Parsing with Sparse Annotated Resources

by

Yuan Zhang

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013

Author .....
Department of Electrical Engineering and Computer Science
May 22, 2013

Certified by .....
Regina Barzilay
Professor
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Students

Parsing with Sparse Annotated Resources

by

Yuan Zhang

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 2013, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

This thesis focuses on algorithms for parsing within the context of sparse annotated resources. Despite recent progress in parsing techniques, existing methods require significant resources for training. Therefore, current technology is limited when it comes to parsing sentences in new languages or new grammars.

We propose methods for parsing when annotated resources are limited. In the first scenario, we explore an automatic method for mapping language-specific part-of-speech (POS) tags into a universal tagset. Universal tagsets play a crucial role in cross-lingual syntactic transfer of multilingual dependency parsers. Our central assumption is that a high-quality mapping yields POS annotations with coherent linguistic properties which are consistent across source and target languages. We encode this intuition in an objective function. Given the exponential size of the mapping space, we propose a novel method for optimizing the objective over mappings. Our results demonstrate that automatically induced mappings rival their manually designed counterparts when evaluated in the context of multilingual parsing.

In the second scenario, we consider the problem of cross-formalism transfer in parsing. We are interested in parsing constituency-based grammars such as HPSG and CCG using a small amount of data annotated in the target formalisms and a large quantity of coarse CFG annotations from the Penn Treebank. While the trees annotated in all of the target formalisms share a similar basic syntactic structure with the Penn Treebank CFG, they also encode additional constraints and semantic features. To handle this apparent difference, we design a probabilistic model that jointly generates CFG and target formalism parses. The model includes features of both parses, enabling transfer between the formalisms, and preserves parsing efficiency. Experimental results show that across a range of formalisms, our model benefits from the coarse annotations.

Thesis Supervisor: Regina Barzilay
Title: Professor

Acknowledgments

First and foremost, I'm grateful to my advisor Regina Barzilay whose guidance and assistance was crucial in the completion of this thesis. The work present here will never be done without her energy and involvement on the research and her never ending source of support.

Many thanks to those who contributed to the papers on which this thesis is based. I would like first express my gratitude to Amir Globerson of the Hebrew University. He always enlightened me and gave a lot of invaluable suggestions when we discussed and worked together. I would also like to thank to Tommi Jaakkola, Tahira Naseem, Tao Lei, S. R. K. Branavan, Yoong Keok Lee, Christina Sauper, Nate Kushman, Zach Hynes, every member of the MIT NLP group whose ideas, feedback, and assistance was critical to this work.

I'm also thankful to Yusuke Miyao of National Institute of Informatics in Japan, Ozlem Cetinoglu of University of Stuttgart, Stephen Clark of University of Cambridge, Michael Auli of Microsoft Research and Yue Zhang of Singapore University of Technology and Design for answering questions and sharing the codes of their work. I would also like to acknowledge the support of the National Science Foundation (IS-0835445), the MURI program (W911NF-10-1-0533), the DARPA BOLT program and the Army Research Office (grant 1130128-258552).

Last but not least, I would like to thank my father and mother. I've learnt many things about life from them. They gave me infinite courages and support to overcome any obstacles and difficulties in my research work.

Bibliographic Notes

Portions of this thesis are based on the following papers:

"Transfer Learning for Constituency-Based Grammars" by Yuan Zhang, Regina Barzilay and Amir Globerson. This paper will appear in the *Proceedings of the 51st Annual Meeting of the Association of Computational Linguistics (ACL)*.

"Learning to Map into a Universal POS Tagset" by Yuan Zhang, Roi Reichart, Regina Barzilay and Amir Globerson. This paper appeared in the *Proceedings of the 2012 Conference on Empirical Methods on Natural Language Processing and Computational Natural Language Learning (EMNLP)*.

Contents

1 Introduction **10**

1.1 Transfer Learning across Languages 11

1.2 Transfer Learning across Grammars 13

1.3 Thesis Overview 15

2 Related Work **16**

2.1 Multilingual Parsing 16

2.2 Syntactic Category Refinement 17

2.3 Parsing for Constituency-based Grammar Formalism 17

2.3.1 Combinatory Categorial Grammar 18

2.3.2 Lexical Functional Grammar 19

2.3.3 Head-Driven Phrase Structure Grammar 20

2.4 Cross-formalism Transfer Learning 20

2.4.1 CCGbank 21

2.4.2 LFG Automatic Annotator 22

2.4.3 HPSGbank 22

2.5 Summary 23

3 Mapping into Universal Tagsets **24**

3.1 Introduction 24

3.2 Task Formulation 25

3.3 Model 26

3.3.1 Mapping Characterization 27

3.3.2 The Overall Objective 29
3.4 Parameter Estimation 30
3.4.1 Optimizing the Mapping A 30
3.4.2 Learning the Objective Weights 32
3.4.3 Source Language Selection 33
3.5 Evaluation Setup 34
3.6 Experiment and Analysis 35
3.7 Summary 39

4 Transfer Learning for Constituency-based Grammars 41
4.1 Introduction 41
4.2 Task Formulation 43
4.3 A Joint Model for Two Formalisms 43
4.4 Implementation 46
4.4.1 Supertagging 46
4.4.2 Feature Forest Pruning 46
4.4.3 Binarization 47
4.4.4 Implementation in Other Formalisms 48
4.5 Features 48
4.6 Evaluation Setup 51
4.7 Experiment and Analysis 53
4.8 Summary 56

5 Conclusions and Future Work 57

A Derivation of the Mapping Optimization Algorithm 59

B Examples of POS Tag Mappings 62

List of Figures

1-1 Illustration of similarities in POS tag statistics across languages. (a) The unigram frequency statistics on five tags for two close languages, English and German. (b) Sample sentences in English and German. Verbs are shown in blue, prepositions in red and noun phrases in green. It can be seen that noun phrases follow prepositions. 13

1-2 Derivation trees for CFG as well as CCG, HPSG and LFG formalisms. 15

2-1 Example CCG partial derivation as a binary tree for the sentence *Following the ...lead has been a generally bovine press.* 22

2-2 An example of partially specified derivation tree. 23

3-1 An iterative algorithm for minimizing our objective in Eq. Equation 3.7. For simplicity we assume that all the weights α_i and λ are equal to one. It can be shown that the objective monotonically decreases in every iteration. 31

3-2 Objective values for the different mappings used in our experiments for four languages. Note that the goal of the optimization procedure is to minimize the objective value. 38

4-1 Illustration of the joint CCG-CFG representation 42

4-2 Example of transfer between CFG and CCG formalisms. 51

4-3 Model performance with 500 target formalism trees and different numbers of CFG trees, evaluated using labeled/unlabeled dependency F-score and unlabeled PARSEVAL. 53

4-4 Model performance with different target formalism trees and zero or 15,000 CFG trees. The first row shows the results of labeled dependency F-score and the second row shows the results of unlabeled PARSEVAL. 54

List of Tables

3.1 The set of typological features that we use for source language selection. The first column gives the ID of the feature as listed in WALS. The second column describes the feature and the last column enumerates the allowable values for each feature; besides these values each feature can also have a value of 'No dominant order'. 33

3.2 Directed dependency accuracy of our model and the baselines using gold POS tags for the target language 36

4.1 Templates of atomic features. 49

4.2 Binary feature templates used in $f(y, S)$. Unary and root features follow a similar pattern. 50

4.3 Training/Dev./Test split on WSJ sections and PARC700 for different grammar formalisms. 52

4.4 The labeled/unlabeled dependency F-score comparisons between our model and state-of-the-art parsers. 55

4.5 Example features with high weight. 56

Chapter 1

Introduction

The lack of annotated resources is a significant obstacle to achieving high parsing performance. In some cases, syntactic annotations are readily available. For instance, the CoNLL dataset [2] consists of syntactic treebanks for 19 languages. However, these treebanks only cover a small fraction of the hundreds of existing world languages. Many of these languages have no available treebanks, and thus are beyond the scope of state-of-the-art supervised parsers. Moreover, existing constituency treebanks are primarily annotated in context-free grammar (CFG). For many newly developed grammars, there are no available treebanks. As such, researchers have not yet developed high-quality parsing systems for these grammars.

In this thesis, we propose methods to improve parsing performance in the context of resource-poor languages or grammars by applying transfer learning techniques. The basic idea of transfer learning is to extract pivotal information from annotations in resource-rich languages or grammars and use it to improve parsing performance in resource-poor contexts. In particular, our work focuses on two specific scenarios: transfer learning across different languages and transfer learning across different grammars.

We first explore an automatic method for mapping language-specific part-of-speech (POS) tags to a set of universal tags. Universal POS tagsets play a crucial role in cross-lingual syntactic transfer of multilingual dependency parsers. Many transfer approaches assume that this universal POS representation is available and learn to

transfer from non-parallel data [37, 60]. The mappings used in previous work were constructed manually. The goal of our work is to automatically construct mappings that are optimized for performance on downstream tasks, like parsing. We evaluate our algorithm on 19 languages and demonstrate that the quality of our mapping rivals the quality of their hand-crafted counterparts.

The second part of our work focuses on the cross-formalism transfer scenario. Over the last several decades, linguists have introduced many different grammars for describing the syntax of natural languages. Moreover, the ongoing process of developing new grammar formalisms is intrinsic to linguistic research. However, in order to parse the sentence within the context of these grammars, annotated sentences for training are required. The standard solution to obtain these annotations relies on manually crafted transformation rules that map readily available syntactic annotations (e.g, the Penn Treebank) to the desired formalism. Designing these transformation rules is a major undertaking which requires multiple correction cycles and a deep understanding of the underlying grammar formalisms. In this thesis, we propose an alternative approach for parsing constituency-based grammars. Instead of using manually-crafted transformation rules, this approach relies on a small amount of annotations in the target formalism. To compensate for the annotation sparsity, our approach utilizes coarsely annotated data that is available in large quantities. Our experimental results show that across all the target formalisms, our model significantly benefits from the coarsely annotated data.

1.1 Transfer Learning across Languages

The goal of our first task is to automatically learn a mapping from language-specific part-of-speech tags to a universal tagset. In multilingual parsing, this universal POS representation is required for cross-lingual syntactic transfer. Specifically, the universal tag annotations enable an unlexicalized parser to make use of on annotations from one language when learning a model for another language.

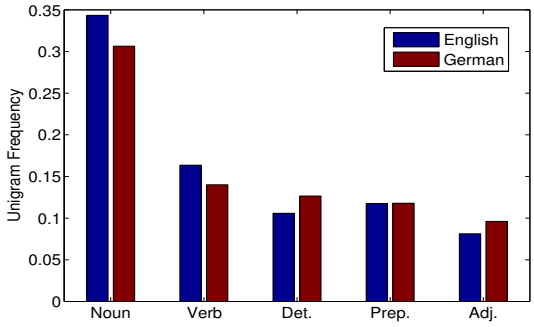
While the notion of a universal POS tagset is widely accepted, it is hardly ever

used in practice for annotation of monolingual resources. In fact, available POS annotations are designed to capture language-specific idiosyncrasies and therefore are substantially more detailed than a coarse universal tagset. To reconcile these cross-lingual annotation differences, a number of mapping schemes have been proposed in the parsing community [60, 51, 44]. In all of these cases, the conversion is performed manually and has to be repeated for each language and each annotation scheme.

Despite the apparent simplicity, deriving a mapping is by no means easy, even for humans. In fact, the universal tagsets manually induced by [51] and by [44] disagree on 10% of the tags. One example of these discrepancies is the mapping of the Japanese tag "PVfin" to a universal tag. In one scheme, "PVfin" maps to "particle"; in another scheme it maps to "verb". Moreover, the quality of this conversion has direct implications on parsing performance. In the Japanese example above, this difference in mapping yields a 6.7% difference in parsing accuracy.

The goal of our algorithm is to induce the mapping for a new language, utilizing existing manually-constructed mappings as training data. The existing mappings developed in the parsing community rely on gold POS tags for the target language. Another possible scenario is to apply the mapping technique to resource-poor languages where gold POS annotations are lacking. In such cases, a mapping algorithm has to operate over automatically induced word clusters on the target language (e.g., using the Brown algorithm) and map the clusters to universal tags. We propose a mapping approach that can effectively handle both gold tags and induced clusters.

Our central hypothesis is that a high-quality mapping yields POS annotations with coherent linguistic properties which are consistent across languages. Since words with the same universal tags play the same linguistic role in source and target languages, we expect similarity in their *global distributional statistics*. Figure 1-1a shows statistics for two close languages, English and German. We can see that their POS unigram frequencies on the five most common tags are very close. Other properties concern *POS tag per sentence statistics* – e.g., every sentence has to have at least one verb. Finally, the mappings can be further constrained by the *typological properties* of the target language that specify likely tag sequences. This information is readily available



(a)

-Investors [are appealing] to the Securities and Exchange Commission not to [limit] their access to information [*about* stock purchases] and sales [*by* corporate insiders]

-Einer der sich [*für* den Milliardär] [ausspricht] [ist] Steve Jobs dem Perot [*für* den aufbau] der Computerfirma Next 20 Millionen Dollar [bereitstellte]

(b)

Figure 1-1: Illustration of similarities in POS tag statistics across languages. (a) The unigram frequency statistics on five tags for two close languages, English and German. (b) Sample sentences in English and German. Verbs are shown in blue, prepositions in red and noun phrases in green. It can be seen that noun phrases follow prepositions.

even for resource-poor languages [23]. For instance, since English and German are prepositional languages, we expect to observe adposition-noun sequences but not the reverse (see Figure 1-1b for sample sentences). We encode these heterogeneous properties into an objective function that guides the search for an optimal mapping.

Our approach encodes these three types of properties into a single objective and optimizes the objective via a variant of the Expectation-Maximization algorithm (EM) [45] and the Convex-Concave procedure [59]. We evaluate the quality of our mapping based on the algorithm's performance on the downstream task, i.e. multilingual transfer parsing. Experimental results show that the quality of our mapping is competitive with that of the manual mapping.

