Transfer Learning for Constituency-Based Grammars

Yuan Zhang, Regina Barzilay
Massachusetts Institute of Technology
{yuanzh, regina}@csail.mit.edu

Amir Globerson
The Hebrew University
gamir@cs.huji.ac.il

Abstract

In this paper, 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 specific for the target formalism, and a large quantity of coarse CFG annotations from the Penn Treebank. While all of the target formalisms share a similar basic syntactic structure with Penn Treebank CFG, they also encode additional constraints and semantic features. To handle this apparent discrepancy, we design a probabilistic model that jointly generates CFG and target formalism parses. The model includes features of both parses, allowing transfer between the formalisms, while preserving parsing efficiency. We evaluate our approach on three constituency-based grammars — CCG, HPSG, and LFG, augmented with the Penn Treebank-1. Our experiments show that across all three formalisms, the target parsers significantly benefit from the coarse annotations.[1]

1 Introduction

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 formalisms is intrinsic to linguistic research. However, before these grammars can be used for statistical parsing, they require annotated sentences for training. The difficulty of obtaining such annotations is a key limiting factor that inhibits the effective use of these grammars.

[1]The source code for the work is available at http://groups.csail.mit.edu/rbg/code/grammar/acl2013.

The standard solution to this bottleneck has relied 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 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 formalism. Frequently, such annotations are available in linguistic texts that introduce the formalism. For instance, a textbook on HPSG (Pollard and Sag, 1994) illustrates grammatical constructions using about 600 examples. While these examples are informative, they are not sufficient for training. To compensate for the annotation sparsity, our approach utilizes coarsely annotated data readily available in large quantities. A natural candidate for such coarse annotations is context-free grammar (CFG) from the Penn Treebank, while the target formalism can be any constituency-based grammars, such as Combinatory Categorial Grammar (CCG) (Steedman, 2001), Lexical Functional Grammar (LFG) (Bresnan, 1982) or Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994). 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, Penn Treebank annotations do not make an explicit distinction between complement and adjunct, while all the above grammars mark these roles explicitly. Moreover, even the 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 Penn Treebank represents it as a functional tag, such as NP-SBJ. Figure 1 shows derivations in the three target formalisms we consider, as well as a CFG derivation. We can see that the derivations of these formalisms share the same basic structure, while the formalism-specific information is mainly encoded in the lexical entries and node labels.

To enable effective transfer the model has to identify shared structural components 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 2). 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.[2] 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 labeled dependency F-score of 72.3%. Adding 15,000 Penn Treebank sentences during training leads to 78.5% labeled dependency F-score, 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.

[2]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.

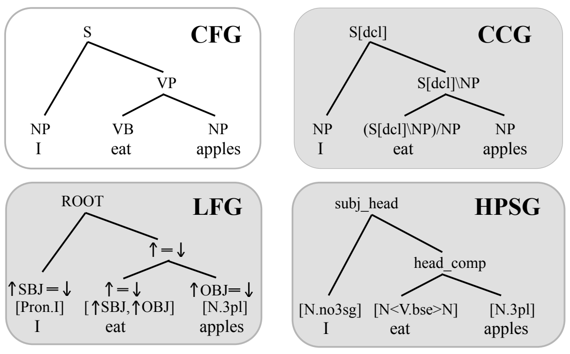


Figure 1: Derivation trees for CFG as well as CCG, HPSG and LFG formalisms.

2 Related Work

Our work belongs to a broader class of research on transfer learning in parsing. This area has garnered significant attention due to the expense associated with obtaining syntactic annotations. Transfer learning in parsing has been applied in different contexts, such as multilingual learning (Snyder et al., 2009; Hwa et al., 2005; McDonald et al., 2006; McDonald et al., 2011; Jiang and Liu, 2009), domain adaptation (McClosky et al., 2010; Dredze et al., 2007; Blitzer et al., 2006), and cross-formalism transfer (Hockenmaier and Steedman, 2002; Miyao et al., 2005; Cahill et al., 2002; Riezler et al., 2002; Chen and Shanker, 2005; Candito et al., 2010).

There have been several attempts to map annotations in coarse grammars like CFG to annotations in richer grammar, like HPSG, LFG, or CCG. Traditional approaches in this area typically rely on manually specified rules that encode the relation between the two formalisms. For instance, mappings may specify how to convert traces and functional tags in Penn Treebank to the f-structure in LFG (Cahill, 2004). 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 (Hockenmaier and Steedman, 2002; Clark and Curran, 2003; Miyao et al., 2005; Miyao and Tsujii, 2008), (2) to translate the output of a CFG parser into the target formalism (Cahill et al., 2002).

