency parsing. *arXiv preprint arXiv:1708.01018*, 2017.

[Carreras, 2007] Xavier Carreras. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[Cerisara *et al.*, 2013] Christophe Cerisara, Alejandra Lorenzo, and Pavel Kral. Weakly supervised parsing with rules. In *INTERSPEECH 2013*, pages 2192–2196, Lyon, France, August 2013.

[Chang *et al.*, 2008] Pi-Chuan Chang, Michel Galley, and Christopher D. Manning. Optimizing Chinese word segmentation for machine translation performance. In *Proceedings of the Third Workshop on Statistical Machine Translation*, StatMT '08, pages 224–232, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[Chen and Manning, 2014] 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, Doha, Qatar, October 2014. Association for Computational Linguistics.

[Chen *et al.*, 2016] Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614*, 2016.

[Choi *et al.*, 2015] Jinho D. Choi, Joel Tetreault, and Amanda Stent. It depends: Dependency parser comparison using a web-based evaluation tool. In *Proceedings of the 53rd 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, ACL*, pages 26–31, 2015.

[Christodouloupoulos and Steedman, 2014] Christos Christodouloupoulos and Mark Steedman. A massively parallel corpus: The Bible in 100 languages. *Language Resources and Evaluation*, pages 1–21, 2014.

[Chu, 1965] Yoeng-Jin Chu. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400, 1965.

[Cohen and Smith, 2009] Shay B. Cohen and Noah A. Smith. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 74–82, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[Cohen *et al.*, 2009] Shay B. Cohen, Kevin Gimpel, and Noah A. Smith. Logistic normal priors for unsupervised probabilistic grammar induction. In *Proceedings of NIPS*, 2009.

[Cohen *et al.*, 2011] Shay B. Cohen, Dipanjan Das, and Noah A. Smith. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 50–61, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[Collins and Roark, 2004] Michael Collins and Brian Roark. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[Collins, 2002] Michael Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, July 2002.

[Collins, 2017] Michael Collins. Feedforward neural networks, 2017.

[Cui *et al.*, 2005] Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 400–407, New York, NY, USA, 2005. ACM.

[Daumé III, 2006] Hal Daumé III. *Practical Structured Learning Techniques for Natural Language Processing*. PhD thesis, University of Southern California, Los Angeles, CA, August 2006.

[Dempster *et al.*, 1977] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[Dozat and Manning, 2016] Timothy Dozat and Christopher D. Manning. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*, 2016.

[Dryer and Haspelmath, 2013] Matthew S. Dryer and Martin Haspelmath, editors. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig, 2013.

[Duh *et al.*, 2011] Kevin Duh, Akinori Fujino, and Masaaki Nagata. Is machine translation ripe for cross-lingual sentiment classification? In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 429–433, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[Duong *et al.*, 2015a] Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. Cross-lingual transfer for unsupervised dependency parsing without parallel data. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 113–122, Beijing, China, July 2015. Association for Computational Linguistics.

[Duong *et al.*, 2015b] Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. A neural network model for low-resource universal dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 339–348, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[Durrett *et al.*, 2012] Greg Durrett, Adam Pauls, and Dan Klein. Syntactic transfer using a bilingual lexicon. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1–11, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[Dyer *et al.*, 2010] Chris Dyer, Jonathan Weese, Hendra Setiawan, Adam Lopez, Ferhan Ture, Vladimir Eidelman, Juri Ganitkevitch, Phil Blunsom, and Philip Resnik. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 7–12. Association for Computational Linguistics, 2010.

[Dyer *et al.*, 2011] Chris Dyer, Jonathan Clark, Alon Lavie, and Noah A. Smith. Unsupervised word alignment with arbitrary features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 409–419. Association for Computational Linguistics, 2011.

[Dyer *et al.*, 2013] 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. Association for Computational Linguistics, 2013.

[Edmonds, 1967] Jack Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71:233–240, 1967.

[Eisner, 1996] Jason M. Eisner. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 340–345. Association for Computational Linguistics, 1996.

