Domain Adaptation in Dependency Parsing via Transformation Based Error Driven Learning

Atreyee Mukherjee, Sandra Kübler
Indiana University, Bloomington, IN
`atremukh@indiana.edu`, `skuebler@indiana.edu`

Abstract

Dependency parsers generally perform well when they are trained and tested on data from the same domain. However, if the data set on which we use the parser is different from the sentences on which it is trained, results tend to be low. Addressing this problem via domain adaptation is still a challenging problem and has achieved only limited improvements in the target domain. One problem that has been ignored to date concerns the differences in annotation schemes between corpora from different domains, even when the annotations are based on the same underlying annotation guidelines. In the most extreme case, the target annotations may contain labels that do not occur in the source domain. This significantly affects the overall performance of the parser. This paper presents an approach of applying transformation based error driven learning (TBL) for domain adaptation of dependency parsing. We use TBL to learn dependency label corrections in the target domain, based on errors made by the source domain parser. We show that this method can reduce dependency label errors significantly. A major advantage of this method is that we can address all types of errors with this method. The method can also be easily applied to any domain without any major change to the rule templates.

Keywords: Domain adaptation; dependency parsing; transformation-based error-driven learning.

1 Introduction

Dependency parsing results tend to be reliable as long as the parser is trained and tested on data from a single domain. However, the situation is considerably more challenging when the data set on which the parser is tested/used is different from the sentences on which it is trained. For example, a parser trained on newspaper corpus would not parse texts based on spontaneous dialogues accurately. Since it is impossible to manually annotate the amount of data needed to train a parser in any domain that we need to parse, we need automatic methods to adapt a parser to those new domains. This problem is generally addressed in domain adaptation. Domain adaptation attempts to use a large set of annotated data from a different domain plus specific strategies to adapt the annotations to the target domain, for which a small amount of annotated data may exist. The problem is compounded when there are differences in the annotation schemes between the source and target domain. Different treebanks, representing different domains, tend to use somewhat different annotations (Dredze et al., 2007). These differences can be due to individual syntactic interpretations of different annotators, but in many cases, they are necessitated by phenomena in the target domain that do not exist in the source domain. Spontaneous dialogues, for example, have a large number of incomplete sentences with words that cannot be attached easily if their head is missing (e.g., the determiner in "I bought the -").

Out-of-domain dependency parsing results tend to be low as compared to in-domain results. The problem can be looked at from different perspectives. Previous work in this area has extensively looked at the structural errors, where the dependency arcs are corrected. For instance, the DeSR parser (Attardi et al., 2007; Attardi and Ciaramita, 2007; Attardi et al., 2009) implements tree revisions. Structural change of dependency trees prove to be an effective domain adaptation method. Yu et al. (2015) report a 1.6% improvement by using self-training to generate training examples for the target domain. To our knowledge, no methods have been reported for addressing functional errors, i.e., errors in the dependency labels.

Our work focuses on a target domain of spontaneous dialogues, using the CReST Corpus (Eberhard et al., 2010), which uses labels that do not occur in the source domain corpus. We propose to use transformation based error driven learning (TBL) (Brill, 1995) to address the problem, with a focus on correcting dependency labels. The method has been proven effective in a variety of NLP problems including POS tagging and syntactic parsing. The idea is simple: We use a small annotated data set in the target domain and automatically annotate it using a source domain parser. Our goal is to learn a set of rules from the errors that occur and a set of rule templates. Although we focus on correcting dependency labels in the current paper, our method can be extended to correct the errors in dependency arcs as well. We demonstrate that using TBL with a small target domain training set improves dependency parsing accuracy by about 10 % absolute, and it learns to use the target-specific labels.

The paper is structured as follows: Section 2 discusses related work, section 3 describes the issues in domain adaptation in more detail, and section 4 explains our approach of using TBL for domain adaptation. In section 5, we describe our experimental setup, and section 6 discusses the results. Section 7 concludes and delineates future work.