1.2 Transfer Learning across Grammars

The goal of the second task is to improve parsing performance within the context of new grammars when the annotated treebanks in those grammars are lacking. Traditional approaches for this problem have focused on the conversion from the Penn Treebank to the treebank annotated in the target grammar. These approaches relied on manually specified conversion rules; the process of designing rules for a new grammar requires a deep understanding of that grammar and significant human ef-

fort. In addition, designing these rules frequently requires external resources such as Wordnet and even involves correction of the existing treebank. This effort has to be repeated for each new grammar formalism, each new annotation scheme, and each new language.

In this paper, we propose an alternative approach for parsing constituency-based grammars. Instead of using manually-crafted transformation rules, this approach relies on a small amount of annotations in the target grammar. Frequently, such annotations are available in linguistic texts that introduce the grammar. For instance, a textbook on HPSG [54] illustrates grammatical constructions using about 600 examples. While these examples are informative, they are not sufficient for training in and of themselves. Our approach utilizes coarsely annotated data available in large quantities to address the annotation sparsity. A natural candidate treebank for such coarse annotations is the Penn Treebank, which is annotated in context-free grammar (CFG). Given that the Penn Treebank is selected as the source treebank, the target formalism can be any constituency-based grammar, such as Combinatory Categorial Grammar (CCG) [56], Lexical Functional Grammar (LFG) [1] or Head-Driven Phrase Structure Grammar (HPSG) [54]. The treebanks annotated in all of these formalisms share a similar basic syntactic structure with Penn Treebank CFG. However, the target formalisms also encode additional constraints and semantic features. For instance, the Penn Treebank annotations do not make an explicit distinction between a complement and an adjunct; CCG, LFG and HPSG all mark these roles explicitly. Moreover, even identical syntactic information is encoded differently in these formalisms. An example of this phenomenon is the marking of subject. In LFG, this information is captured in the mapping equation, namely $\uparrow$ SBJ $=\downarrow$, while the Penn Treebank represents this information as a functional tag, such as NP-SBJ. Figure 1-2 shows derivations in the three target formalisms we consider, as well as a CFG derivation. We can see that the derivations in each of these formalisms share the same basic structure while the formalism-specific information is mainly encoded in the lexical entries and node labels.

Our model utilizes syntactic structure sharing across formalisms and exploits the

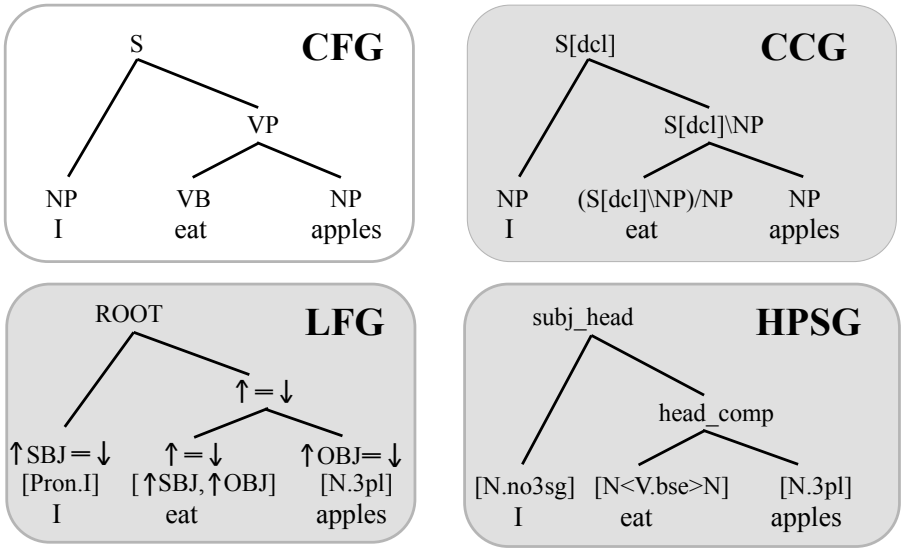


Figure 1-2: Derivation trees for CFG as well as CCG, HPSG and LFG formalisms.

syntactic information in the source treebank to improve parsing for the target treebank. We evaluate our approach on three constituency-based grammars — CCG, HPSG, and LFG. Our experimental results demonstrate that for all three formalisms, parsing accuracy can be improved by training with additional coarse annotations.

1.3 Thesis Overview

The remainder of the thesis is organized as follows. In the next chapter we discuss related work in the areas of multilingual parsing, syntactic category refinement, parsing for different grammars, and cross-formalism transfer learning. In chapter 3, we provide formal descriptions of our algorithm for mapping the language-specific tags to the universal tagset, and we give empirical results showing the efficacy of this algorithm. In chapter 4, we describe our method of transferring syntactic information across different formalisms, and provide empirical results. Chapter 5 concludes with the main ideas and contributions of this work, along with potential directions for future research.

Chapter 2

Related Work

Our work focuses on two specific tasks in transfer learning. Our first model learns to map the language-specific tags into a universal tagset. Our second model enables transfer between different grammar formalisms. Specifically, our coarsely annotated treebank contains rich syntactic information, and our model utilizes this information to improve parsing performance on refined target grammars. While on the high level both tasks belong to the thread of transfer learning, they require fundamentally different approaches, and are related to different categories of previous work. In the following sections we describe the prior work relevant to these two tasks.

2.1 Multilingual Parsing

Early approaches for multilingual parsing used parallel data to bridge the gap between languages when modeling syntactic transfer. In this setup, finding the mapping between various POS annotation schemes was not essential; instead, the transfer algorithm could induce it directly from the parallel data [26, 58, 3]. However, more recent transfer approaches focus on learning to transfer from non-parallel data and such mappings between POS become essential [60, 38, 12, 44, 43]. These approaches assume access to a common input representation in the form of universal tags, which enables the model to connect patterns observed in the source language to their counterparts in the target language.

Despite ongoing efforts to standardize POS tags across languages (e.g., EAGLES initiative [15]), many corpora are still annotated with language-specific tags. In previous work, their mappings to universal tags were performed manually. Yet, even though some of these mappings have been developed for the same CoNLL dataset [2, 46], they are not identical and yield different parsing performance [60, 51, 44]. The goal of our work is to automate this process and construct mappings that are optimized for performance on downstream tasks (here we focus on parsing). As our results show, we achieve this goal on a broad range of languages and evaluation scenarios.

2.2 Syntactic Category Refinement

Our work of learning to map language-specific POS tags also relates to work in syntactic category refinement in which POS categories and parse tree non-terminals are refined in order to improve parsing performance [16, 30, 35, 50, 52, 33]. Our work differs from these approaches in two ways. First, these methods have been developed in the monolingual setting, while our mapping algorithm is designed for multilingual parsing. Second, these approaches are trained on the syntactic trees of the target language, which enables them to directly link the quality of newly induced categories with the quality of syntactic parsing. In contrast, we are not given trees in the target language. Instead, our model is informed by mappings derived for other languages.

2.3 Parsing for Constituency-based Grammar Formalism

The goal of our second task is to utilize a treebank annotated in some coarse formalism to improve parsing performance in some target formalism. In particular, we choose the Penn Treebank, which is annotated in Context Free Grammar (CFG), as the source treebank and the following three constituency-based grammars as our target formalisms: Combinatory Categorial Grammar (CCG), Lexical Functional Grammar (LFG) and Head-Driven Phrase Structure Grammar (HPSG). In this section,

we briefly introduce each target grammar and the existing parsing models for these grammars.

2.3.1 Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG) [56] is a constituency-based lexicalized grammar based on Categorial Grammar. It is also one of the unification-based grammars. Each word in a CCG analysis is associated with a lexical entry that encodes the syntactic and semantic constraints. The formalism generates constituency-based structures and is therefore a type of constituency-based structure grammar rather than a dependency grammar.

Clark et al. proposed log-linear models for wide-coverage CCG parsing [10, 11]. They presented two types of models: a dependency model and a normal-form model. The two models generated features of a CCG tree from different aspects. The dependency model has features defined over the CCG predicate-argument dependencies, whereas the dependencies for the normal-form model are defined in terms of local rule instantiations in the derivation. The models were both evaluated in terms of the predicate-argument dependency F-score. Though the performance of dependency model was slightly better, its time cost is much higher than the time cost of the normal-form model because of the weaker locality of the features. In our work, we follow the idea of the normal-form model. The trees in our source treebank (Penn Treebank) are represented in derivation form. Therefore the normal-form model fits our scenario better.

One of the key components in the log-linear model for CCG parsing is the packed chart representation of derivations, also known as feature forests [41]. The packed charts compactly represent the exponential number of possible derivations or dependency structures by grouping together equivalent chart entries. Consequently, the marginal probability for each cell can be computed efficiently by using a variant of the inside-outside algorithm.

Another key component of the parsing system is a Maximum Entropy supertagger which assigns CCG lexical categories to words in a sentence. If we don't prune the

category candidates for each words, the packed chart is still too large to be computed in practice. The supertagger can greatly reduce the number of candidates per word from more than a hundred to around 2; when the supertagger is used, the accuracy remains around 95%.

2.3.2 Lexical Functional Grammar

Lexical Functional Grammar (LFG) [1] also belongs to the family of unification-based grammars and constituency-based grammars. In our work, we consider an LFG with the two most basic levels of representation: c(onstituent)-structure and f(unctional)-structure. The c-structure of a sentence captures basic syntactic information. It is similar to the context-free trees. The f-structure of a sentence represents abstract syntactic functions such as SUB(ject), OBJ(ect), PRED(icate), COMP(lement), XCOMP(lement), OBL(ique), ADJUNCT etc., in the form of recursive attribute-value structures. The f-structures are analogues to the predicate-argument representations. Each LFG analysis is also accompanied with a set of mapping equations in form of $\uparrow \ldots = \downarrow \ldots$ which represent the connections between the c-structure and the f-structure of a sentence. The goal of LFG parsing is to derive the c-structure, the f-structure and the mapping equations of a sentence.

In [7], Cahill et al. proposed an automatic annotator for LFG. First, the annotator decorated the trees in the Penn Treebank with LFG labels based on manually specified annotation rules. Second, a constraint solver is applied to derive the prototype f-structure and resolve the long-distance dependencies.

Statistical parsing models also exist for LFG. For example, [27] presented a log-linear model. In [28], Kaplan et al. applied a log-linear model to disambiguate the multiple analysis generated by the XLE parser, which is built based on manually crafted rules [4]. In [55], Riezler et al. introduced fragment grammar to improve the coverage of the XLE parser and applied log-linear model with incomplete data.

2.3.3 Head-Driven Phrase Structure Grammar

Head-Driven Phrase Structure Grammar (HPSG) [54] is another unification-based and constituency-based grammar formalism. Each word in the sentence is associated with a lexical entry, and a set of grammar rules and linguistic principles are applied in each derivation.

There have been several attempts to do HPSG parsing with log-linear models. In [41, 42], Miyao et al. proposed the feature forest model as a solution to the inference of a tree within discriminative framework. By representing the exponential number of possible structures using feature forests of a tractable size, the parameters of maximum entropy models are efficiently estimated via a variant of inside-outside algorithm.

Miyao et al. also explored different modeling methods: defining features on each predicate-argument dependency or each local derivation in an HPSG tree. This is similar to the dependency model and normal-form model in [11]. The performance of dependency model was similar to that of normal-form model.

Toutanova et al. presented experiment results on Redwoods treebank [49] using a similar log-linear model [57]. Their results were slightly worse than those from Miyao.

2.4 Cross-formalism Transfer Learning

For refined grammars like CCG and HPSG, manual construction of their treebank from scratch is extremely expensive. It needs a lot of linguistic expertise to annotate and is extremely time consuming. Therefore, researchers have been attempting to convert the existing annotations in coarse grammars (e.g. CFG) to the annotations in refined grammars (e.g. HPSG, LFG, or CCG). Traditional approaches were mainly based on manually specified rules. For instance, the rules may specify how to convert traces and functional tags in the Penn Treebank to the f-structure in LFG [5]. These conversion rules are typically utilized in two ways: (1) to create a new treebank which is consequently used to train a parser for the target formalism [25, 10, 40, 42], and (2) to translate the output of a CFG parser into the target formalism [7].

The design of these rules is a major linguistic and computational undertaking, which requires multiple iterations over the data to increase coverage [40, 48]. By nature, the mapping rules are formalism specific and therefore not transferable. Moreover, frequently designing such mappings involves modification to the original annotations. For instance, [25] made thousands of POS and constituent modifications to the Penn Treebank to facilitate transfer to CCG. More importantly, in some transfer scenarios, deterministic rules are not sufficient, due to the high ambiguity inherent in the mapping. Therefore, our work considers an alternative set-up for cross-formalism transfer where a small amount of annotations in the target formalism are used instead of deterministic rules.

The limitation of deterministic transfer rules has been recognized in prior work [55]. Their method uses a hand-crafted LFG parser to create a set of multiple parsing candidates for a given sentence. Using the partial mapping from CFG to LFG as guidance, the resulting trees are ranked based on their consistency with the labeled LFG bracketing imported from CFG. In contrast to this method, we neither require a parser for the target formalism nor manual rules for partial mapping. Consequently, our method can be applied to many different target grammar formalisms without significant engineering effort for each one. The utility of coarse-grained treebanks is determined by the degree of structural overlap with the target formalism.

In the next subsections, we briefly introduce the previous approaches of converting the Penn Treebank for each target formalism.

2.4.1 CCGbank

The CCGbank [25, 24] is a CCG version of the Penn Treebank. CCGbank was created by converting the phrase-structure trees in the Penn Treebank into CCG normal-form derivations. Some preprocessing of the treebank was required, including corrections of POS tags and modifications for tree structures. These changes allow the correct CCG analyses for constructions like coordination. [24] gives a detailed description of the procedure used to create CCGbank. Figure 2-1 shows an example normal-form derivation for a partial CCGbank sentence from [24].