The design of these rules is a major linguistic and computational undertaking, which requires multiple iterations over the data to increase coverage (Miyao et al., 2005; Oepen et al., 2004). 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, Hockenmaier and Steedman (2002) 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 is used as an alternative to using deterministic rules.

The limitation of deterministic transfer rules has been recognized in prior work (Riezler et al., 2002). 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 the 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.

3 The Learning Problem

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 5.4.

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.

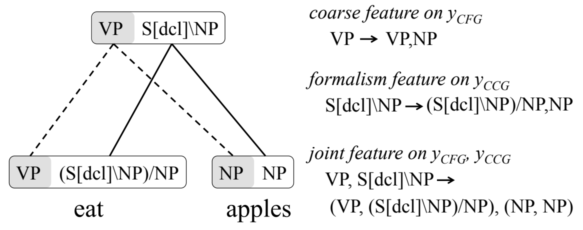

VP S[dcl]\NP
VP (S[dcl]\NP)/NP
NP NP
eat
apples
coarse feature on y_{CFG}
VP → VP,NP
formalism feature on y_{CCG}
S[dcl]\NP → (S[dcl]\NP)/NP,NP
joint feature on y_{CFG}, y_{CCG}
VP, S[dcl]\NP →
(VP, (S[dcl]\NP)/NP), (NP, NP)

Figure 2: 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 6).

4 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 5.3.

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 6. The representation and features are illustrated in Figure 2.

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:

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

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 1 to obtain distributions over y_{CCG} and y_{CFG} via marginalization. For the distribution over y_{CCG} we do precisely this, namely use:

$$p_{CCG}(y_{CCG}|S;\theta) = \sum_{y_{CFG}} p_{joint}(y|S;\theta) \qquad (2)$$

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:

$$p_{CFG}(y_{CFG}|S;\theta_{CFG}) = \frac{e^{f_{CFG}(y_{CFG},S)\cdot\theta_{CFG}}}{Z(S;\theta_{CFG})} \,. \qquad (3)$$

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 2 and 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 (Nocedal and Wright, 1999). 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.[3]

Parsing Using the Model: To parse a sentence S, we calculate the maximum probability assignment for $p_{joint}(y|S;\theta)$.[4] 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 5.

5 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.

5.1 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 (Clark and Curran, 2007). A supertagger is used for selecting a small set of lexical entry candidates for each word in the sentence. We use the tagger in (Clark and Curran, 2007) as a general suppertagger for all the grammars considered. The only difference is that we use different lexical entries in different grammars.

5.2 Feature Forest Pruning

In the BFGS algorithm (see Section 4), 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 (Miyao, 2006) to represent the exponential number of all possible tree

[3]To speed up the implementation, gradient computation is parallelized, using the Message Passing Interface package (Gropp et al., 1999).

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

structures. However, a common problem for lexicalized grammars is that the forest size is too large. In CFG, the forest is pruned according to the inside probability of a simple generative PCFG model and a prior (Collins, 2003). 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 (Clark and Curran, 2007; Miyao, 2006). 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).

5.3 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, 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 6).

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:

$$\text{NN} \rightarrow (\text{DT} \quad \text{JJ} \quad \text{NN}) \tag{4}$$

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 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%.

5.4 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.

6 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 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 (Miyao, 2006). The features are based on the atomic features listed in Table 1. These will be used to construct $f(y, S)$ as explained next.

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 1: Templates of atomic features.

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 Table 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 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 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.

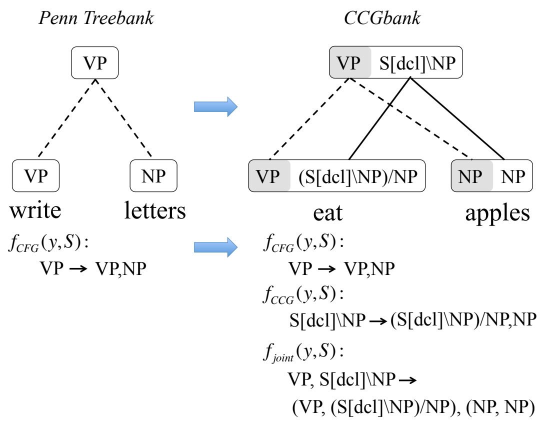


Figure 3: Example of transfer between CFG and CCG formalisms.