2 Related Work

There has been a significant amount of work done on domain adaptation in dependency parsing. The primary challenge of domain adaptation is the unavailability of annotated examples from the target domain. Previous work in this area focused on analyzing how this affects parsing quality and on building systems to bypass the need for having a large amount of target domain data. Although distributional difference between the domains is a common problem, in general, Dredze et al. (2007) conclude that domain adaptation is most challenging when there are dissimilarities in annotation schemes between the domains.

Domain adaptation for dependency parsing was one of the tasks in the CoNLL 2007 Shared Task. The shared task was focused on the scenario where there is no data available from the target domain. Out of the 10 systems participating in the domain adaptation task, the highest results (81.06% LAS; 83.42% UAS) are achieved by Sagae and Tsujii (2007). To add target domain sentences to the training set, they emulate a single iteration of co-training by using MaxEnt and SVMs, selecting the sentences where both models agreed. The system by Attardi and Ciaramita (2007) (80.40% LAS; 83.08% UAS) produced similar results. They use a tree revision method (Attardi and Ciaramita, 2007) to correct structural mistakes. They formulate the problem as a supervised classification task using a multiclass perceptron, where the set of revision rules is the output space and features are based on syntactic and morphological properties of the dependency tree.

A popular approach for domain adaptation is selecting appropriate training data by using self-training or co-training for domain adaptation. Although testing on a single domain, McClosky et al. (2006a,b) show the effectiveness of using reranking and self training. They achieve an improvement of around 1% on unlabeled data. Kawahara and Uchimoto (2008) devised a method to select reliable parses from the output of a single dependency parser, MST parser (McDonald et al., 2005), instead of using an ensemble of parsers for domain adaptation. They use a self training method and combine labeled data from target domain with unlabeled data from the source by "concatenation". To estimate whether a parse is reliable, they apply binary classification using SVM on features such as sentence length, dependency length, unknown words, punctuation, average frequency of words in a sentence. They report a 1% increase in accuracy over the contemporary state of the art CoNLL shared task results on the shared task data. Yu et al. (2015) applied self-training for domain adaptation using confidence scores to select appropriate parse trees. Their highest improvement in terms of LAS is 1.6% on the CoNLL data.

Blitzer et al. (2006) used structural correspondence learning (SCL) for POS tagging and parsing to find "frequently occurring" pivot features, i.e., features that occur frequently in unlabeled data and equally characterize source and target domains. They used the WSJ as the source and MEDLINE abstracts as the target domain. They established that SCL reaches better results in both POS tagging and parsing than supervised and semi-supervised learning, even when there is no training data available on the target domain.

Transformation based error driven learning (TBL) was originally developed for POS tagging (Brill, 1992, 1995). Brill (1993) also used this approach to parse text by learning a "tranformational" grammar. The algorithm repeatedly compares the bracketed structure of the syntactic tree to the gold standard structure and learns the required transformations in the process. Brill and Resnik (1994) also use this technique for prepositional phrase attachment disambiguation. Transformation based error driven learning has also been used for information retrieval by Woodley and Geva (2005).

	Cases	# instances
1	Incorrect dependency label (incl. correct & incorrect heads)	13 839
2	Incorrect dependency head (incl. correct & incorrect label)	10 294
3	Incorrect dependency label & correct dependency head	5 389
4	Incorrect dependency label & incorrect dependency Head	8 450

Table 1: Analysis of sentences from CReST containing incorrect dependency predictions.

3 Issues in Domain Adaptation

Before we present our approach to domain adaptation, we need to better understand the different facets of the domain adaptation problem. This analysis will motivate our solution, which can handle all of those cases.

3.1 Error Types

The first phenomenon that we look at is the types of errors that can occur in dependency parsing:

1. Predicting the **head of a dependency arc** inaccurately, resulting in a structural error.
2. Predicting the **label of a dependency arc** wrong, resulting in a functional error.
3. Predicting both **label and the head of a dependency arc** wrong, resulting in structural and functional problems.