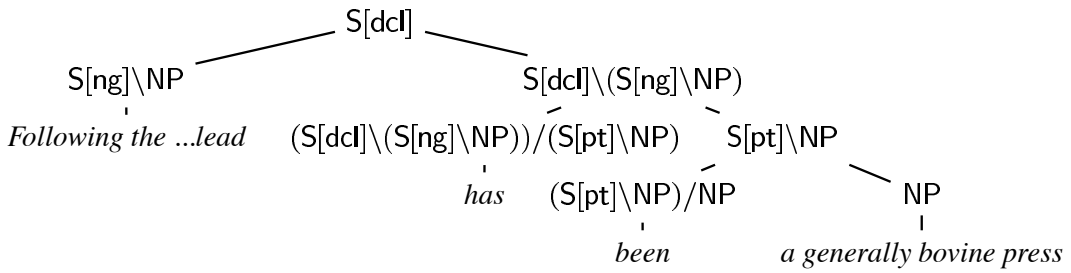


Figure 2-1: Example CCG partial derivation as a binary tree for the sentence *Following the . . .lead has been a generally bovine press.*

2.4.2 LFG Automatic Annotator

In [7], Cahill et al. presented models to automatically annotate the Penn Treebank with LFG labels. They first proposed an automatic f-structure annotation algorithm that annotates treebank trees with proto f-structure information based on a set of manually crafted annotation rules. Secondly, they presented two parsing architectures based on this algorithm. The first one is the pipeline architecture which first parsed a sentence into a PCFG tree using existing parsing model and then automatically annotated a CFG tree with their f-structure annotation algorithm. By contrast, in the integrated architecture they first automatically annotated the treebank trees with f-structure information and then extracted an annotated PCFG (A-PCFG) from the treebank. training dataset.

2.4.3 HPSGbank

In [40], Miyao et al. described a method of semi-automatically acquiring an English HPSG grammar from the Penn Treebank. First, they designed a set of heuristic rules manually and employed them to annotate the treebank with partially-specified derivation trees of HPSG. Second, lexical entries are automatically extracted from the annotated corpus by inversely applying HPSG schemata to partially-specified derivation trees. Third, they investigated the errors and inconsistencies during the automatic annotation and added rules or cleaned the treebank in order to get a better annotation result. After several iterations over these three steps, they built

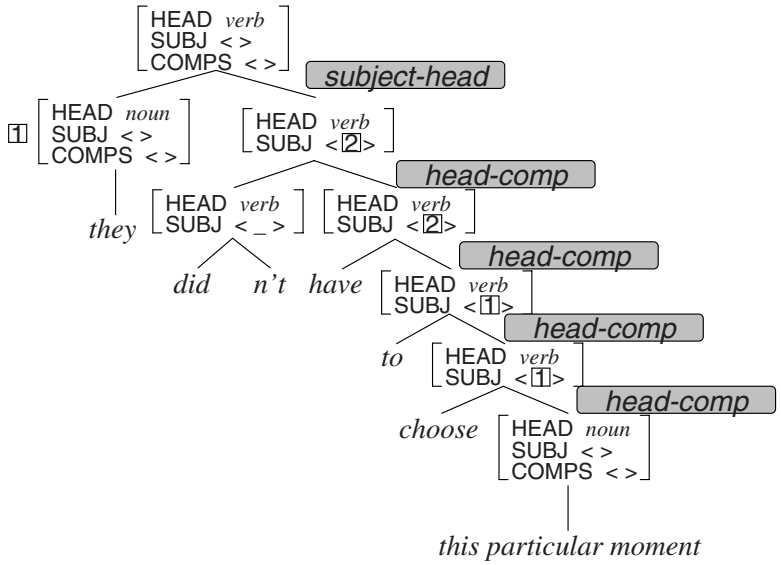


Figure 2-2: An example of partially specified derivation tree.

HPSGbank and extracted an HPSG grammar from it. Figure 2-2 shows an example of partially-specified derivation tree from [40].

2.5 Summary

In this section, we have briefly introduced some past work related to our tasks. Previous work on multilingual parsing has been emphasizing the importance of a universal POS tag representation for different languages. Traditional methods of acquiring such representation are based on the annotation of language-specific tags and the manual mapping from language-specific tags to the universal tagset. Our goal is to automate this process of finding the mappings.

We have also introduced some previous work on the parsing of different grammar formalisms. Though a plenty of parsing models already exist for each formalism, none of them have been evaluated on other formalisms. We propose a general parsing framework for constituency-based grammar, and evaluate it on all of the three target formalisms.

Chapter 3

Mapping into Universal Tagsets

3.1 Introduction

In recent work on multilingual dependency parsing, unified representations of POS tags have become essential because they bridge the gap between two different languages without using parallel data. Traditional ways of acquiring this universal representation are mainly based on manual mappings from language-specific tags to universal tagsets [51]. Despite the fact that manual mappings have achieved good empirical results as shown in [38] and [43], many obvious limiting factors still exist. For example, the human knowledge of mappings is needed for each new language and each new annotation scheme. Moreover, annotations of language-specific tags may not always be available in different scenarios.

We explore a method to automatically find a mapping from language-specific tags to universal tags. The basic idea behind our work is to extract information available in resource-rich languages (source) and exploit it to guide the mappings in resource-poor languages (target). We assume that the universal POS representation is available in the source language. As mentioned in Chapter 1, our central hypothesis is that a high quality mapping for target language yield to some linguistic properties which are consistent with the properties on the source language side. In particular, we consider three types of properties in our work: global distributional statistics, POS tag per sentence statistics and typological properties.

Having defined a quality measure for mappings, our goal is to find the optimal mapping. However, such partition optimization problems[1] are NP hard [18]. A naive approach to the problem is to greedily improve the map, but it turns out that this approach yields poor quality mappings. We therefore develop a method for optimizing over soft mappings, and use entropy regularization to drive those towards hard mappings. We construct the objective in a way that facilitates simple monotonically improving updates corresponding to solving convex optimization problems.

We evaluate our mapping approach on 19 languages that include representatives of Indo-European, Semitic, Basque, Japonic and Turkic families. We measure mapping quality based on the target language parsing accuracy. In addition to considering gold POS tags for the target language, we also evaluate the mapping algorithm on automatically induced POS tags. In all evaluation scenarios, our model consistently rivals the quality of manually induced mappings. We also demonstrate that the proposed inference procedure outperforms greedy methods by a large margin, highlighting the importance of good optimization techniques. We further show that while all characteristics of the mapping contribute to the objective, our largest gain comes from distributional features that capture global statistics. Finally, we establish that the mapping quality has a significant impact on the accuracy of syntactic transfer, which motivates further study of this topic.

3.2 Task Formulation

The input to our task consists of a *target* corpus written in a language T, and a set of non-parallel *source* corpora written in languages $\{S_1, \ldots, S_n\}$. In the source corpora, each word is annotated with both a language-specific POS tag and a universal POS tag [51]. In the target corpus each word is annotated only with a language-specific POS tag, either gold or automatically induced.

Our goal is to find a map from the set of L_T target language tags to the set of K universal tags. We assume that each language-specific tag is only mapped to

[1]Instances of related hard problems are 3-partition and subset-sum.

one universal tag, which means we never split a language-specific tag and $L_T \geq K$ holds for every language. We represent the map by a matrix A of size $K \times L_T$ where $A(c|f) = 1$ if the target language tag f is mapped to the universal tag c, and $A(c|f) = 0$ otherwise.[2] Note that each column of A should contain a single value of 1. We will later relax the requirement that $A(c|f) \in \{0, 1\}$. A candidate mapping A can be applied to the target language to produce sentences labeled with universal tags.

3.3 Model

In this section we describe an objective that reflects the quality of an automatic mapping.

Our key insight is that for a good mapping, the statistics over the universal tags should be similar for source and target languages because these tags play the same role cross-linguistically. For example, we should expect the frequency of a particular universal tag to be similar in the source and target languages.

One choice to make when constructing an objective is the source languages to which we want to be similar. It is clear that choosing all languages is not a good idea, since they are not all expected to have distributional properties similar to the target language. There is strong evidence that projecting from single languages can lead to good parsing performance [38]. Therefore, our strategy is to choose a single source language for comparison. The choice of the source language is based on similarity between typological properties; we describe this in detail in Section 3.4.

We must also determine which statistical properties we expect to be preserved across languages. Our model utilizes three linguistic phenomena which are consistent across languages: POS tag global distributional statistics, POS tag per sentence statistics, and typology-based ordering statistics. We define each of these below.

[2]We use c and f to reflect the fact that universal tags are a coarse version (hence c) of the language specific fine tags (hence f).

3.3.1 Mapping Characterization

We focus on three categories of mapping properties. For each of the relevant statistics we define a function $F_i(A)$ that has low values if the source and target statistics are similar.

Global distributional statistics: The unigram and bigram statistics of the universal tags are expected to be similar across languages with close typological profiles. We use $p_S(c_1, c_2)$ to denote the bigram distribution over universal tags in the source language, and $p_T(f_1, f_2)$ to denote the bigram distribution over language specific tags in the target language. The bigram distribution over universal tags in the target language depends on A and $p_T(f_1, f_2)$ and is given by:

$$p_T(c_1, c_2; A) = \sum_{f_1, f_2} A(c_1|f_1)A(c_2|f_2)p_T(f_1, f_2) \tag{3.1}$$

To enforce similarity between source and target distributions, we wish to minimize the KL divergence between the two: [3]

$$F_{bi}(A) = D_{KL}[p_S(c_1, c_2)|p_T(c_1, c_2; A)] \tag{3.2}$$

We similarly define $F_{uni}(A)$ as the distance between unigram distributions.

Per sentence statistics: Another defining property of POS tags is their average count per sentence. Specifically, we focus on the verb count per sentence, which we expect be similar across languages. To express this constraint, we use $n_v(s, A)$ to denote the number of verbs (i.e., the universal tags corresponding to verbs according to A) in sentence s. This is a linear function of A. We also use $E[n_v(s, A)]$ to denote the average number of verbs per sentence, and $V[n_v(s, A)]$ to denote the variance. We estimate these two statistics from the source language and denote them by E_{Sv}, V_{Sv}. Good mappings are expected to follow these patterns by having a variance

[3] We use the KL divergence because it assigns low weights to infrequent universal tags. Furthermore, this choice results in a simple, EM-like parameter estimation algorithm as discussed in Section 3.4.

upper bounded by V_{Sv} and an average lower bounded by E_{Sv}.[4] This corresponds to minimizing the following objectives:

$$\begin{aligned} F_{Ev}(A) &= \max\left[0, E_{Sv} - E[n_v(s,A)]\right] \\ F_{Vv}(A) &= \max\left[0, V[n_v(s,A)] - V_{Sv}\right] \end{aligned}$$

Note that the above objectives are convex in A, which will make optimization simpler. We refer to the two terms jointly as $F_{verb}(A)$.

Typology-based ordering statistics: Typological features can be useful for determining the relative order of different tags. If we know that the target language has a particular typological feature, we expect its universal tags to obey the given relative ordering. Specifically, we expect it to agree with ordering statistics for source languages with a similar typology. We consider two such features here. First, in pre-position languages the preposition is followed by the noun phrase. Thus, if T is such a language, we expect the probability of a noun phrase following the adposition to be high, i.e., cross some threshold. Formally, we define $C_1 = \{$noun, adj, num, pron, det$\}$ and consider the set of bigram distributions $\mathcal{S}_{\text{pre}}$ that satisfy the following constraint:

$$\sum_{c \in C_1} p_T(\text{adp},c) \geq a_{\text{pre}} \tag{3.3}$$

where $a_{\text{pre}} = \sum_{c \in C_1} p_S(\text{adp},c)$ is calculated from the source language. This constraint set is non-convex in A due to the bilinearity of the bigram term. To simplify optimization[5] we take an approach inspired by the posterior regularization method [17] and use the objective:

$$F_{\text{c}}(A) = \min_{r(c_1,c_2) \in \mathcal{S}_{\text{pre}}} D_{KL}[r(c_1,c_2) | p_T(c_1,c_2;A)] \tag{3.4}$$

[4]The rationale is that we want to put a lower bound on the number of verbs per sentence, and induce it from the source language. Furthermore, we expect the number of verbs to be well concentrated, and we induce its maximal variance from the source language.

[5]In Section 3.4 we shall see that this makes optimization easier.

The above objective will attain lower values for A such that $p_T(c_1, c_2; A)$ is close to the constraint set. Specifically, it will have a value of zero when the bigram distribution induced by A has the property specified in $\mathcal{S}_{\text{pre}}$. We similarly define a set $\mathcal{S}_{\text{post}}$ for post-positional languages.

As a second typological feature, we consider the Demonstrative-Noun ordering. In DN languages we want the probability of a determiner to come before $C_2 = \{$noun, adj, num$\}$, (i.e., frequent universal noun-phrase tags), to cross a threshold. This constraint translates to:

$$\sum_{c \in C_2} p_T(\text{det}, c) \geq a_{\text{det}} \tag{3.5}$$

where $a_{\text{det}} = \sum_{c \in C_2} p_S(\text{det}, c)$ is a threshold determined from the source language. We denote the set of distributions that have this property by $\mathcal{S}_{\text{DN}}$, and add them to the constraint in Equation 3.4. The overall constraint set is denoted by $\mathcal{S}$.

3.3.2 The Overall Objective

We have defined a set of functions $F_i(A)$ that are expected to have low values for good mappings. To combine those, we use a weighted sum: $F_\alpha(A) = \sum_i \alpha_i \cdot F_i(A)$. (The weights in this equation are learned; we discussed the procedure in Section 3.4)

Optimizing over the set of mappings is difficult since each mapping is a discrete set whose size is exponential size in L_T. Technically, the difficulty comes from the requirement that elements of A are integral and its columns sum to one. To relax this restriction, we will allow $A(c|f) \in [0, 1]$ and encourage A to correspond to a mapping by adding an entropy regularization term:

$$H[A] = -\sum_f \sum_c A(c|f) \log A(c|f) \tag{3.6}$$

This term receives its minimal value when the conditional probability of the universal tags given a language-specific tag is 1 for one universal tag and zero for the others.

The overall objective is then: $F(A) = F_\alpha(A) + \lambda \cdot H[A]$, where λ is the weight of

the entropy term.[6] The resulting optimization problem is:

$$\min_{A \in \Delta} F(A) \tag{3.7}$$

where Δ is the set of non-negative matrices whose columns sum to one:

$$\Delta = \left\{ A : \begin{array}{ll} A(c|f) \geq 0 & \forall c, f \\ \sum_{c=1}^{K} A(c|f) = 1 & \forall f \end{array} \right\} \tag{3.8}$$

3.4 Parameter Estimation

In this section we describe the parameter estimation process for our model. We start by describing how to optimize A. Next, we discuss the weight selection algorithm, and finally the method for choosing source languages.