[Fan *et al.*, 2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9, 2008.

[Freund and Schapire, 1999] Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.

[Ganchev *et al.*, 2009] Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. Dependency grammar induction via bitext projection constraints. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 369–377, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[Gillenwater *et al.*, 2010] Jennifer Gillenwater, Kuzman Ganchev, João Graça, Fernando Pereira, and Ben Taskar. Sparsity in dependency grammar induction. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 194–199, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[Gillenwater *et al.*, 2011] Jennifer Gillenwater, Kuzman Ganchev, João Graça, Fernando Pereira, and Ben Taskar. Posterior sparsity in unsupervised dependency parsing. *The Journal of Machine Learning Research*, 12:455–490, 2011.

[Goldberg and Nivre, 2013] Yoav Goldberg and Joakim Nivre. Training deterministic parsers with non-deterministic oracles. *Transactions of the Association of Computational Linguistics*, 1:403–414, 2013.

[Goldberg, 2017] Yoav Goldberg. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309, 2017.

[Grave and Elhadad, 2015] Edouard Grave and Noémie Elhadad. A convex and feature-rich discriminative approach to dependency grammar induction. 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)*, pages 1375–1384, Beijing, China, July 2015. Association for Computational Linguistics.

[Guo *et al.*, 2015] 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)*, pages 1234–1244, Beijing, China, July 2015. Association for Computational Linguistics.

[Guo *et al.*, 2016] Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. A representation learning framework for multi-source transfer parsing. In *The Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, Phoenix, Arizona, USA, 2016.

[Hacioglu, 2004] Kadri Hacioglu. Semantic role labeling using dependency trees. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[Hakimov *et al.*, 2017] Sherzod Hakimov, Soufian Jebbara, and Philipp Cimiano. Amuse: Multilingual semantic parsing for question answering over linked data. In Claudia d'Amato, Miriam Fernandez, Valentina Tamma, Freddy Lecue, Philippe Cudré-

Mauroux, Juan Sequeda, Christoph Lange, and Jeff Heflin, editors, *The Semantic Web – ISWC 2017*, pages 329–346, Cham, 2017. Springer International Publishing.

[He *et al.*, 2013] He He, Hal Daumé III, and Jason Eisner. Dynamic feature selection for dependency parsing. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1455–1464, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[Headden III *et al.*, 2009] William P. Headden III, Mark Johnson, and David McClosky. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Honnibal and Johnson, 2014] Matthew Honnibal and Mark Johnson. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association of Computational Linguistics*, 2(1):131–142, 2014.

[Hosseini *et al.*, 2018] Pedram Hosseini, Ali Ahmadian Ramaki, Hassan Maleki, Mansoureh Anvari, and Seyed Abolghasem Mirroshandel. Sentipers: a sentiment analysis corpus for Persian. *arXiv preprint arXiv:1801.07737*, 2018.

[Hu and Liu, 2004] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[Huang *et al.*, 2012] Liang Huang, Suphan Fayong, and Yang Guo. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151, Montréal, Canada, June 2012. Association for Computational Linguistics.

[Hudson, 1984] Richard A Hudson. *Word grammar*. Blackwell Oxford, 1984.

[Hwa *et al.*, 2005] Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(03):311–325, 2005.

[Jiang *et al.*, 2016] Yong Jiang, Wenjuan Han, and Kewei Tu. Unsupervised neural dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 763–771, 2016.

[Joshi *et al.*, 2010] Aditya Joshi, A. R. Balamurali, and Pushpak Bhattacharyya. A fallback strategy for sentiment analysis in Hindi: a case study. *Proceedings of the 8th ICON*, 2010.

[Jurafsky and Martin, 2009] Daniel Jurafsky and James H. Martin. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.* Pearson/Prentice Hall, 2009.