Note that structural errors affect the labeled and unlabeled attachment scores, functional errors only affect labeled attachment scores.

Previous work focused on correcting structural errors introduced by inaccurate prediction of the head of a dependency arc. To our knowledge, there are no approaches to address functional errors.

Next, we need to determine how serious these problems are. We concentrate on a setting where the source domain is the WSJ part of the Penn Treebank (Marcus et al., 1994) and the target domain is the CReST Corpus (Eberhard et al., 2010) (for more details on the corpora, see section 5). We now take a closer look at a parsing experiment where we use 17 181 WSJ sentences for training the MATE parser (Bohnet, 2010) and 5 759 CReST sentences for testing. The different types of errors are shown in table 1. This analysis shows that functional errors, i.e., errors involving dependency labels, are more frequent than structural errors. Thus, by ignoring those errors in domain adaptation, we artificially limit the possible improvements.

3.2 Error Sources

A second type of difference concerns the source of the parser errors. Here we need to distinguish two cases:

1. Parser inherent errors, i.e., errors that the parser makes independent of the out-of-domain setting.
2. Parser errors caused by differences in distribution between the two domains.
3. Differences in annotation, i.e., the domain data on which we evaluate differ from the annotations in the source data.

In the following, we focus on the differences of type 2 and 3. More specifically, we work on errors in the dependency labels since these are easier to distinguish. Structural differences tend

Treebank	No. of dep. labels	Intersection with WSJ
WSJ	67	-
CReST	31	22
Brown	71	60

Table 2: Differences in the dependency label sets across treebanks.

to be differences of degree, which are difficult to separate into distributional differences (type 2) and annotation differences (type 3). In order to establish whether annotation differences cause issues, we have analyzed a range of treebanks that are annotated using (variants of) the label set, i.e., the labels in the Penn Treebank when converted to dependencies using *pennconverter*. (Johansson and Nugues, 2007).

Table 2 looks at differences in the dependency label sets in the following treebanks: the WSJ part of the Penn Treebank, the Brown Corpus (Francis and Kucera, 1979) (Sections cg, ck, cl, cm, cn, cp, cr), and the CReST Corpus (Eberhard et al., 2010). The CoNLL style dependency trees of the WSJ and the Brown Corpus are created using the pennconverter tool (Johansson and Nugues, 2007); CReST was natively annotated in dependencies as well as constituents. It is obvious that there are considerable differences in the annotation schemes, especially between WSJ and CReST, which only have 22 labels in common. We can establish three different scenarios:

1. The source is a superset of the target data set.
2. The source is a subset of the target data set,
3. Both the source and target annotations have labels that do not occur in the other data set.

In the first case, the distribution of labels may be different between source and target, which can cause incorrect parsing decisions. The problem in the second case is more difficult since the parser is supposed to predict labels that it has not seen in training. The third case is a combination of the first two and thus the most difficult to handle.

Returning to our treebanks, we assume that since WSJ has the largest number of manually annotated sentences, it will serve as the source domain. In this case, CReST mainly shows a subset of labels from WSJ, but also has 9 labels that do not occur in WSJ.

Our hypothesis is that we can address *all* of the problems described above by using transformation based error driven learning (Brill, 1995) (for a description of this method see section 4). The method has been proven effective in many tasks such as POS tagging, syntactic parsing, and machine translation.

For the work presented here, we will focus on cases where the parser has predicted an incorrect label. This is motivated by our findings in table 1, which show that label errors are more frequent than structural ones. We investigate the scenario using CreST, where the source has a superset of labels in the target annotations, but with some unique labels in the target annotations. We hypothesize that our method can lead to an improved performance of the parser on the target domain, including dependency labels that do not exist in the source treebank. We evaluate this by the Labeled Attachment Scores (LAS) and Label Accuracy (LA).