3.4.1 Optimizing the Mapping A

Recall that our goal is to solve the optimization problem in Eq. Equation 3.7. This objective is non convex since the function $H[A]$ is concave, and the objective $F(A)$ involves bilinear terms in A and logarithms of their sums (see Equations Equation 3.1 and Equation 3.2).

While we do not attempt to solve the problem globally, we do have a simple update scheme that monotonically decreases the objective. The update can be derived in a similar manner to expectation maximization (EM) [45] and convex concave procedures [59]. Figure 3-1 describes our optimization algorithm. The key ideas in deriving it are using posterior distributions as in EM, and using a variational formulation of entropy. The term $F_c(A)$ is handled in a similar way to the posterior regularization algorithm derivation. A detailed derivation is provided in the Appendix A.

The k^{th} iteration of the algorithm involves several steps:

- In step 1, we calculate the current estimate of the bigram distribution over tags,

[6] Note that as $\lambda \to \infty$, only valid maps will be selected by the objective.

Initialize A^0.
Repeat

Step 1 (calculate current bigram estimate):

$$p_T(c_1, c_2; A^k) = \sum_{f_1, f_2} A^k(c_1|f_1) A^k(c_2|f_2) p_T(f_1, f_2)$$

Step 2 (incorporate constraints):

$$r^k(c_1, c_2) = \arg\min_{r \in \mathcal{S}} D_{KL}[r(c_1, c_2) | p_T(c_1, c_2; A^k)]$$

Step 3 (calculate model posterior):

$$p(f_1, f_2 | c_1, c_2; A^k) \propto A^k(c_1|f_1) A^k(c_2|f_2) p_T(f_1, f_2)$$

Step 4: (complete joint counts):

$$N^k(c_1, c_2, f_1, f_2) = p(f_1, f_2 | c_1, c_2; A^k) \left(r^k(c_1, c_2) + p_S(c_1, c_2) \right)$$

Step 5 (obtain pairwise):

$$M^k(c, f) = N_1^k(c, f) + N_2^k(c, f)$$

where $N_1^k(c, f) = \sum_{c_2, f_2} N^k(c, c_2, f, f_2)$ and similarly for $N_2^k(c, f)$.
Step 6 (M step with entropy linearization): Set A^{k+1} to be the solution of

$$\min_{A \in \Delta} - \sum_{c, f} \left[M^k(c, f) \log A(c|f) + A(c|f) \log A^k(c|f) \right] + F_{verb}(A)$$

Until Convergence of A^k

Figure 3-1: An iterative algorithm for minimizing our objective in Eq. Equation 3.7. For simplicity we assume that all the weights α_i and λ are equal to one. It can be shown that the objective monotonically decreases in every iteration.

$p_T(c_1, c_2; A^k)$.

- In step 2, we find the bigram distribution in the constraint set $\mathcal{S}$ that is closest in KL divergence to $p_T(c_1, c_2; A^k)$, and denote it by $r^k(c_1, c_2)$. This optimization problem is convex in $r(c_1, c_2)$.

- In step 3, we calculate the bigram posterior over language specific tags given a pair of universal tags. This is analogous to the standard E-step in EM.

- In step 4, we use the posterior in step 3 and the bigram distributions $p_S(c_1, c_2)$ and $r^k(c_1, c_2)$ to obtain joint counts over language specific and universal bigrams.

- In step 5, we use the joint counts from step 4 to obtain counts over pairs of

language specific and universal tags.

- In step 6, analogous to the M-step in EM, we optimize over the mapping matrix A. The objective is similar to the Q function in EM, and also includes the $F_{verb}(A)$ term, and a linear upper bound on the entropy term. The objective can be seen to be convex in A.

As mentioned above, each of the optimization problems in steps 2 and 6 is convex, and can therefore be solved using standard convex optimization solvers. Here, we use the CVX package [19, 20]. It can be shown that the algorithm improves $F(A)$ at every iteration and converges to a local optimum.

The above algorithm generates a mapping A that may contain fractional entries. To turn it into a *hard* mapping we round A by mapping each f to the c that maximizes $A(c|f)$ and then perform greedy improvement steps (one f at a time) to further improve the objective. The regularization constant λ is tuned to minimize the $F_\alpha(A)$ value of the rounded A.

3.4.2 Learning the Objective Weights

Our $F_\alpha(A)$ objective is a weighted sum of the individual $F_i(A)$ functions. In the following, we describe how to learn the α_i weights for every target language. We would like $F_\alpha(A)$ to have low values when A is a *good* map. Since our performance goal is parsing accuracy, we consider a map to be good if it results in high parsing accuracy, as measured when projecting a parser from to S to T.

Since we do not have annotated parses in T, we use the other source languages $S = \{S_1, \ldots, S_n\}$ to learn the weight. For each S_i as the target, we first train a parser for each language in $S \setminus \{S_i\}$ as if it was the source, using the map of [51], and choose $S_i^* \in S \setminus \{S_i\}$ which gives the highest parsing accuracy on S_i. Next we generate 7000 candidate mappings for S_i by randomly perturbing the map of [51]. We evaluate the quality of each candidate A by projecting the parser of S_i^* to S_i, and recording the parsing accuracy. Among all the candidates we choose the highest accuracy one and denote it by $A^*(S_i)$. We now want the score $F(A^*(S_i))$ to be lower than that of all

ID	Feature Description	Values
81A	Order of Subject, Object and Verb	SVO, SOV, VSO, VOS, OVS, OSV
85A	Order of Adposition and Noun	Postpositions, Prepositions, Inpositions
86A	Order of Genitive and Noun	Genitive-Noun, Noun-Genitive
87A	Order of Adjective and Noun	Adjective-Noun, Noun-Adjective
88A	Order of Demonstrative and Noun	Demonstrative-Noun, Noun-Demonstrative, before and after

Table 3.1: The set of typological features that we use for source language selection. The first column gives the ID of the feature as listed in WALS. The second column describes the feature and the last column enumerates the allowable values for each feature; besides these values each feature can also have a value of 'No dominant order'.

other candidates. To achieve this, we train a ranking SVM whose inputs are pairs of maps $A^*(S_i)$ and another *worse* $A(S_i)$. These map pairs are taken from many different traget languages, i.e. many different S_i. The features given to the SVM are the terms of the score $F_i(A)$. The goal of the SVM is to weight these terms such that the better map $A^*(S_i)$ has a lower score. The weights assigned by the SVM are taken as α_i.

3.4.3 Source Language Selection

As noted in Section 3.3 we construct $F(A)$ by choosing a single source language S. Here we describe the method for choosing S. Our goal is to choose S that is closest to T in terms of typology. Assume that languages are described by binary typological vectors $\mathbf{v}_L$. We would like to learn a diagonal matrix D such that $d(S,T;D) = (\mathbf{v}_S - \mathbf{v}_T)^T D(\mathbf{v}_S - \mathbf{v}_T)$ reflects the similarity between the languages. In our context, a good measure of similarity is the performance of a parser trained on S and projected on T (using the optimal map A). We thus seek a matrix D such that $d(S,T;D)$ is ranked according to the parsing accuracy. The matrix D is trained using an SVM ranking algorithm that tries to follow the ranking of parsing accuracy. Similar to the technique for learning the objective weights, we train across many pairs of source languages.[7]

The typological features we use are a subset of the features described in "The

[7]Ties are broken using the $F(A)$ objective.

World Atlas of Languages Structure" (WALS, [23]), and are shown in Table 3.1.

3.5 Evaluation Setup

Datasets We test our model on 19 languages: Arabic, Basque, Bulgarian, Catalan, Chinese, Czech, Danish, Dutch, English, German, Greek, Hungarian, Italian, Japanese, Portuguese, Slovene, Spanish, Swedish, and Turkish. Our data is taken from the CoNLL 2006 and 2007 shared tasks [2, 46]. The CoNLL datasets consist of manually created dependency trees and language-specific POS tags. Following [51], our model maps these language-specific tags to a set of 12 universal tags: noun, verb, adjective, adverb, pronoun, determiner, adposition, numeral, conjunction, particle, punctuation mark and X (a general tag).

Evaluation Procedure We perform a separate experiment for each of the 19 languages as the target and a source language chosen from the rest (using the method from Section 3.4.3). For the selected source language, we assume access to the mapping of [51].

Evaluation Measures We evaluate the quality of the derived mapping in the context of the target language parsing accuracy. In both the training and test data, the language-specific tags are replaced with universal tags: Petrov's tags for the source languages and learned tags for the target language. We train two non-lexicalized parsers using source annotations and apply them to the target language. The first parser is a non-lexicalized version of the MST parser [36] successfully used in the multilingual context [38]. In the second parser, parameters of the target language are estimated as a weighted mixture of parameters learned from supervised source languages [12]. For the parser of [12], we trained the model on the four languages used in the original paper — English, German, Czech and Italian. When measuring the performance on each of these four languages, we selected another set of four

languages with a similar level of diversity.[8]

Following the standard evaluation practice in parsing, we use directed dependency accuracy as our measure of performance.

Baselines We compare mappings induced by our model against three baselines: the manually constructed mapping of [51], a randomly constructed mapping and a greedy mapping. The greedy mapping uses the same objective as our full model, but optimizes it using a greedy method. In each iteration, this method makes $|L_T|$ passes over the language-specific tags, selecting a substitution that contributes the most to the objective.

Initialization To reduce the dimension of our algorithm's search space and speed up our method, we start by clustering the language-specific POS tags of the target into $|K| = 12$ clusters using an unsupervised POS induction algorithm [32].[9] Our mapping algorithm then learns the connection between these clusters and universal tags.

For initialization, we perform multiple random restarts and select the one with the lowest final objective score.

3.6 Experiment and Analysis

We first present the results of our model using the gold POS tags for the target language. Table 3.2 summarizes the performance of our model and the baselines.

Comparison against Baselines On average, the mapping produced by our model yields parsers with higher accuracy than all of the baselines. These results are consistent for both parsers [38, 12]. As expected, random mappings yield abysmal results

[8] We also experimented with a version of the [12] model trained on all the source languages. This set-up resulted in decreased performance. For this reason, we chose to train the model on the four languages.

[9] This pre-clustering results in about 3% improvement, presumably since it uses contextual information beyond what our algorithm does.

	Direct Transfer Parser (Accuracy)					Mixture Weight Parser (Accuracy)				Tag Diff.
	Random	Greedy	Petrov	Model	Best Pair	Random	Greedy	Petrov	Model.	
Catalan	15.9	32.5	74.8	**79.3**	79.3	12.6	24.6	65.6	**73.9**	8.8
Italian	16.4	41.0	**68.7**	68.3	71.4	11.7	33.5	**64.2**	61.9	6.7
Portuguese	15.8	24.6	72.0	**75.1**	75.1	10.7	14.1	70.4	**72.6**	12.2
Spanish	11.5	27.4	**72.1**	68.9	68.9	6.4	26.5	58.8	**62.8**	7.5
Danish	35.5	23.7	**46.6**	46.5	49.2	4.2	23.7	51.4	**51.7**	5.0
Dutch	18.0	22.1	**58.2**	56.8	57.3	7.1	15.3	**54.9**	53.2	4.9
English	14.7	19.0	**51.6**	49.0	49.0	13.3	15.1	**47.5**	41.8	17.7
German	15.8	24.3	**55.7**	50.4	51.6	20.9	18.7	**52.4**	51.8	15.0
Swedish	15.1	26.3	63.1	**63.1**	63.1	9.1	36.5	55.7	**55.9**	8.2
Bulgarian	17.4	28.0	51.6	**63.4**	63.4	22.6	39.9	**64.6**	60.4	35.7
Czech	19.0	34.4	47.7	**57.3**	57.3	12.7	26.2	48.3	**55.7**	28.5
Slovene	15.6	21.8	43.5	**51.4**	52.8	11.3	20.7	42.2	**53.0**	38.8
Greek	17.3	19.5	**62.3**	59.7	59.8	22.0	15.2	56.2	**57.0**	17.0
Hungarian	28.4	44.1	**53.8**	52.3	52.3	4.0	43.8	46.4	**51.7**	18.1
Arabic	22.1	45.4	**51.5**	51.2	52.9	3.9	40.9	48.3	**51.1**	15.7
Basque	18.0	19.2	27.9	**33.1**	35.1	6.3	8.3	**32.3**	30.6	43.8
Chinese	22.4	34.1	46.0	**47.6**	49.5	17.7	34.9	**44.0**	40.4	38.1
Japanese	36.5	46.2	51.4	**53.6**	53.6	15.4	18.0	25.7	**28.7**	73.8
Turkish	28.8	34.9	**53.2**	49.8	49.8	19.7	20.3	**27.7**	27.5	9.9
Average	20.2	29.9	55.4	**56.7**	57.4	12.7	25.4	50.8	**51.7**	21.3

Table 3.2: Directed dependency accuracy of our model and the baselines using gold POS tags for the target language. The first section of the table is for the direct transfer of the MST parser [38]. The second section is for the weighted mixture parsing model [12]. The first two columns (Random and Greedy) of each section present the parsing performance with a random or a greedy mapping. The third column (Petrov) shows the results when the mapping of [51] is used. The fourth column (Model) shows the results when our mapping is used and the fifth column in the first section (Best Pair) shows the performance of our model when the best source language is selected for every target language. The last column (Tag Diff.) presents the difference between our mapping and the mapping of [51] by showing the percentage of target language tokens for which the two mappings select a different universal tag.

— 20.2% and 12.7% for the two parsers. The low accuracy of parsers that rely on the *Greedy* mapping — 29.9% and 25.4% — show that a greedy approach is a poor strategy for mapping optimization.

Surprisingly, our model slightly outperforms the mapping of [51], yielding an average accuracy of 56.7% as compared to the 55.4% achieved by its manually constructed counterpart for the direct transfer method [38]. Similar results are observed for the mixture weights parser [12]. The main reason for these differences comes from mistakes introduced in the manual mapping. For example, in Czech tag "R" is labeled as "pronoun", while actually it should be mapped to "adposition". By correcting this

mistake, we gain 5% in parsing accuracy for the direct transfer parser.