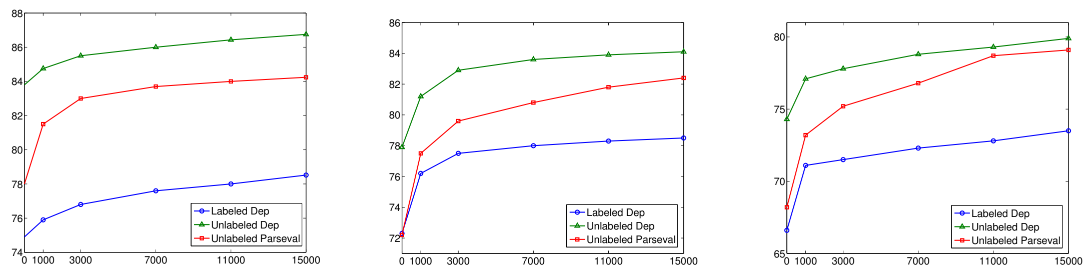
Figure 3 gives an example in CCG of how features help transfer the syntactic information from Penn Treebank and learn the correspondence to the formalism-specific information. From the Penn Treebank CFG annotations, we can learn that the derivation VP→(VP, NP) is common, as shown on the left of Figure 3. 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 three types of features work together to generate a tree skeleton and fill in the CFG and CCG labels.

7 Evaluation Setup

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

Table 3: Training/Dev./Test split on WSJ sections and PARC700 for different grammar formalisms.

Datasets: As a source of coarse annotations, we use the Penn Treebank-1 (Marcus et al., 1993). In addition, for CCG, HPSG and LFG, we rely on formalism-specific corpora developed in prior research (Hockenmaier and Steedman, 2002; Miyao et al., 2005; Cahill et al., 2002; King et al., 2003). All of these corpora were derived via conversion of Penn Treebank to the target formalisms. In particular, our CCG and HPSG datasets were converted from the Penn Treebank based on hand-



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

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

crafted rules (Hockenmaier and Steedman, 2002; Miyao et al., 2005). Table 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 (Cahill et al., 2002). However, we choose to use PARC700 as our LFG tesing and development datasets, following the previous work by (Kaplan et al., 2004). It contains 700 manually annotated sentences that are randomly selected from Penn Treebank Section 23. The split of PARC700 follows the setting in (Kaplan et al., 2004). Since our model does not assume parallel data, we use distinct sentences in the source and target treebanks. Following previous work (Hockenmaier, 2003; Miyao and Tsujii, 2008), we only consider sentences not exceeding 40 words, except on PARC700 where all sentences are used.

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 (Clark and Curran, 2003; Miyao and Tsujii, 2008; Cahill et al., 2004). For CCG, we use the evaluation script from the C&C tools.[5] 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 (Petrov and Klein, 2007; Charniak and Johnson, 2005; Charniak, 2000; Collins, 1997). 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.

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

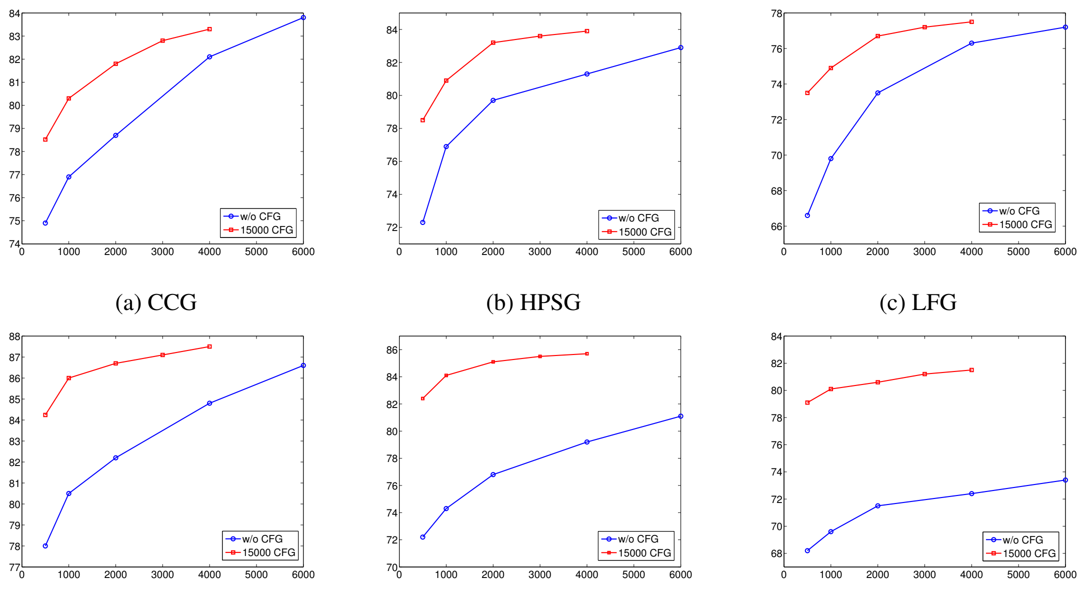
Training without CFG Data: To assess the impact of coarse data in the experiments below, we also consider the model trained only on formalism-specific annotations. When no CFG sentences are available, we assign all the CFG labels to a special value shared by all the nodes. In this set-up, the model reduces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