4 Transformation-Based Error-Driven Learning for Domain Adaptation

4.1 TBL: The Brill Tagger

We implement the idea of transformation based error driven learning, which was originally developed by Brill (1992) for POS tagging. The method works with two versions of the training data: the gold standard and (an initially unannotated) working copy that simulates the learning process and records the errors. The unannotated version of the training data is tagged by a simple part of speech tagger[1], to create the initial state of the working copy of the text. The goal is to bridge the difference between the gold standard text and the working copy by learning rules that change the incorrect POS tags to the correct ones.

Learning is based on rule templates, which consist of two parts: rewrite rules and triggering environment. The templates need to be determined by the researcher, based on the problem. In general, a rule template is of the form:

Rule Template $(A, B) : X \rightarrow Y$

I.e., if we observe conditions A and B (triggering environment), we change the output variable (POS tags, for Brill and dependency labels, for us) from X to Y (rewrite rule). Note that X can remain unspecified, then the rule is applied independent of the current variable (label or POS tag).

The following is an example of a rule template for POS tagging: Change the POS tag of the current word from X to Y if the previous word is tagged as Z, where X, Y, and Z are variables that need to be instantiated during learning.

The learner creates rule hypotheses out of those templates by identifying incorrectly tagged words in the working copy and instantiating the template variables. For example, one possible rule hypothesis could be the following: Change the POS tag of the word "impact" from verb to noun if the previous word is tagged as a determiner.

In one pass through the system, all possible rule hypotheses are created and ranked based on an objective function which determines for each hypothesis how many corrections occur. The rule hypothesis with the highest score is applied to the working copy and added to the final list of transformations, stored in order, during the training process. Then the process repeats, and new rule hypotheses are generated from the templates based on the remaining errors in the working copy. The process finishes when there is no more improvement in terms of reduction of errors.

4.2 Domain Adaptation Using TBL

We use the underlying concept of transformation based error driven learning for reducing the dependency label errors resulting from parsing across domains. In particular, we address the scenario where the target domain contains a subset of the labels in the source annotation, but also contains new labels.

To use TBL for domain adaptation in dependency parsing, we need to adapt the following parts: In our case, the initial state consists of the training set from the target corpus parsed using a dependency parser, which was trained on the treebank from the source domain. We need to choose rule templates describing the relevant information that can help decide in which case to

[1]This could be any simple part of speech tagger, or a heuristic that labels every work with the most frequent POS tag

change a dependency label. We use POS and lemma[2] information rather than word forms in order to balance generality and specificity in the transformation rules. We utilize information on head of a word and its dependents since these provide essential contextual information. We currently do not use dependency labels as variables in our rule templates, but using this type of information is a viable modification in the future.

We use the following rule templates to describe the contextual properties of situations when a dependency label needs to be changed. Note that it is straightforward to include more templates, and in doing so, we can also model cases of structural change. In the latter case, the replacement part would take the form of "replace the current head of the word by word X". This will need to be accompanied by a check to ensure that the resulting dependency graph is still valid.

For this paper, we examine the following four rule templates.

- Rule template 1 (RT1)- $[(P_S, P_P, P_D, L_S, L_P, L_D) \rightarrow D_1]$

  If the part of speech tags of the word, its parent, and its dependent(s) are P_S, P_P and P_D, and the lemma of the word, its parent, and its dependent(s) are L_S, L_P, L_D respectively, the dependency label should be D_1. We consider all the dependents of a word as a single variable. E.g., $[(NN, VBZ, [DT, JJ, CC], box, be, [a, pink, and]) \rightarrow PRD]$

- Rule template 2 (RT2)- $[(P_S, P_P, L_S, L_P) \rightarrow D_1]$

  If the part of speech tags of the word and its parent are P_S and P_P, and the lemma of the word and its parent are L_S and L_P respectively, the dependency label should be D_1.