[Katz-Brown *et al.*, 2011a] Jason Katz-Brown, Slav Petrov, Ryan McDonald, Franz Och, David Talbot, Hiroshi Ichikawa, Masakazu Seno, and Hideto Kazawa. Training a parser for machine translation reordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 183–192, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[Katz-Brown *et al.*, 2011b] Jason Katz-Brown, Slav Petrov, Ryan McDonald, Franz Och, David Talbot, Hiroshi Ichikawa, Masakazu Seno, and Hideto Kazawa. Training a parser for machine translation reordering. In *Proceedings of the conference on empirical methods in natural language processing*, pages 183–192. Association for Computational Linguistics, 2011.

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics, 2014.

[Kingma and Ba, 2014] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[Klein and Manning, 2004] 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*, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[Koehn *et al.*, 2007] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[Koehn, 2005] Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86, 2005.

[Koo and Collins, 2010] Terry Koo and Michael Collins. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11. Association for Computational Linguistics, 2010.

[Koo *et al.*, 2008] Terry Koo, Xavier Carreras, and Michael Collins. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[Kübler *et al.*, 2009] Sandra Kübler, Ryan McDonald, and Joakim Nivre. *Dependency parsing*. Number 1 in 1. Morgan & Claypool Publishers, 2009.

[Kuhlmann *et al.*, 2011] Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 673–682. Association for Computational Linguistics, 2011.

[Lacroix *et al.*, 2016] Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. Frustratingly easy cross-lingual transfer for transition-based dependency parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1058–1063, San Diego, California, June 2016. Association for Computational Linguistics.

[Le and Zuidema, 2015] Phong Le and Willem Zuidema. Unsupervised dependency parsing: Let's use supervised parsers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 651–661, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[Lerner and Petrov, 2013] 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. Association for Computational Linguistics, 2013.

[Liang, 2005] Percy Liang. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology, 2005.

[Liu, 2012a] Bing Liu. *Sentiment analysis and opinion mining*, volume 5 of *Synthesis lectures on human language technologies*. Morgan & Claypool Publishers, 2012.

[Liu, 2012b] Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

[Ma and Xia, 2014] Xuezhe Ma and Fei Xia. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1337–1348, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[Ma and Zhao, 2012] Xuezhe Ma and Hai Zhao. Fourth-order dependency parsing. In *Proceedings of COLING 2012: Posters*, pages 785–796, Mumbai, India, December 2012. The COLING 2012 Organizing Committee.

[Marcheggiani and Titov, 2017] Diego Marcheggiani and Ivan Titov. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[Marcus *et al.*, 1993] 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.

[Mareček and Straka, 2013] David Mareček and Milan Straka. Stop-probability estimates computed on a large corpus improve unsupervised dependency parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 281–290, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[Martínez Alonso *et al.*, 2017] Héctor Martínez Alonso, Željko Agić, Barbara Plank, and Anders Søgaard. Parsing universal dependencies without training. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 230–240. Association for Computational Linguistics, 2017.

[McClosky *et al.*, 2006] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 152–159, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[McDonald and Pereira, 2006] Ryan McDonald and Fernando Pereira. Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, 2006.

[McDonald and Satta, 2007] Ryan McDonald and Giorgio Satta. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the 10th International Conference on Parsing Technologies*, pages 121–132. Association for Computational Linguistics, 2007.

[McDonald *et al.*, 2005a] Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 91–98, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[McDonald *et al.*, 2005b] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 523–530, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[McDonald *et al.*, 2011] Ryan McDonald, Slav Petrov, and Keith Hall. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[McDonald *et al.*, 2013] Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, et al. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[Mel'čuk and Polguere, 1987] Igor A Mel'čuk and Alain Polguere. A formal lexicon in the meaning-text theory (or how to do lexica with words). *Computational linguistics*, 13(3-4):261–275, 1987.

[Meng *et al.*, 2012] Xinfan Meng, Furu Wei, Xiaohua Liu, Ming Zhou, Ge Xu, and Houfeng Wang. Cross-lingual mixture model for sentiment classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 572–581, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[Mihalcea *et al.*, 2007] Rada Mihalcea, Carmen Banea, and Janyce Wiebe. Learning multilingual subjective language via cross-lingual projections. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 976–983, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[Mikolov *et al.*, 2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[Mikolov *et al.*, 2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[Mozetič *et al.*, 2016] Igor Mozetič, Miha Grčar, and Jasmina Smailović. Twitter sentiment for 15 European languages, 2016. Slovenian language resource repository CLARIN.SI.