Overall, the manually constructed mapping and our model's output disagree on 21% of the assignments (measured on the token level). However, the extent of disagreement is not necessarily predictive of the difference in parsing performance. For instance, the manual and automatic mappings for Catalan disagree on 8% of the tags and their parsing accuracy differs by 5%. For Greek on the other hand, the disagreement between mappings is much higher — 17%, yet the parsing accuracy is very close. This phenomenon shows that not all mistakes have equal weight. For instance, a confusion between "pronoun" and "noun" is less severe in the parsing context than a confusion between "pronoun" and "adverb".

Impact of Language Selection To assess the quality of our language selection method, we compare the model against an oracle that selects the best source for a given target language. As Table 3.2 shows our method is very close to the oracle performance, with only 0.7% gap between the two. In fact, for 10 languages our method correctly predicts the best pairing. This result is encouraging in other contexts as well. Specifically, [38] have demonstrated that projecting from a single oracle-chosen language can lead to good parsing performance, and our technique may allow such projection without an oracle.

Relations between Objective Values and Optimization Performance The suboptimal performance of the Greedy method shows that choosing a good optimization strategy plays a critical role in finding the desired mapping. A natural question to ask is whether the objective value is predictive of the end goal parsing performance. Figure 3-2 shows the objective values for the mappings computed by our method and the baselines for four languages. Overall, our method and the manual mappings reach similar values, both considerably better than other baselines. While the parsing performance correlates with the objective, the correlation is not perfect. For instance, on Greek our mapping has a better objective value, but lower parsing performance.

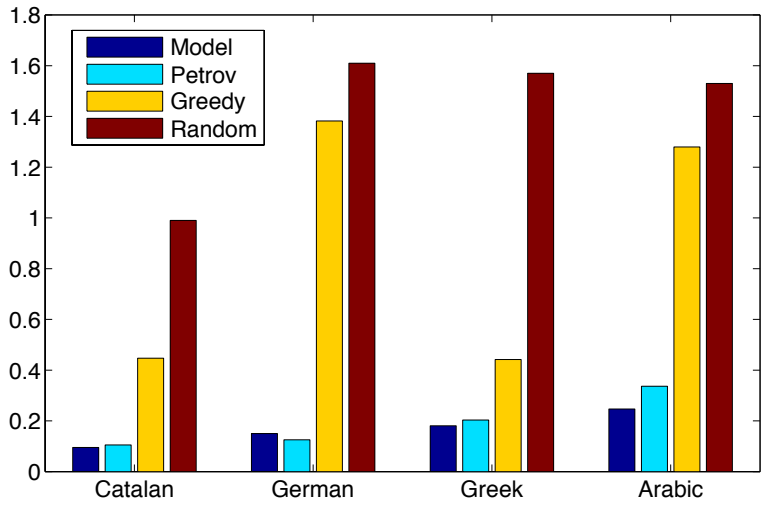


Figure 3-2: Objective values for the different mappings used in our experiments for four languages. Note that the goal of the optimization procedure is to minimize the objective value.

Ablation Analysis We next analyze the contribution of each component of our objective to the resulting performance.[10] The strongest factor in our objective is the distributional features capturing global statistics. Using these features alone achieves an average accuracy of 51.1%, only 5.6% less than the full model score. Adding just the verb-related constraints to the distributional similarity objectives improves the average model performance by 2.1%. Adding just the typological constraints yields a very modest performance gain of 0.5%. This is not surprising — the source language is selected to be typologically similar to the target language, and thus its distributional properties are consistent with typological features. However, adding both the verb-related constraints and the typological constraints results in a synergistic performance gain of 5.6% over the distributional similarity objective, a gain which is much better than the sum of the two individual gains.

Application to Automatically Induced POS Tags A potential benefit of the proposed method is to relate automatically induced clusters in the target language to

[10]The results are consistent for both parsers, here we report the accuracy for the direct transfer method [38].

universal tags. In our experiments, we induce such clusters using Brown clustering,[11] which has been successfully used for similar purposes in parsing research [31]. We then map these clusters to the universal tags using our algorithm.

The average parsing accuracy on the 19 languages is 45.5%. Not surprisingly, automatically induced tags negatively impact parsing performance, yielding a decrease of 11% when compared to mappings obtained using manual POS annotations (see Table 3.2). To further investigate the impact of inaccurate tags on the mapping performance, we compare our model against the oracle mapping model that maps each cluster to the most common universal tag of its members. Parsing accuracy obtained using this method is 45.1%, closely matching the performance of our mapping algorithm.

An alternative approach to mapping words into universal tags is to directly partition words into K clusters (without passing through language specific tags). In order for these clusters to be meaningful as universal tags, we can provide several prototypes for each cluster (e.g., "walk" is a verb etc.). To test this approach we used the prototype driven tagger of [22] with 15 prototypes per universal tag.[12] The resulting universal tags yield an average parsing accuracy of 40.5%. Our method (using Brown clustering as above) outperforms this baseline by about 5%.

3.7 Summary

In this chapter, we present an automatic method for mapping language-specific POS tags to a set of universal tags. Our work capitalizes on manually designed conversion schemes to automatically create mappings for new languages. Our experimental results demonstrate that automatically induced mappings rival the quality of their hand-crafted counterparts. We also establish that the mapping quality has a significant impact on the accuracy of syntactic transfer, which motivates further study of

[11] In our experiments, we employ Liang's implementation `http://cs.stanford.edu/~pliang/software/`. The number of clusters is set to 30.

[12] Oracle prototypes were obtained by taking the 15 most frequent words for each universal tag. This yields almost the same total number of prototypes as those in the experiment of [22].

this topic. Finally, our experiments show that the choice of mapping optimization scheme plays a crucial role in the quality of the derived mapping, highlighting the importance of optimization for the mapping task.

Chapter 4

Transfer Learning for Constituency-based Grammars

4.1 Introduction

The goal of this task is to improve parsing performance for constituency-based grammars. Specifically, we assume that we have a large treebank annotated in some coarse constituency-based grammar. A natural choice is the Penn Treebank, which is annotated in context-free grammar. Besides this source treebank, we also have another small treebank annotated in a more refined target grammar formalism, such as CCG. Our central hypothesis is that the target and source grammars shared a similar basic tree structure. With the help of the source treebank, we achieve a better parsing performance for the target grammar than the performance achieved by training on the small treebank only.

In order to train a high quality parser, we need two sources of information: (1) basic syntactic structure information which forms the tree skeleton and (2) formalism-specific labels which decorate the tree and represent predicate-argument dependency relations based on the grammar theory. Our source treebank contains rich syntactic structure information and our proposed model automatically utilizes this information to improve the parsers in target grammar formalism.

To enable effective transfer, the model has to identify shared structural compo-

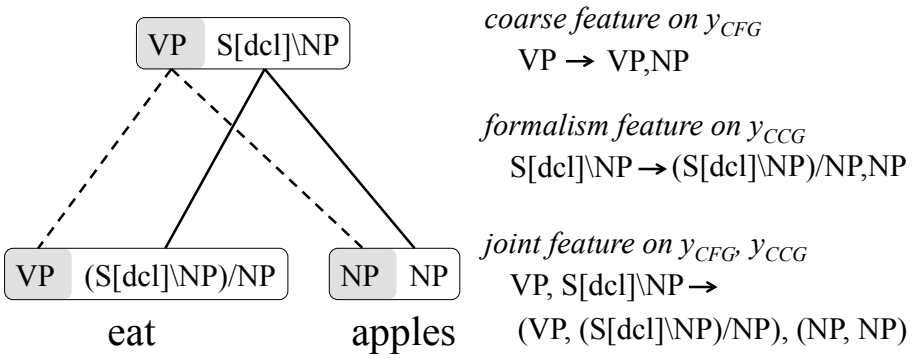


Figure 4-1: Illustration of the joint CCG-CFG representation. The shadowed labels correspond to the CFG derivation y_{CFG}, whereas the other labels correspond to the CCG derivation y_{CCG}. Note that the two derivations share the same (binarized) tree structure. Also shown are features that are turned on for this joint derivation (see Section 4.5).

nents between the formalisms despite the apparent differences. Moreover, we do not assume parallel annotations. To this end, our model jointly parses the two corpora according to the corresponding annotations, enabling transfer via parameter sharing. In particular, we augment each target tree node with hidden variables that capture the connection to the coarse annotations. Specifically, each node in the target tree has two labels: an entry which is specific to the target formalism, and a latent label containing a value from the Penn Treebank tagset, such as NP (see Figure 4-1). This design enables us to represent three types of features: the target formalism-specific features, the coarse formalism features, and features that connect the two. This modeling approach makes it possible to perform transfer to a range of target formalisms, without manually drafting formalism-specific rules.

We evaluate our approach on three constituency-based grammars — CCG, HPSG, and LFG. As a source of coarse annotations, we use the Penn Treebank.[1] Our results clearly demonstrate that for all three formalisms, parsing accuracy can be improved by training with additional coarse annotations. For instance, the model trained on 500 HPSG sentences achieves a labeled dependency F-score of 72.3%. Adding 15,000 Penn Treebank sentences during training leads to a 78.5% labeled dependency F-score,

[1]While the Penn Treebank-2 contains richer annotations, we decided to use the Penn Treebank-1 to demonstrate the feasibility of transfer from coarse annotations.

an absolute improvement of 6.2%. To achieve similar performance in the absence of coarse annotations, the parser has to be trained on about 1,500 sentences, namely three times what is needed when using coarse annotations. Similar results are also observed on CCG and LFG formalisms.

4.2 Task Formulation

Recall that our goal is to learn how to parse the target formalisms while using two annotated sources: a small set of sentences annotated in the target formalism (e.g., CCG), and a large set of sentences with coarse annotations. For the latter, we use the CFG parses from the Penn Treebank. For simplicity we focus on the CCG formalism in what follows. We also generalize our model to other formalisms, as explained in Section 4.4.4.

Our notations are as follows: an input sentence is denoted by S. A CFG parse is denoted by y_{CFG} and a CCG parse is denoted by y_{CCG}. Clearly the set of possible values for y_{CFG} and y_{CCG} is determined by S and the grammar. The training set is a set of N sentences $S_1, \ldots, S_N$ with CFG parses $y^1_{CFG}, \ldots, y^N_{CFG}$, and M sentences $\bar{S}_1, \ldots, \bar{S}_M$ with CCG parses $y^1_{CCG}, \ldots, y^M_{CCG}$. It is important to note that we do not assume we have parallel data for CCG and CFG.

Our goal is to use such a corpus for learning how to generate CCG parses to unseen sentences.

4.3 A Joint Model for Two Formalisms

The key idea behind our work is to learn a joint distribution over CCG and CFG parses. Such a distribution can be marginalized to obtain a distribution over CCG or CFG and is thus appropriate when the training data is not parallel, as it is in our setting.

It is not immediately clear how to jointly model the CCG and CFG parses, which are structurally quite different. Furthermore, a joint distribution over these will

become difficult to handle computationally if not constructed carefully. To address this difficulty, we make several simplifying assumptions. First, we assume that both parses are given in normal form, i.e., they correspond to binary derivation trees. CCG parses are already provided in this form in CCGBank. CFG parses in the Penn Treebank are not binary, and we therefore binarize them, as explained in Section 4.4.3.

Second, we assume that any y_{CFG} and y_{CCG} jointly generated must share the same derivation tree structure. This makes sense. Since both formalisms are constituency-based, their trees are expected to describe the same constituents. We denote the set of valid CFG and CCG joint parses for sentence S by $\mathcal{Y}(\mathcal{S})$.

The above two simplifying assumptions make it easy to define joint features on the two parses, as explained in Section 4.5. The representation and features are illustrated in Figure 4-1.

We shall work within the discriminative framework, where given a sentence we model a distribution over parses. As is standard in such settings, the distribution will be log-linear in a set of features of these parses. Denoting $y = (y_{CFG}, y_{CCG})$, we seek to model the distribution $p(y|S)$ corresponding to the probability of generating a pair of parses (CFG and CCG) given a sentence. The distribution thus has the following form:

$$p_{joint}(y|S;\theta) = \frac{1}{Z(S;\theta)} e^{f(y,S)\cdot\theta} \ . \tag{4.1}$$

where θ is a vector of parameters to be learned from data, and $f(y, S)$ is a feature vector. $Z(S;\theta)$ is a normalization (partition) function normalized over $y \in \mathcal{Y}(\mathcal{S})$ the set of valid joint parses.

The feature vector contains three types of features: CFG specific, CCG specific and joint CFG-CCG. We denote these by $f_{CFG}, f_{CCG}, f_{joint}$. These depend on y_{CCG}, y_{CFG} and y respectively. Accordingly, the parameter vector θ is a concatenation of $\theta_{CCG}, \theta_{CFG}$ and θ_{joint}.

As mentioned above, we can use Equation 4.1 to obtain distributions over y_{CCG} and y_{CFG} via marginalization. For the distribution over y_{CCG} we do precisely this,

namely use:

$$p_{CCG}(y_{CCG}|S;\theta) = \sum_{y_{CFG}} p_{joint}(y|S;\theta) \tag{4.2}$$

For the distribution over y_{CFG} we could have marginalized p_{joint} over y_{CCG}. However, this computation is costly for each sentence, and has to be repeated for all the sentences. Instead, we assume that the distribution over y_{CFG} is a log-linear model with parameters θ_{CFG} (i.e., a sub-vector of θ) , namely:

$$p_{CFG}(y_{CFG}|S;\theta_{CFG}) = \frac{e^{f_{CFG}(y_{CFG},S)\cdot\theta_{CFG}}}{Z(S;\theta_{CFG})} \ . \tag{4.3}$$

Thus, we assume that both p_{joint} and p_{CFG} have the same dependence on the f_{CFG} features.

The Likelihood Objective: Given the models above, it is natural to use maximum likelihood to find the optimal parameters. To do this, we define the following regularized likelihood function:

$$L(\theta) = \sum_{i=1}^{N} \log\left(p_{CFG}(y^i_{CFG}|S_i,\theta_{CFG})\right) + \sum_{i=1}^{M} \log\left(p_{CCG}(y^i_{CCG}|\bar{S}_i,\theta)\right) - \frac{\lambda}{2}\|\theta\|_2^2$$

where p_{CCG} and p_{CFG} are defined in Equations 4.2 and 4.3 respectively. The last term is the l_2-norm regularization. Our goal is then to find a θ that maximizes $L(\theta)$.