- Rule template 3 (RT3) - $[(P_S, P_P, P_D) \rightarrow D_1]$

  If the part of speech tags of the word, its parent and its dependent(s) are P_S, P_P and P_D respectively, the dependency label should be D_1.

- Rule template 4 (RT4) - $[(P_S, P_P) \rightarrow D_1]$

  If the part of speech tags of the word and its parent are P_S and P_P respectively, the dependency label should be D_1.

Figure 1 shows the TBL approach through four iterations of creating rule hypotheses from the templates, selecting and applying the best one.

We outline the process in algorithm 1. The learning iterates until there are no more rule hypotheses that would cause a further reduction in errors. During each iteration, based on the templates, the algorithm creates rule hypotheses from the errors found by comparing the working copy to the gold standard. It then chooses the best rule hypothesis that maximally reduces the errors in the working copy and applies this rule to the working copy. Since the templates that we use are independent of each other, we currently apply the top 10 rules per iteration. Choosing to apply multiple rules speeds up the overall process. To prevent over-generation, we favor the more specific hypothesis if there is a tie in the scores. We then store these rules in the final list of learned rules. After the application of the rules chosen in each iteration, we use the updated work copy in the next iteration.

After learning, a new text is processed by first having it parsed by the source domain dependency parser and then applying the learned rules in order.

[2]We derive lemma information using the TreeTagger (Schmid, 1994)

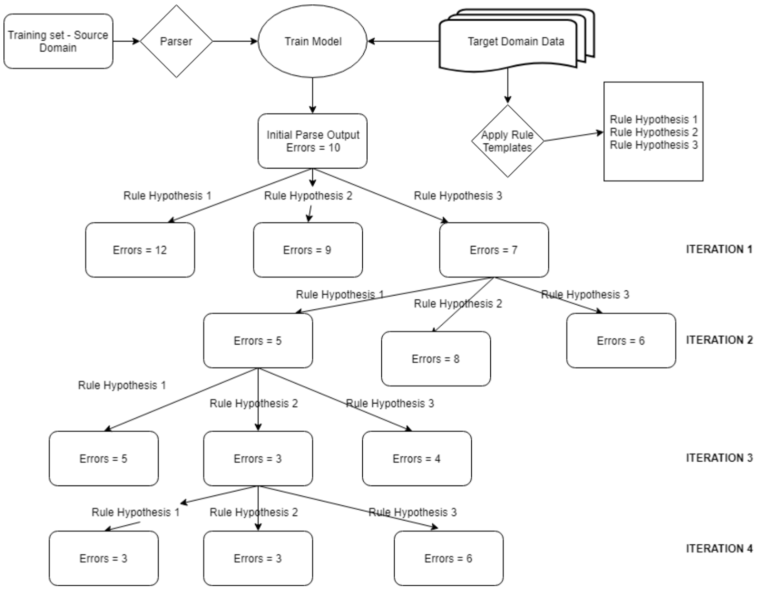

Figure 1: Reducing dependency label errors via transformation-based error-driven learning

Algorithm 1 TBL for Domain Adaptation

```
 1: procedure $TBL\_learn(training\_data)$
 2:     $work\_copy = extract\ text\ from\ training\_data;\ parse\ using\ WSJ\ model$
 3:     $learned\_rules \leftarrow [\,]$
 4:     repeat
 5:         ▷ Generate rule hypothesis from errors in corpus given rule templates
 6:         $rule\_hypotheses = generate\_hypotheses(training\_data, work\_copy)$
 7:         ▷ calculate improvement score for each rule: # errors fixed
 8:         $rule\_scores = determine\_scores(rule\_hypotheses, work\_copy)$
 9:         ▷ Rank rules given scores; select rule with highest score
10:         $best\_rule, best\_score = argmax(rank\_rules(rule\_hypotheses, rule\_scores))$
11:         ▷ If best rule causes improvement, apply rule to work copy
12:         if $best\_score > 0$ then
13:             $work\_copy = apply\_rule(work\_copy, best\_rule)$
14:             $learned\_rules += best\_rule$
15:
16:     until $best\_score <= 0$
17:     return $learned\_rules$

18: procedure $TBL\_apply(learned\_rules, test)$
19:     $test\_annotated = parse\ test\ data\ using\ WSJ\ model$
20:     ▷ Apply learned rules in order to test set
21:     for each rule in learned_rules do
22:         $test\_annotated = apply\_rule(test\_annotated, rule)$
```