[Mukund and Srihari, 2010] Smruthi Mukund and Rohini K Srihari. A vector space model for subjectivity classification in Urdu aided by co-training. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 860–868. Association for Computational Linguistics, 2010.

[Nair and Hinton, 2010] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[Naseem *et al.*, 2010] Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1244, Cambridge, MA, October 2010. Association for Computational Linguistics.

[Naseem *et al.*, 2012] Tahira Naseem, Regina Barzilay, and Amir Globerson. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 629–637. Association for Computational Linguistics, 2012.

[Neubig *et al.*, 2017] Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*, 2017.

[Nivre *et al.*, 2006] Joakim Nivre, Johan Hall, and Jens Nilsson. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, volume 6, pages 2216–2219, 2006.

[Nivre *et al.*, 2016] Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, et al. Universal Dependencies 1.3, 2016. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

[Nivre *et al.*, 2017] Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, et al. Universal Dependencies 2, 2017. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

[Nivre, 2003] Joakim Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*, pages 149–160, 2003.

[Nivre, 2004] Joakim Nivre. Incrementality in deterministic dependency parsing. In Frank Keller, Stephen Clark, Matthew Crocker, and Mark Steedman, editors, *Proceedings of the ACL Workshop Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[Novikoff, 1963] Albert B. J. Novikoff. On convergence proofs for perceptrons. Technical report, STANFORD RESEARCH INST MENLO PARK CALIF, 1963.

[Och and Ney, 2003] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

[Pang and Lee, 2004] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics, 2004.

[Pang *et al.*, 2002] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 79–86, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[Pasha *et al.*, 2014] Arfath Pasha, Mohamed Al-Badrashiny, Mona T. Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan

Roth. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. In *LREC*, volume 14, pages 1094–1101, 2014.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[Qiao *et al.*, 2016] Xiuming Qiao, Hailong Cao, and Tiejun Zhao. Improving unsupervised dependency parsing with knowledge from query logs. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 16(1):3, 2016.

[Quirk *et al.*, 2005] Chris Quirk, Arul Menezes, and Colin Cherry. Dependency treelet translation: Syntactically informed phrasal smt. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 271–279, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[Rasooli and Collins, 2015] Mohammad Sadegh Rasooli and Michael Collins. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 328–338, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[Rasooli and Collins, 2017] Mohammad Sadegh Rasooli and Michael Collins. Cross-lingual syntactic transfer with limited resources. *Transactions of the Association of Computational Linguistics*, 5(1):279–293, 2017.

[Rasooli and Faili, 2012] Mohammad Sadegh Rasooli and Heshaam Faili. Fast unsupervised dependency parsing with arc-standard transitions. In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP*, pages 1–9, Avignon, France, April 2012. Association for Computational Linguistics.

[Rasooli and Tetreault, 2013] Mohammad Sadegh Rasooli and Joel Tetreault. Joint parsing and disfluency detection in linear time. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 124–129, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[Rasooli and Tetreault, 2015] Mohammad Sadegh Rasooli and Joel Tetreault. Yara parser: A fast and accurate dependency parser. *arXiv preprint arXiv:1503.06733*, 2015.

[Rasooli *et al.*, 2017] Mohammad Sadegh Rasooli, Noura Farra, Axinia Radeva, Tao Yu, and Kathleen McKeown. Cross-lingual sentiment transfer with limited resources. *Machine Translation*, Nov 2017.

[Ratnaparkhi, 1996] Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Conference on Empirical Methods in Natural Language Processing*, 1996.

[Rosa and Zabokrtsky, 2015] Rudolf Rosa and Zdenek Zabokrtsky. Klcpos3 - a language similarity measure for delexicalized parser transfer. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 243–249, Beijing, China, July 2015. Association for Computational Linguistics.