Training Algorithm: For maximizing $L(\theta)$ w.r.t. θ we use the limited-memory BFGS algorithm [47]. Calculating the gradient of $L(\theta)$ requires evaluating the expected values of $f(y,S)$ and f_{CFG} under the distributions p_{joint} and p_{CFG} respectively. This can be done via the inside-outside algorithm.[2]

[2]To speed up the implementation, gradient computation is parallelized, using the Message Passing Interface package [21].

Parsing Using the Model: To parse a sentence S, we calculate the maximum probability assignment for $p_{joint}(y|S;\theta)$.[3] The result is both a CFG and a CCG parse. Here we will mostly be interested in the CCG parse. The joint parse with maximum probability is found using a standard CYK chart parsing algorithm. The chart construction will be explained in Section 4.4.

4.4 Implementation

This section introduces important implementation details, including supertagging, feature forest pruning and binarization methods. Finally, we explain how to generalize our model to other constituency-based formalisms.

4.4.1 Supertagging

When parsing a target formalism tree, one needs to associate each word with a lexical entry. However, since the number of candidates is typically more than one thousand, the size of the chart explodes. One effective way of reducing the number of candidates is via supertagging [11]. A supertagger is used for selecting a small set of lexical entry candidates for each word in the sentence. We use the tagger in [11] as a general suppertagger for all the grammars considered. The only difference is that we use different lexical entries in different grammars.

4.4.2 Feature Forest Pruning

In the BFGS algorithm (see Section 4.3), feature expectation is computed using the inside-outside algorithm. To perform this dynamic programming efficiently, we first need to build the packed chart, namely the feature forest [39] to represent the exponential number of all possible tree structures. However, a common problem for lexicalized grammars is that the forest size is too large. In CFG, the forest is pruned

[3] An alternative approach would be to marginalize over y_{CFG} and maximize over y_{CCG}. However, this is a harder computational problem.

according to the inside probability of a simple generative PCFG model and a prior [14]. The basic idea is to prune the trees with lower probability. For the target formalism, a common practice is to prune the forest using the supertagger [11, 39]. In our implementation, we applied all pruning techniques, because the forest is a combination of CFG and target grammar formalisms (e.g., CCG or HPSG).

4.4.3 Binarization

We assume that the derivation tree in the target formalism is in a normal form, which is indeed the case for the treebanks we consider. As mentioned in Section 4.3, we would also like to work with binarized CFG derivations, such that all trees are in normal form and it is easy to construct features that link the two (see Section 4.5).

Since Penn Treebank trees are not binarized, we construct a simple procedure for binarizing them. The procedure is based on the available target formalism parses in the training corpus, which are binarized. We illustrate it with an example. In what follows, we describe derivations using the POS of the head words of the corresponding node in the tree. This makes it possible to transfer binarization rules between formalisms.

Suppose we want to learn the binarization rule of the following derivation in CFG:

$$\text{NN} \rightarrow (\text{DT} \quad \text{JJ} \quad \text{NN}) \tag{4.4}$$

We now look for binary derivations with these POS in the target formalism corpus, and take the most common binarization form. For example, we may find that the most common binarization to binarize the CFG derivation in Equation 4.4 is:

$$\text{NN} \rightarrow (\text{DT} \quad (\text{JJ} \quad \text{NN}))$$

If no (DT JJ NN) structure is observed in the CCG corpus, we first apply the binary branching on the children to the left of the head, and then on the children to the right of the head.

We also experiment with using fixed binarization rules such as left/right branching, instead of learning them. This results in a drop on the dependency F-score by about 5%.

4.4.4 Implementation in Other Formalisms

We introduce our model in the context of CCG, but the model can easily be generalized to other constituency-based grammars, such as HPSG and LFG. In a derivation tree, the formalism-specific information is mainly encoded in the lexical entries and the applied grammar rules, rather than the tree structures. Therefore we only need to change the node labels and lexical entries to the language-specific ones, while the framework of the model remains the same.

4.5 Features

Feature functions in log-linear models are designed to capture the characteristics of each derivation in the tree. In our model, as mentioned in Section 4.1, the features are also defined to enable information transfer between coarse and rich formalisms. In this section, we first introduce how different types of feature templates are designed, and then show an example of how the features help transfer the syntactic structure information. Note that the same feature templates are used for all the target grammar formalisms.

Recall that our y contains both the CFG and CCG parses, and that these use the same derivation tree structure. Each feature will consider either the CFG derivation, the CCG derivation or these two derivations jointly.

The feature construction is similar to constructions used in previous work [39]. The features are based on the atomic features listed in Table 4.1. These will be used to construct $f(y, S)$ as explained next.

We define the following feature templates: f_{binary} for binary derivations, f_{unary} for unary derivations, and f_{root} for the root nodes. These use the atomic features in

hl	lexical entries/CCG categories of the head word
r	grammar rules, i.e. HPSG schema, resulting CCG categories, LFG mapping equations
sy	CFG syntactic label of the node (e.g. NP, VP)
d	distance between the head words of the children
c	whether a comma exists between the head words of the children
sp	the span of the subtree rooted at the node
hw	surface form of the head word of the node
hp	part-of-speech of the head word
p_i	part-of-speech of the i-th word in the sentence

Table 4.1: Templates of atomic features.

Table 4.1, resulting in the following templates:

$$f_{binary} = \left\langle \begin{array}{l} r, sy_p, d, c \\ sy_l, sp_l, hw_l, hp_l, hl_l, \\ sy_r, sp_r, hw_r, hp_r, hl_r, \\ p_{st-1}, p_{st-2}, p_{en+1}, p_{en+2} \end{array} \right\rangle$$

$$f_{unary} = \langle r, sy_p, hw, hp, hl \rangle$$

$$f_{root} = \langle sy, hw, hp, hl \rangle$$

In the above we used the following notation: p, l, r denote the parent node and left/right child node, and st, en denote the starting and ending index of the constituent.

We also consider templates with subsets of the above features. The final list of binary feature templates is shown in Table 4.2. It can be seen that some features depend only on the CFG derivations (i.e., those without r,hl), and are thus in $f_{CFG}(y, S)$. Others depend only on CCG derivations (i.e., those without sy), and are in $f_{CCG}(y, S)$. The rest depend on both CCG and CFG and are thus in $f_{joint}(y, S)$.

Note that after binarization, grandparent and sibling information becomes very important in encoding the structure. However, we limit the features to be designed locally in a derivation in order to run inside-outside efficiently. Therefore we use the preceding and succeeding POS tag information to approximate the grandparent and sibling information. Empirically, these features yield a significant improvement on the constituent accuracy.

f_{CFG}	$\langle d, w_{l,r}, hp_{l,r}, sy_{p,l,r}\rangle, \langle d, w_{l,r}, sy_{p,l,r}\rangle,$ $\langle c, w_{l,r}, hp_{l,r}, sy_{p,l,r}\rangle, \langle c, w_{l,r}, sy_{p,l,r}\rangle,$ $\langle d, c, hp_{l,r}, sy_{p,l,r}\rangle, \langle d, c, sy_{p,l,r}\rangle,$ $\langle c, sp_{l,r}, hp_{l,r}, sy_{p,l,r}\rangle, \langle c, sp_{l,r}, sy_{p,l,r}\rangle,$ $\langle p_{st-1}, sy_{p,l,r}\rangle, \langle p_{en+1}, sy_{p,l,r}\rangle,$ $\langle p_{st-1}, p_{en+1}, sy_{p,l,r}\rangle,$ $\langle p_{st-1}, p_{st-2}, sy_{p,l,r}\rangle, \langle p_{en+1}, p_{en+2}, sy_{p,l,r}\rangle,$ $\langle p_{st-1}, p_{st-2}, p_{en+1}, p_{en+2}, sy_{p,l,r}\rangle,$
f_{CCG}	$\langle r, d, c, hw_{l,r}, hp_{l,r}, hl_{l,r}\rangle, \langle r, d, c, hw_{l,r}, hp_{l,r}\rangle$ $\langle r, d, c, hw_{l,r}, hl_{l,r}\rangle,$ $\langle r, c, sp_{l,r}, hw_{l,r}, hp_{l,r}, hl_{l,r}\rangle$ $\langle r, c, sp_{l,r}, hw_{l,r}, hp_{l,r}, \rangle, \langle r, c, sp_{l,r}, hw_{l,r}, hl_{l,r}\rangle$ $\langle r, d, c, hp_{l,r}, hl_{l,r}\rangle, \langle r, d, c, hp_{l,r}\rangle, \langle r, d, c, hl_{l,r}\rangle$ $\langle r, c, hp_{l,r}, hl_{l,r}\rangle, \langle r, c, hp_{l,r}\rangle, \langle r, c, hl_{l,r}\rangle$
f_{joint}	$\langle r, d, c, sy_{l,r}, hl_{l,r}\rangle, \langle r, d, c, sy_{l,r}\rangle$ $\langle r, c, sp_{l,r}, sy_{l,r}, hl_{l,r}\rangle, \langle r, c, sp_{l,r}, sy_{l,r}\rangle$

Table 4.2: Binary feature templates used in $f(y, S)$. Unary and root features follow a similar pattern.

In order to apply the same feature templates to other target formalisms, we only need to assign the atomic features r and hl with the formalism-specific values. We do not need extra engineering work on redesigning the feature templates.

Figure 4-2 gives an example in CCG of how features help transfer the syntactic information from the Penn Treebank and learn the correspondence to the formalism-specific information. From the Penn Treebank CFG annotations, we can learn that the derivation VP→(VP, NP) is common, as shown on the left of Figure 4-2. In a CCG tree, this tendency will encourage the y_{CFG} (latent) variables to take this CFG parse. Then weights on the f_{joint} features will be learned to model the connection between the CFG and CCG labels. Moreover, the formalism-specific features f_{CCG} can also encode the formalism-specific syntactic and semantic information. These

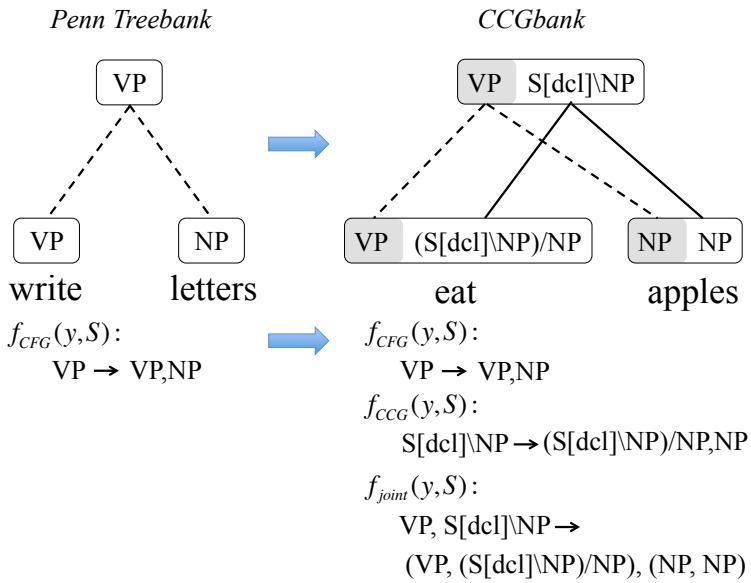


Figure 4-2: Example of transfer between CFG and CCG formalisms.

three types of features work together to generate a tree skeleton and fill in the CFG and CCG labels.

4.6 Evaluation Setup

Datasets: As a source of coarse annotations, we use the Penn Treebank-1 [34]. In addition, for CCG, HPSG and LFG, we rely on formalism-specific corpora developed in prior research [25, 40, 7, 29]. All of these corpora were derived via conversion of the Penn Treebank to the target formalisms. In particular, our CCG and HPSG datasets were converted from the Penn Treebank based on handcrafted rules [25, 40]. Table 4.3 shows which sections of the treebanks were used in training, testing and development for both formalisms. Our LFG training dataset was constructed in a similar fashion [7]. However, we choose to use PARC700 as our LFG tesing and development datasets, following the previous work by [28]. It contains 700 manually annotated sentences that are randomly selected from the Penn Treebank Section 23. The split of PARC700 follows the setting in [28]. Since our model does not assume parallel data, we use distinct sentences in the source and target treebanks. Following previous work [24, 42], we only consider sentences not exceeding 40 words, except on

PARC700 where all sentences are used.

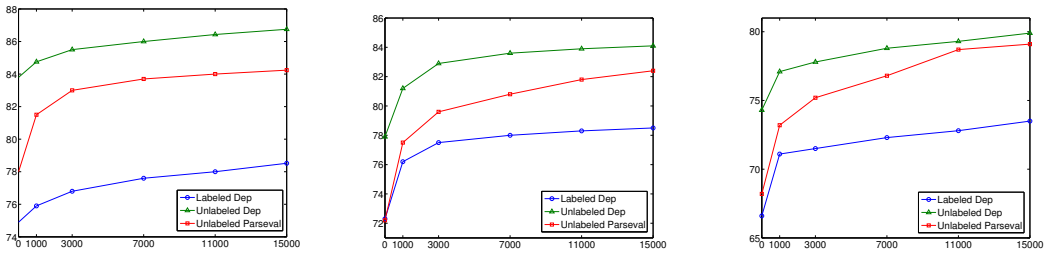
Grammar	Train	Dev.	Test
CCG	Sec. 02-21	Sec. 00	Sec. 23
HPSG			
LFG		140 sents. in PARC700	560 sents. in PARC700

Table 4.3: Training/Dev./Test split on WSJ sections and PARC700 for different grammar formalisms.