WSJ	In an Oct. 19 review of " The Misanthrope " at Chicago 's Goodman Theatre (" Revitalized Classics Take the Stage in Windy City , " Leisure & Arts) , the role of Celimene , played by Kim Cattrall , was mistakenly attributed to Christina Haag .
CReST	it's like as soon - like if I were to close the door it's right next to like the bottom of the floor like where the door closes

Table 3: Sample sentences from Wall Street Journal (WSJ) & CReST corpus

	# Dialogues	# Sentences	
		with errors	without errors
Training Set	19	4384	459
Test Set	4	831	85

Table 4: Target domain training and test set for TBL

5 Experimental Setting

5.1 Data Sets

In our current work, we focus on two domains: financial news articles and dialogues in a collaborative task. We use the Wall Street Journal (Marcus et al., 1994) and the CReST corpus (Eberhard et al., 2010) as representative corpora for these two domains. For both corpora, we use teh dependency version: CReST was originally annotated in constituents and dependencies, the Penn Treebank was automatically converted to dependencies using *pennconverter* (Johansson and Nugues, 2007).

These corpora are very dissimilar in nature. On an average, the length of the sentences for WSJ is 24 and 7 for CReST. In terms of linguistic factors, CReST has a subset of WSJ's dependency labels with 10 new labels added. Example sentences from each corpora are shown in table 3. We use WSJ as the source domain and CReST as the target domain.

Target Domain The CReST corpus consists of 23 dialogues that were manually annotated for dependencies. We randomly select 19 dialogues as our training data for the TBL algorithm and the rest as test. Since the system needs to learn rule hypotheses from incorrect predictions of the source domain parser, we can safely ignore sentences that were parsed completely correctly. For the test data, we use all the sentences from the designated dialogues (with correct and incorrect dependency label predictions). Table 4 shows the division of sentences for training and test.

5.2 TBL Components

Initial State System We use the MATE parser (Bohnet, 2010) as the initial state annotator. We use 17 181 sentences from WSJ to train the MATE parser. We use this model to parse the sentences from the CReST training set to obtain the initial state annotated text.

Baseline We benchmark our results against the performance of the MATE parser trained on WSJ sentences, i.e., the out-of-domain parser, without any modification. We also report results for an in-domain setting where the MATE parser is trained on the CReST training set.

	Setting	# Incorrect Labels	LAS	LA	EM
1	Baseline (trained on WSJ)	2 112	59.02	63.73	8.41
2	In-domain baseline (trained on CReST)	572	**87.39**	90.18	**67.58**
3	Rule Templates (1+2)	600	69.91	89.70	48.03
4	Rule Templates (1+2+3+4)	451	***70.10***	**92.25**	***48.80***

Table 5: Results of applying TBL rules on test set.

Label	# in Gold	% Correct			
		baseline	in-domain	2 templates	4 templates
INTJ	690	0	97.39	98.26	99.71
ROOT*	195	0	67.35	63.59	77.44

Table 6: Accuracy in predicting CReST-specific labels.

5.3 Evaluation

For evaluation, we use the CoNLL-X script. We report the Labeled Attachment Scores (LAS) and Label Accuracy (LA) as the primary metrics for evaluating our system. LAS measures the percentage of words which have the correct head and dependency label. LA measures the number of correctly assigned labels. Reporting LA is important since this measure focuses solely on the accuracy of the labels while LAS also evaluates structural issues in the parse, which we do not address currently. We also report the exact syntactic match (EM), which measures the percentage of sentences that has correct overall predicted dependency trees.