[Rosenblatt, 1958] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[Rosenthal *et al.*, 2017] Sara Rosenthal, Noura Farra, and Preslav Nakov. Semeval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[Ruder, 2017] Sebastian Ruder. A survey of cross-lingual embedding models. *CoRR*, abs/1706.04902, 2017.

[Salameh *et al.*, 2015] Mohammad Salameh, Saif M. Mohammad, and Svetlana Kiritchenko. Sentiment after translation: A case-study on Arabic social media posts. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 767–777, 2015.

[Schlichtkrull and Søgaard, 2017] Michael Schlichtkrull and Anders Søgaard. Cross-lingual dependency parsing with late decoding for truly low-resource languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 220–229. Association for Computational Linguistics, 2017.

[Sennrich and Haddow, 2016] Rico Sennrich and Barry Haddow. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91, Berlin, Germany, August 2016. Association for Computational Linguistics.

[Shen *et al.*, 2008] Libin Shen, Jinxi Xu, and Ralph Weischedel. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pages 577–585, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[Simion *et al.*, 2013] Andrei Simion, Michael Collins, and Cliff Stein. A convex alternative to IBM model 2. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1574–1583. Association for Computational Linguistics, 2013.

[Smith and Eisner, 2005] Noah A. Smith and Jason Eisner. Guiding unsupervised grammar induction using contrastive estimation. In *Proceedings of IJCAI Workshop on Grammatical Inference Applications*, pages 73–82, 2005.

[Socher *et al.*, 2011] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*, pages 151–161. Association for Computational Linguistics, 2011.

[Socher *et al.*, 2013] Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, Christopher Potts, et al. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

[Spitkovsky *et al.*, 2011a] Valentin I. Spitkovsky, Hiyan Alshawi, Angel X. Chang, and Daniel Jurafsky. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1281–1290. Association for Computational Linguistics, 2011.

[Spitkovsky *et al.*, 2011b] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. Lateen EM: Unsupervised training with multiple objectives, applied to dependency grammar induction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1269–1280, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[Spitkovsky *et al.*, 2012] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. Bootstrapping dependency grammar inducers from incomplete sentence fragments via austere models. In *International Conference on Grammatical Inference*, pages 189–194, 2012.

[Spitkovsky *et al.*, 2013] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1983–1995, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[Spreyer and Kuhn, 2009] Kathrin Spreyer and Jonas Kuhn. Data-driven dependency parsing of new languages using incomplete and noisy training data. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 12–20, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[Straka and Straková, 2017] Milan Straka and Jana Straková. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[Stratos *et al.*, 2014] Karl Stratos, Do-kyum Kim, Michael Collins, and Daniel Hsu. A spectral algorithm for learning class-based n-gram models of natural language. *Proceedings of the Association for Uncertainty in Artificial Intelligence*, 2014.

[Stratos *et al.*, 2015] Karl Stratos, Michael Collins, and Daniel Hsu. Model-based word embeddings from decompositions of count matrices. 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)*, pages 1282–1291, Beijing, China, July 2015. Association for Computational Linguistics.

[Täckström *et al.*, 2012] 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 association for computational linguistics: Human language technologies*, pages 477–487. Association for Computational Linguistics, 2012.

[Täckström *et al.*, 2013] Oscar Täckström, Ryan McDonald, and Joakim Nivre. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1061–1071, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[Tarjan, 1977] Robert Endre Tarjan. Finding optimum branchings. *Networks*, 7(1):25–35, 1977.

[Tesnière, 1953] Lucien Tesnière. *Esquisse d'une syntaxe structurale.* Klincksieck, 1953.

[Tetreault *et al.*, 2010] Joel Tetreault, Jennifer Foster, and Martin Chodorow. Using parse features for preposition selection and error detection. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 353–358, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[Tiedemann and Agić, 2016] Jörg Tiedemann and Željko Agić. Synthetic treebanking for cross-lingual dependency parsing. *Journal of Artificial Intelligence Research*, 55:209–248, 2016.