Evaluation Metrics: We use two evaluation metrics. First, following previous work, we evaluate our method using the labeled and unlabeled predicate-argument dependency F-score. This metric is commonly used to measure parsing quality for the formalisms considered in this paper. The detailed definition of this measure as applied for each formalism is provided in [10, 42, 6]. For CCG, we use the evaluation script from the C&C tools.[4] For HPSG, we evaluate all types of dependencies, including punctuations. For LFG, we consider the *preds-only* dependencies, which are the dependencies between pairs of words. Secondly, we also evaluate using unlabeled PARSEVAL, a standard measure for PCFG parsing [53, 9, 8, 13]. The dependency F-score captures both the target-grammar labels and tree-structural relations. The unlabeled PARSEVAL is used as an auxiliary measure that enables us to separate these two aspects by focusing on the structural relations exclusively.

Training without CFG Data: To assess the impact of coarse data in the experiments below, we also consider the model trained only on formalism-specific annotations. When no CFG sentences are available, we assign all the CFG labels to a special value shared by all the nodes. In this set-up, the model reduces to a normal log-linear model for the target formalism.

[4]Available at http://svn.ask.it.usyd.edu.au/trac/candc/wiki



(a) CCG (b) HPSG (c) LFG

Figure 4-3: Model performance with 500 target formalism trees and different numbers of CFG trees, evaluated using labeled/unlabeled dependency F-score and unlabeled PARSEVAL.

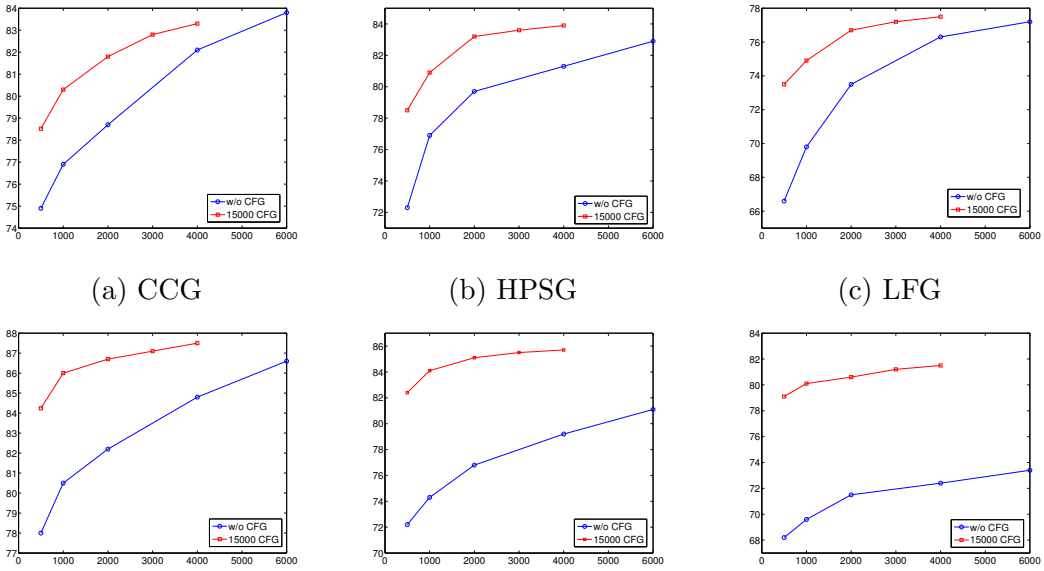
Parameter Settings: During training, all the feature parameters θ are initialized to zero. The hyperparameters used in the model are tuned on the development sets. We noticed, however, that the resulting values are consistent across different formalisms. In particular, we set the l_2-norm weight to $\lambda = 1.0$, the supertagger threshold to $\beta = 0.01$, and the PCFG pruning threshold to $\alpha = 0.002$.

4.7 Experiment and Analysis

Impact of Coarse Annotations on Target Formalism: To analyze the effectiveness of annotation transfer, we fix the number of annotated trees in the target formalism and vary the amount of coarse annotations available to the algorithm during training. In particular, we use 500 sentences with formalism-specific annotations, and vary the number of CFG trees from zero to 15,000.

As Figure 4-3 shows, CFG data boosts parsing accuracy for all the target formalisms. For instance, there is a gain of 6.2% in labeled dependency F-score for HPSG formalism when 15,000 CFG trees are used. Moreover, increasing the number of coarse annotations used in training leads to further improvement on different evaluation metrics.

Tradeoff between Target and Coarse Annotations: We also assess the relative contribution of coarse annotations when the size of annotated training corpus in the



(a) CCG (b) HPSG (c) LFG

(d) CCG (e) HPSG (f) LFG

Figure 4-4: Model performance with different target formalism trees and zero or 15,000 CFG trees. The first row shows the results of labeled dependency F-score and the second row shows the results of unlabeled PARSEVAL.

target formalism varies. In this set of experiments, we fix the number of CFG trees to 15,000 and vary the number of target annotations from 500 to 4,000. Figure 4-4 shows the relative contribution of formalism-specific annotations compared to that of the coarse annotations. For instance, Figure 4-4a shows that the parsing performance achieved using 2,000 CCG sentences can be achieved using approximately 500 CCG sentences when coarse annotations are available for training. More generally, the result convincingly demonstrates that coarse annotations are helpful for all the sizes of formalism-specific training data. As expected, the improvement margin decreases when more formalism-specific data is used.

Figure 4-4 also illustrates a slightly different characteristics of transfer performance between two evaluation metrics. Across all three grammars, we can observe that adding CFG data has a more pronounced effect on the PARSEVAL measure than the dependency F-score. This phenomenon can be explained as follows. The unlabeled PARSEVAL score (Figure 4-4d-f) mainly relies on the coarse structural information. On the other hand, predicate-argument dependency F-score (Figure 4-4a-c) also relies

on the target grammar information. Because that our model only transfers structural information from the source treebank, the gains of PARSEVAL are expected to be larger than that of dependency F-score.

Comparison to State-of-the-art Parsers: We would also like to demonstrate that the above gains of our transfer model are achieved using an adequate formalism-specific parser. Since our model can be trained exclusively on formalism-specific data, we can compare it to state-of-the-art formalism-specific parsers. For this experiment, we choose the C&C parser [10] for CCG, Enju parser [42] for HPSG and pipeline automatic annotator [6] with Charniak parser for LFG. For all three parsers, we use the implementation provided by the authors with the default parameter values. All the models are trained on either 1,000 or 15,000 sentences annotated with formalism-specific trees, thus evaluating their performances on small scale or large scale of data. As Table 4.4 shows, our model is competitive with all the baselines described above. It's not surprising that Cahill's model outperforms our log-linear model because it relies heavily on handcrafted rules optimized for the dataset.

Grammar	Parser	# Grammar trees	
		1,000	15,000
CCG	C&C	74.1 / 83.4	82.6 / 90.1
	Model	76.8 / 85.5	84.7 / 90.9
HPSG	Enju	75.8 / 80.6	84.2 / 87.3
	Model	76.9 / 82.0	84.9 / 88.3
LFG	Pipeline Annotator	68.5 / 74.0	82.6 / 85.9
	Model	69.8 / 76.6	81.1 / 84.7

Table 4.4: The labeled/unlabeled dependency F-score comparisons between our model and state-of-the-art parsers.

Correspondence between CFG and Target Formalisms: Finally, we analyze highly weighted features. Table 4.5 shows such features for HPSG; similar patterns are also found for the other grammar formalisms. The first two features are formalism-

specific ones, the first for HPSG and the second for CFG. They show that we correctly learn a frequent derivation in the target formalism and CFG. The third one shows an example of a connection between CFG and the target formalism. Our model correctly learns that a syntactic derivation with children VP and NP is very likely to be mapped to the derivation (head_comp)→ ([N⟨V⟩N],[N.3sg]) in HPSG.

Feature type	Features with high weight
Target formalism	*Template* $(r) \rightarrow (hl_l, hp_l)(hl_r, p_r)$ *Examples* (head_comp)→ ([N⟨V⟩N],VB)([N.3sg],NN)
Coarse formalism	*Template* $(sy_p) \rightarrow (sy_l, hp_l)(sy_r, hp_r)$ *Examples* (VP)→(VP,VB)(NP,NN)
Joint features	*Template* $(r) \rightarrow (hl_l, sy_l)(le_r, sy_r)$ *Examples* (head_comp)→ ([N⟨V⟩N],VP)([N.3sg],NP)

Table 4.5: Example features with high weight.

4.8 Summary

In this chapter, we present a method for cross-formalism transfer in parsing. Our model utilizes coarse syntactic annotations to supplement a small number of formalism-specific trees for training on constituency-based grammars. Our experimental results show that across a range of such formalisms, the model significantly benefits from the coarse annotations.

Chapter 5

Conclusions and Future Work

In this thesis, we have presented two methods that enable automatic transfer from the source domain to the target domain. We assume rich annotated resources are available in the source domain, while annotations in the target domain are not available or very sparse. Our models make it possible to adapt the information in the source domain to improve the task performance in the target domain.

Our first approach automatically finds a part-of-speech mapping from language-specific tags into a universal tagset. Our key hypothesis is that a high quality mapping for the target language yield to some linguistic statistics and properties that are consistent to those of the source language. Given the universal POS representation in the source language, we encode the divergence of distributions between target and source languages and the constraints of linguistic properties into a linear combination of convex and concave functions. We further propose a variant of EM algorithm to optimize this objective. Moreover, our method does not necessarily require the language-specific POS annotation in the target language. We also evaluate our model in the scenario where language-specific POS annotation is not available. The result shows that the performance based on our mapping is closed to that based on oracle mapping.

Our second approach automatically extracts the syntactic information from the Penn Treebank to improve the parsing in target grammar formalisms, i.e. CCG, HPSG and LFG. Our key hypothesis is that the target constituency-based grammar

shares a similar syntactic structure with the coarse CFG grammar. We propose a log-linear model which provides an interface to transfer the syntactic information from source treebank while allows the parser to learn the formalism-specific information from the target treebank at the same time. We evaluate our model on three different target formalism, without much engineering work on each. This result indicates the possibility to generalize our model to other constituency-based grammar without much effort.

Our approaches focus on automating the transfer processes which are manually performed before. Often the expensive human cost of these processes and the lack of the resulting resources become the limiting factors of research in natural language processing. In the future, we would like to explore more similar scenarios and propose automatic transfer learning method to address the problems. On the other hand, it would be interesting to apply our optimization method on mapping to other learning and optimization tasks. It would also be interesting to investigate the common characteristics among dependency grammars (e.g. LTAG) and design a similar transfer model as in our work.

Appendix A

Derivation of the Mapping Optimization Algorithm

Recall that our objective is given by (we neglect the $F_{verb}(A)$ since it is convex and adding it is straightforward):

$$F(A) = D_{KL}[p_S(c_1, c_2)|p_T(c_1, c_2; A)] + \min_{r(c_1,c_2)\in\mathcal{S}} D_{KL}[r(c_1, c_2)|p_T(c_1, c_2; A)] + H[A] \tag{A.1}$$

We assume that the weights α_i and the coefficient λ are one for simplicity. Other values can easily be plugged in.

Since we are minimizing over A we can recast the problem as minimization over both A and $r \in \mathcal{S}$ of the objective:

$$F(A, r) = D_{KL}[p_S(c_1, c_2)|p_T(c_1, c_2; A)] + D_{KL}[r(c_1, c_2)|p_T(c_1, c_2; A)] + H[A] \tag{A.2}$$

We now recall the variational property of entropy, namely:[1]

$$H[p] = -\sum_x p(x) \log p(x) = \min_q -\sum_x p(x) \log q(x) \tag{A.3}$$

Where optimization is over distributions q. Thus we can again expand F to contain

[1]This is a direct result of $D_{KL}[p|q] \geq 0$ and zero if and only if $p = q$.

another variable $q(c|f)$ such that:

$$F(A,r,q) = D_{KL}[p_S(c_1,c_2)|p_T(c_1,c_2;A)] + D_{KL}[r(c_1,c_2)|p_T(c_1,c_2;A)] - \sum_{f,c} A(c|f)\log q(c|f) \tag{A.4}$$

Clearly $F(A,r,q) \geq F(A)$ for all r,q and $\min_{r,q} F(A,r,q) = F(A)$. Thus, we can proceed in alternating optimization over A,r,q.

We can now see how the algorithm in the paper is obtained. Denote by A^k, r^{k-1}, q^{k-1} the values of these variables at iteration k. Then:

$$r^k(c_1,c_2) = \arg\min_{r(c_1,c_2)\in\mathcal{S}} D_{KL}[r(c_1,c_2)|p_T(c_1,c_2;A^k)] \tag{A.5}$$

This clearly corresponds to steps 1 and 2 in the algorithm.

Next, we optimize over q, which results in:

$$q^k(c|f) = A^k(c|f) \tag{A.6}$$

for all c,f. We note turn to optimizing over A. The objective as a function of A, given the current q^k, r^k is (up to additive constants)

$$F^k(A) = -\sum_{c_1,c_2} \left[p_S(c_1,c_2) + r^k(c_1,c_2)\right] \log p_T(c_1,c_2;A) - \sum_{f,c} A(c|f)\log A^k(c|f)$$

This is non-convex due to the bilinear form of $p_T(c_1,c_2;A)$. To simplify things further we use the standard EM trick and define an auxiliary function:

$$\begin{aligned}\bar{F}^k(A) \;\equiv\; & -\sum_{c_1,c_2,f_1,f_2} p(f_1,f_2|c_1,c_2;A^k)\left[p_S(c_1,c_2) + r^k(c_1,c_2)\right]\log p_T(c_1,f_1,c_2,f_2;A) \\ & -\sum_{f,c} A(c|f)\log A^k(c|f) + g(A^k)\end{aligned}$$

where $p(f_1,f_2|c_1,c_2;A^k)$ is the posterior calculated in step 3 of the algorithm and $g(A^k)$ is a function of A^k and not A.[2] As in standard EM, it can be shown that

[2]It is given by $g(A^k) = -\sum_{c_1,c_2}\left[p_S(c_1,c_2) + r^k(c_1,c_2)\right] H[p(f_1,f_2|c_1,c_2;A^k)]$.

$F^k(A) \leq \bar{F}^k(A)$ with equality if $A = A^k$. Thus we can minimize $\bar{F}^k(A)$ over A and decrease the objective $F(A, r, q)$.