6 Experimental Results

In this section, we discuss the outcome of our experiments for the TBL process. The results of applying the rules learned from TBL are given in table 5.

The first setting is our baseline. We evaluate the performance of our system against the results of parsing the sentences from the test set with the model trained on WSJ. The second baseline consists of an in-domain parser trained on the small CReST training set. Settings 3 and 4 evaluate the performance of the learned rules. The third setting derives rule hypotheses instantiated by 2 rule templates (rule templates 1 & 2). Since rule templates 1 & 2 are more specific (it accounts for lemma as well as part of speech tags), we add more general templates such as POS tags specifically (rule templates 3 & 4) to investigate if that leads to a significant difference in the results.

From table 5, we observe that applying TBL has a significant effect on the quality of the predicted dependency labels, with an improvement in LAS of almost 20% absolute over the out-of-domain parses. The number of completely correct sentences shows an improvement of 40% absolute over this baseline. We also see that extending the set of templates to include less-specific templates (templates 3 and 4) using POS tags information, has a minimal effect on LAS but increases label accuracy (LA) by 2.5% absolute.

When comparing the TBL results to the in-domain baseline, we notice that the baseline reaches an LAS considerably higher than the TBL results, but the TBL approach using all 4 templates reaches higher results with regard to LA (around 2% improvement). The different trends can be explained by the fact that we are currently restricting the TBL approach to functional changes, and we ignore structural errors. Since LAS evaluates both aspects, we see that the in-domain parser fares better in this respect. However, the LA, which only evaluates dependency labels,

Rule Template	Lemma (self)	Pos (self)	Lemma (head)	POS (head)	Lemma (children)	POS (children)	Deprel	Improvement
RT2	okay	UH	(root word)	(root word)	-	-	INTJ	1292
RT2	um	UH	(root word)	(root word)	-	-	INTJ	386
RT2	be	VBZ	(root word)	(root word)	-	-	ROOT	384
RT2	alright	UH	(root word)	(root word)	-	-	INTJ	216
RT2	yeah	UH	(root word)	(root word)	-	-	INTJ	205
RT2	the	DT	(root word)	(root word)	-	-	ROOT*	131
RT3	and	CC	be	VBZ	(no children)	(no children)	COORD	117
RT2	go	VBI	(root word)	(root word)	-	-	ROOT	100
RT3	that	DDT	be	VBZ	(no children)	(no children)	SBJ	83
RT2	well	UH	(root word)	(root word)	-	-	INTJ	77

Table 7: Top learned rules for the setting using 2 templates.

Rule Template	Lemma (self)	POS (self)	Lemma (head)	POS (head)	Lemma (children)	POS (children)	Deprel	Improvement
RT4	-	UH	-	(root word)	-	-	INTJ	2886
RT5	-	UH	-	(root word)	-	(no children)	INTJ	2766
RT2	okay	UH	(root word)	(root word)	-	-	INTJ	1292
RT3	okay	UH	(root word)	(root word)	(no children)	(no children)	INTJ	1271
RT4	-	AP	-	(root word)	-	-	INTJ	453
RT5	-	AP	-	(root word)	-	(no children)	INTJ	438
RT2	um	UH	(root word)	(root word)	-	-	INTJ	386
RT2	be	VBZ	(root word)	(root word)	-	-	ROOT	384
RT3	um	UH	(root word)	(root word)	(no children)	(no children)	INTJ	383
RT5		XY		(root word)		(no children)	ROOT*	268

Table 8: Top learned rules for the setting using 4 templates.

shows that our method is successful in improving the labels. Since the results of the experiment with 2 templates are similar to the in-domain treebank, we assume that we need to experiment with more specific templates in the future.