[Tiedemann *et al.*, 2014] Jörg Tiedemann, Željko Agić, and Joakim Nivre. Treebank translation for cross-lingual parser induction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 130–140, Ann Arbor, Michigan, June 2014. Association for Computational Linguistics.

[Tiedemann, 2015] Jörg Tiedemann. Improving the cross-lingual projection of syntactic dependencies. In *Nordic Conference of Computational Linguistics NODALIDA 2015*, pages 191–199, 2015.

[Turian *et al.*, 2010] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[Vogel *et al.*, 1996] Stephan Vogel, Hermann Ney, and Christoph Tillmann. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 836–841. Association for Computational Linguistics, 1996.

[Wan, 2008] Xiaojun Wan. Using bilingual knowledge and ensemble techniques for unsupervised Chinese sentiment analysis. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 553–561, Honolulu, Hawaii, October 2008. Association for Computational Linguistics.

[Wang and Eisner, 2016] Dingquan Wang and Jason Eisner. The galactic dependencies treebanks: Getting more data by synthesizing new languages. *Transactions of the Association for Computational Linguistics*, 4:491–505, 2016.

[Wang and Manning, 2012] Sida Wang and Christopher D Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics, 2012.

[Weiss *et al.*, 2015] David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 323–333, Beijing, China, July 2015. Association for Computational Linguistics.

[Wilson *et al.*, 2005] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics, 2005.

[Wu and Weld, 2010] Fei Wu and Daniel S. Weld. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 118–127, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[Xiao and Guo, 2015] Min Xiao and Yuhong Guo. Annotation projection-based representation learning for cross-lingual dependency parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 73–82, Beijing, China, July 2015. Association for Computational Linguistics.

[Yoshikawa *et al.*, 2016] Masashi Yoshikawa, Hiroyuki Shindo, and Yuji Matsumoto. Joint transition-based dependency parsing and disfluency detection for automatic speech recognition texts. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1036–1041, Austin, Texas, November 2016. Association for Computational Linguistics.

[Yu *et al.*, 2017] Tao Yu, Christopher Hidey, Owen Rambow, and Kathleen McKeown. Leveraging sparse and dense feature combinations for sentiment classification. *arXiv preprint arXiv:1708.03940*, 2017.

[Zeman and Resnik, 2008] Daniel Zeman and Philip Resnik. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*, pages 35–42, 2008.

[Zhang and Barzilay, 2015] 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.

[Zhang and Clark, 2008] 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.

[Zhang and Nivre, 2011] Yue Zhang and Joakim Nivre. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the*

Association for Computational Linguistics: Human Language Technologies, pages 188–193, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[Zhang *et al.*, 2016] Rui Zhang, Honglak Lee, and Dragomir Radev. Dependency sensitive convolutional neural networks for modeling sentences and documents. *arXiv preprint arXiv:1611.02361*, 2016.

[Zhou *et al.*, 2014] Guangyou Zhou, Tingting He, and Jun Zhao. Bridging the language gap: Learning distributed semantics for cross-lingual sentiment classification. In Chengqing Zong, Jian-Yun Nie, Dongyan Zhao, and Yansong Feng, editors, *Natural Language Processing and Chinese Computing*, pages 138–149, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[Zhou *et al.*, 2015] HuiWei Zhou, Long Chen, Fulin Shi, and Degen Huang. Learning bilingual sentiment word embeddings for cross-language sentiment classification. 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)*, pages 430–440, Beijing, China, July 2015. Association for Computational Linguistics.

[Zhou *et al.*, 2016a] Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. Attention-based lstm network for cross-lingual sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 247–256, Austin, Texas, November 2016. Association for Computational Linguistics.

[Zhou *et al.*, 2016b] Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. Cross-lingual sentiment classification with bilingual document representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1412, Berlin, Germany, August 2016. Association for Computational Linguistics.

[Zipf, 1935] George K Zipf. The psychology of language. *NY Houghton-Mifflin*, 1935.