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 $\alpha = 0.002$.

8 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 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.



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

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

Figure 5: 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.

Tradeoff between Target and Coarse Annotations: We also assess the relative contribution of coarse annotations when the size of annotated training corpus in the 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 5 shows the relative contribution of formalism-specific annotations compared to that of the coarse annotations. For instance, Figure 5a 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 5 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 5d-f) mainly relies on the coarse structural information. On the other hand, predicate-argument dependency F-score (Figure 5a-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.

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

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 (Clark and Curran, 2003) for CCG, Enju parser (Miyao and Tsujii, 2008) for HPSG and pipeline automatic annotator (Cahill et al., 2004) 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 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 hand-crafted rules optimized for the dataset.

Correspondence between CFG and Target Formalisms: Finally, we analyze highly weighted features. Table 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 5: Example features with high weight.

9 Conclusions

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.

Acknowledgments

The authors acknowledge the support of the Army Research Office (grant 1130128-258552). We thank Yusuke Miyao, Ozlem Cetinoglu, Stephen Clark, Michael Auli and Yue Zhang for answering questions and sharing the codes of their work. We also thank the members of the MIT NLP group and the ACL reviewers for their suggestions and comments. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128. Association for Computational Linguistics.

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

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

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef Van Genabith, and Andy Way. 2004. 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.

Aoife Cahill. 2004. *Parsing with Automatically Acquired, Wide-Coverage, Robust, Probabilistic LFG Approximation*. Ph.D. thesis.

Marie Candito, Benoît Crabbé, Pascal Denis, et al. 2010. Statistical french dependency parsing: treebank conversion and first results. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010)*, pages 1840–1847.

Eugene Charniak and Mark Johnson. 2005. 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.

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

John Chen and Vijay K Shanker. 2005. Automated extraction of tags from the penn treebank. *New developments in parsing technology*, pages 73–89.

Stephen Clark and James R Curran. 2003. 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.

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

Michael Collins. 1997. 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.

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

Mark Dredze, John Blitzer, Partha Pratim Talukdar, Kuzman Ganchev, Joao V Graça, and Fernando Pereira. 2007. Frustratingly hard domain adaptation for dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, volume 2007.

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

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

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

Rebecca Hwa, Philip Resnik, and Amy Weinberg. 2005. Breaking the resource bottleneck for multilingual parsing. Technical report, DTIC Document.

Wenbin Jiang and Qun Liu. 2009. Automatic adaptation of annotation standards for dependency parsing: using projected treebank as source corpus. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 25–28. Association for Computational Linguistics.

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

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

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 28–36. Association for Computational Linguistics.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 216–220. Association for Computational Linguistics.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. 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.

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

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

Yusuke Miyao. 2006. *From Linguistic Theory to Syntactic Analysis: Corpus-Oriented Grammar Development and Feature Forest Model*. Ph.D. thesis.

Jorge Nocedal and Stephen J Wright. 1999. *Numerical optimization*. Springer verlag.

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

Slav Petrov and Dan Klein. 2007. 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.

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

Stefan Riezler, Tracy H King, Ronald M Kaplan, Richard Crouch, John T Maxwell III, and Mark Johnson. 2002. 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.

Benjamin Snyder, Tahira Naseem, and Regina Barzilay. 2009. Unsupervised multilingual grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the*

4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1, pages 73–81. Association for Computational Linguistics.

Mark Steedman. 2001. *The syntactic process*. MIT press.

Yue Zhang, Stephen Clark, et al. 2011. Shift-reduce ccg parsing. In *Proceedings of the 49th Meeting of the Association for Computational Linguistics*, pages 683–692.