Using the notation in step 4 of the paper, this simplifies to:

$$\begin{aligned} \bar{F}^k(A) &= -\sum_{c_1,c_2,f_1,f_2} N^k(c_1, c_2, f_1, f_2) \log p_T(c_1, f_1, c_2, f_2; A) \\ &\quad -\sum_{f,c} A(c|f) \log A^k(c|f) + g(A^k) \end{aligned}$$

We can now use the fact that $p_T(c_1, f_1, c_2, f_2; A)$ factors according to:

$$p_T(c_1, f_1, c_2, f_2; A) = A(c_1|f_1)A(c_2|f_2)p_T(f_1, f_2) \tag{A.7}$$

to obtain (up to additive constants):

$$\begin{aligned} \bar{F}^k(A) &\equiv -\sum_{c,f} N_1^k(c, f) \log A(c|f) - \sum_{c,f} N_2^k(c, f) \log A(c|f) \\ &\quad -\sum_{f,c} A(c|f) \log A^k(c|f) \end{aligned}$$

And using the definition of M^k in step 5 of the paper, we obtain that:

$$\bar{F}^k(A) \equiv -\sum_{c,f} \left[M^k(c, f) \log A(c|f) + A(c|f) \log A^k(c|f) \right]$$

We now just need to minimize it over A, and this indeed corresponds to step 6 in the algorithm (except for the term $F_{verb}(A)$ which is straightforward to add).

The above establishes that the F objective decreases monotonically with each update. Convergence to local optima can be established as in EM.

Appendix B

Examples of POS Tag Mappings

Given below are a few examples of our mappings from language-specifics POS tags (first column) to universal POS tags (second column) for different languages. We also provide the manual mappings (third column) by [51] for comparison.

Arabic

Language-specific	Model	Petrov	Language-specific	Model	Petrov
N-	NOUN	NOUN	Q-	VERB	NUM
A-	ADJ	ADJ	F-	PRON	PRT
VI	VERB	VERB	P-	ADP	ADP
VP	VERB	VERB	Z-	ADV	NOUN
C-	CONJ	CONJ	SR	NUM	PRON
S-	PRON	PRON	D-	ADV	ADV
_	ADV	X	FN	PRON	PRT
SD	NOUN	PRON	FI	PRON	PRT
–	VERB	X	Y-	NUM	X
I-	VERB	X	VC	VERB	VERB

English

Language-specific	Model	Petrov	Language-specific	Model	Petrov
JJ	ADJ	ADJ	RB	ADV	ADV
TO	PRT	PRT	DT	PRON	DET
RP	ADV	PRT	RBR	ADV	ADV
RBS	ADJ	ADV	LS	ADV	X
JJS	ADJ	ADJ	FW	NOUN	X
JJR	ADJ	ADJ	NN	NOUN	NOUN
NNPS	NOUN	NOUN	VBN	ADV	VERB
VB	VERB	VERB	PDT	ADP	DET
VBP	VERB	VERB	WP$	CONJ	PRON
PRP	PRON	PRON	SYM	CONJ	X
MD	PRT	VERB	WDT	PRON	DET
VBZ	VERB	VERB	WP	PRON	PRON
IN	ADP	ADP	EX	PRON	DET
POS	CONJ	PRT	VBG	ADV	VERB
VBD	VERB	VERB	UH	ADV	X
PRP$	PRON	PRON	NNS	NOUN	NOUN
CC	CONJ	CONJ	CD	NUM	NUM
NNP	NOUN	NOUN	WRB	ADP	ADV

Greek

Language-specific	Model	Petrov	Language-specific	Model	Petrov
RgAbXx	NUM	X	RgFwTr	NUM	X
VbIs	VERB	VERB	LSPLIT	CONJ	X
NmCt	ADJ	NUM	NoCm	NOUN	NOUN
PtFu	NOUN	PRT	RgAnXx	NOUN	X
AtDf	DET	DET	VbMn	VERB	VERB
PtSj	PRON	PRT	NmCd	ADJ	NUM
COMP	ADJ	X	CjCo	CONJ	CONJ
NmOd	ADJ	NUM	Ad	ADV	ADV
CjSb	ADV	CONJ	AsPpSp	ADP	ADP
Aj	ADJ	ADJ	PnIr	ADV	PRON
PnId	ADJ	PRON	ENUM	ADJ	NUM
DIG	ADJ	NUM	PtNg	NOUN	PRT
PtOt	NOUN	PRT	PnPe	NOUN	PRON
AsPpPa	ADP	ADP	NoPr	NUM	NOUN
PnRi	ADV	PRON	PnDm	ADV	PRON
RgFwOr	NUM	X	PnRe	ADP	PRON
DATE	NUM	NUM	INIT	NUM	X
AtId	DET	DET	NmMl	ADJ	NUM
PnPo	CONJ	PRON			

Spanish

Language-specific	Model	Petrov	Language-specific	Model	Petrov
dt	DET	DET	vm	VERB	VERB
rn	PRON	ADV	np	NOUN	NOUN
p0	PRON	PRON	rg	ADV	ADV
vs	VERB	VERB	de	DET	DET
dd	DET	DET	di	DET	DET
dn	NUM	DET	Y	NUM	X
va	PRON	VERB	dp	DET	DET
sn	PRON	ADP	da	DET	DET
sp	ADP	ADP	aq	ADJ	ADJ
i	ADV	X	cs	ADV	CONJ
pd	NOUN	PRON	w	NOUN	NUM
Zm	NOUN	NUM	pe	ADP	PRON
px	NOUN	PRON	ao	NUM	ADJ
nc	NOUN	NOUN	Zp	NOUN	NUM
pt	NOUN	PRON	pn	NOUN	PRON
pp	PRON	PRON	pi	NOUN	PRON
z	NUM	NUM	cc	CONJ	CONJ
pr	PRON	PRON			

Turkish

Language-specific	Model	Petrov	Language-specific	Model	Petrov
Noun	NOUN	NOUN	DemonsP	PRON	PRON
Dup	PRON	PRT	NFutPart	NOUN	NOUN
Det	DET	DET	Adv	ADV	ADV
Verb	VERB	VERB	Zero	ADJ	VERB
Interj	PRON	X	APastPart	DET	ADJ
Ques	ADV	.	Ord	ADJ	NUM
PersP	PRON	PRON	Prop	NUM	NOUN
Pron	CONJ	PRON	Conj	CONJ	CONJ
AFutPart	DET	ADJ	Adj	ADJ	ADJ
Distrib	ADJ	NUM	NPastPart	NOUN	NOUN
Postp	ADP	ADP	Range	ADJ	NUM
Num	DET	NUM	ReflexP	PRON	PRON
APresPart	DET	ADJ	Real	ADJ	NUM
QuesP	PRON	PRON	NInf	NOUN	NOUN
Card	ADJ	NUM			

Bibliography

[1] Joan Bresnan. *The mental representation of grammatical relations*, volume 1. The MIT Press, 1982.

[2] Sabine Buchholz and Erwin Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL*, pages 149–164, 2006.

[3] David Burkett and Dan Klein. Two languages are better than one (for syntactic parsing). In *Proceedings of EMNLP*, pages 877–886, 2008.

[4] Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. The parallel grammar project. In *Proceedings of the 2002 workshop on Grammar engineering and evaluation-Volume 15*, pages 1–7. Association for Computational Linguistics, 2002.

[5] Aoife Cahill. *Parsing with Automatically Acquired, Wide-Coverage, Robust, Probabilistic LFG Approximation*. PhD thesis, 2004.

[6] Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef Van Genabith, and Andy Way. Long-distance dependency resolution in automatically acquired wide-coverage pcfg-based lfg approximations. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 319. Association for Computational Linguistics, 2004.

[7] Aoife Cahill, Mairad McCarthy, Josef van Genabith, and Andy Way. Parsing with pcfgs and automatic f-structure annotation. In *Proceedings of the Seventh International Conference on LFG*, pages 76–95. CSLI Publications, 2002.

[8] Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139, 2000.

[9] Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics, 2005.

[10] Stephen Clark and James R Curran. Log-linear models for wide-coverage ccg parsing. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 97–104. Association for Computational Linguistics, 2003.

[11] Stephen Clark and James R Curran. Wide-coverage efficient statistical parsing with ccg and log-linear models. *Computational Linguistics*, 33(4):493–552, 2007.

[12] Shay B. Cohen, Dipanjan Das, and Noah A. Smith. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of EMNLP*, pages 50–61, 2011.

[13] Michael Collins. Three generative, lexicalised models for statistical pprsing. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 16–23. Association for Computational Linguistics, 1997.

[14] Michael Collins. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637, 2003.

[15] Frank Van Eynde. Part of speech tagging en lemmatisering van het corpus gesproken nederlands. In *Technical report*, 2004.

[16] Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. The infinite tree. In *Proceedings of ACL*, pages 272–279, 2007.

[17] Kuzman Ganchev, Joao Graca, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *JMLR*, 11:2001–2049, 2010.

[18] Michael Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[19] Michael C. Grant and Stephen P. Boyd. Graph implementations for nonsmooth convex programs. In *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008.

[20] Michael C. Grant and Stephen P. Boyd. CVX: Matlab software for disciplined convex programming, version 1.21, 2011.

[21] William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI: portable parallel programming with the message passing interface*, volume 1. MIT press, 1999.

[22] Aria Haghighi and Dan Klein. Prototype-driven learning for sequence models. In *Proceedings of NAACL*, pages 320–327, 2006.

[23] Martin Haspelmath, Matthew S. Dryer, David Gil, and Bernard Comrie, editors. *The World Atlas of Language Structures*. Oxford University Press, 2005.

[24] Julia Hockenmaier. Data and models for statistical parsing with combinatory categorial grammar. 2003.

[25] Julia Hockenmaier and Mark Steedman. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third LREC Conference*, pages 1974–1981, 2002.

[26] Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. Bootstrapping parsers via syntactic projection across parallel texts. *Journal of Natural Language Engineering*, 11:311–325, 2005.

[27] Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. Estimators for stochastic unification-based grammars. In *Proceedings of the 37th*

annual meeting of the Association for Computational Linguistics on Computational Linguistics, pages 535–541. Association for Computational Linguistics, 1999.

[28] Ronald M. Kaplan, Stefan Riezler, Tracy H. King, John T. Maxwell III, Alexander Vasserman, and Richard Crouch. Speed and accuracy in shallow and deep stochastic parsing. In *Proceedings of NAACL*, 2004.

[29] Tracy Holloway King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M Kaplan. The parc 700 dependency bank. In *Proceedings of the EACL03: 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*, pages 1–8, 2003.

[30] Dan Klein and Christopher Manning. Accurate unlexicalized parsing. In *Proceedings of ACL*, pages 423–430, 2003.

[31] Terry Koo, Xavier Carreras, and Michael Collins. Simple semi-supervised dependency parsing. In *Proceedings of ACL*, pages 595–603, 2008.

[32] Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. Simple type-level unsupervised pos tagging. In *Proceedings of EMNLP*, pages 853–861, 2010.

[33] Percy Liang, Slav Petrov, Michael I. Jordan, and Dan Klein. The infinite pcfg using hierarchical dirichlet processes. In *Proceedings of EMNLP-CoNLL*, pages 688–697, 2007.

[34] 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.

[35] Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. Probabilistic cfg with latent annotations. In *Proceedings of ACL*, pages 75–82, 2005.

[36] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of EMNLP*, pages 523–530, 2005.

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

[38] Ryan McDonald, Slav Petrov, and Keith Hall. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of EMNLP*, pages 62–72, 2011.

[39] Yusuke Miyao. *From Linguistic Theory to Syntactic Analysis: Corpus-Oriented Grammar Development and Feature Forest Model*. PhD thesis, 2006.

[40] Yusuke Miyao, Takashi Ninomiya, and Junichi Tsujii. Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the penn treebank. *Natural Language Processing–IJCNLP 2004*, pages 684–693, 2005.

[41] Yusuke Miyao and Jun'ichi Tsujii. Probabilistic disambiguation models for wide-coverage hpsg parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 83–90. Association for Computational Linguistics, 2005.

[42] Yusuke Miyao and Jun'ichi Tsujii. Feature forest models for probabilistic hpsg parsing. *Computational Linguistics*, 34(1):35–80, 2008.

[43] 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.

[44] Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of EMNLP*, pages 1234–1244, 2010.

[45] Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. *NATO ASI SERIES D BEHAVIOURAL AND SOCIAL SCIENCES*, 89:355–370, 1998.

[46] Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, 2007.

[47] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer verlag, 1999.

[48] Stephan Oepen, Dan Flickinger, and Francis Bond. Towards holistic grammar engineering and testing–grafting treebank maintenance into the grammar revision cycle. In *Proceedings of the IJCNLP workshop beyond shallow analysis*. Citeseer, 2004.

[49] Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. The lingo redwoods treebank motivation and preliminary applications. In *Proceedings of the 19th international conference on Computational linguistics-Volume 2*, pages 1–5. Association for Computational Linguistics, 2002.

[50] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of ACL-COLING*, pages 433–440, 2006.

[51] Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. In *ArXiv*, April 2011.

[52] Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *Proceedings of NAACL*, pages 404–411, 2007.

[53] Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, pages 404–411, 2007.

[54] Carl Pollard and Ivan A Sag. *Head-driven phrase structure grammar*. University of Chicago Press, 1994.

[55] Stefan Riezler, Tracy H King, Ronald M Kaplan, Richard Crouch, John T Maxwell III, and Mark Johnson. Parsing the wall street journal using a lexical-functional grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 271–278. Association for Computational Linguistics, 2002.

[56] Mark Steedman. *The syntactic process*. MIT press, 2001.

[57] Kristina Toutanova, Christopher Manning, Stuart Shieber, Dan Flickinger, and Stephan Oepen. Parse disambiguation for a rich hpsg grammar. In *First Workshop on Treebanks and Linguistic Theories (TLT2002), 253-263*. Stanford InfoLab, 2002.

[58] Chenhai Xi and Rebecca Hwa. A backoff model for bootstrapping resources for non-english languages. In *Proceedings of EMNLP*, pages 851–858, 2005.

[59] Alan Yuille and Anand Rangarajan. The concave-convex procedure (cccp). In *Proceedings of Neural Computation*, volume 15, pages 915–936, 2003.

[60] Daniel Zeman and Philip Resnik. Cross-language parser adaptation between related languages. *NLP for Less Privileged Languages*, page 35, 2008.