There are 9 dependency labels in CReST which do not occur in WSJ. Out of these, 2 labels (ROOT*, INTJ) occur in the test data. ROOT* is used for words whose head is missing because the sentence was interrupted; and INTJ is used for interjections. Since WSJ does not use these labels, these are predicted incorrectly in all the cases in the out-of-domain baseline.

We investigate the number of times these labels have been corrected by TBL. Table 6 shows the results. As expected, the labels do not show up in the baseline setting. For the settings using 2 or 4 templates, there is a significant improvement for the labels INTJ and ROOT*. TBL predicts these labels more accurately than the in-domain setting as well.

Tables 7 and 8 show the 10 learned rules with the highest scores, for the setting with 2 or 4 templates respectively. We can observe that many of the rules concern the labels INTJ and ROOT*. Since these labels do not appear in the source domain, applying rule hypotheses specific to these labels leads to the highest gains.

To have a closer look, we extracted the most frequent label confusions for each setting after parsing and TBL domain adaptation. The results are shown in table 9. For the baseline, the most frequently incorrect label is INTJ, followed by ROOT and ROOT*. By applying TBL, we have countered the most frequent label confusion as evident from the settings using 2 or 4 templates, where the numbers are lower across the board, but also the types of confusion are more typical of standard parsing issues (e.g., subject (SBJ) vs. predicate (PRD)).

7 Conclusion and Future Work

In this paper, we introduce transformation-based error-driven learning (TBL) for domain adaptation in dependency parsing. Since there tend to be significant differences between annotation

Baseline			2 Templates			4 Templates		
Gold	Predicted	Counts	Gold	Predicted	Counts	Gold	Predicted	Counts
INTJ	ROOT	354	ROOT*	ROOT	39	ROOT*	ROOT	38
INTJ	DEP	210	SBJ	PRD	20	SBJ	PRD	25
ROOT	NMOD	136	ROOT*	NMOD	20	ROOT	ROOT*	21
ROOT*	NMOD	100	LOC	NMOD	19	DIR	LOC	19
INTJ	NMOD	85	ROOT	ROOT*	17	DIR	ADV	17
LOC	NMOD	55	DIR	ADV	15	LOC	NMOD	15
COORD	DEP	38	ROOT	NMOD	13	TMP	ADV	10
ROOT	COORD	37	DIR	NMOD	12	LOC	ADV	10
LOC	LOC-PRD	35	LOC	ADV	11	ROOT	NMOD	8
LOC	PRD	33	PMOD	NMOD	10	OBJ	SBJ	8

Table 9: The 10 most frequent label confusions across the different settings.

schemes of different corpora from different domains, we focus on methods that can address those differences systematically along with differences due to the domain itself. When a text is parsed with a model trained on one domain, it leads to a significant number of incorrect predictions, especially for dependency labels. We address this problem in our work by using TBL to learn rules from a small target domain training set and a small set of manually defined rule templates. In comparison to a pure out-of-domain parser, we observe a significant increase in the labeled attachment scores (∼20%), labeled accuracy (∼30%) and exact match (∼40%) for WSJ as source and CReST as target corpus. The observed improvement is largely due to the labels that are used in CReST, but do not occur in WSJ since those are mislabeled consistently by the out-of-domain parser. However, when we apply TBL, these labels are corrected in most of the cases. When we compare our results to an in-domain parser, the results show that the TBL approach is very good at correcting the dependency labels, but we need to extend the approach and use more specific templates and cover structural errors as well to be able to compete with the in-domain parser with regard to LAS.

Our method is flexible in that any number of variables can be used in the templates, and it can be extended to cover structural changes, i.e., to correct dependency head annotations. In addition, the templates are valid across different target domains.

As a part of our future work, we plan to evaluate the effectiveness of this method for multiple domains. We will also introduce new variables in our rule templates including contextual information on dependency labels. In addition to correcting dependency labels, we will extend the method to correct predicted dependency heads by introducing new templates. This will need to be accompanied by a check to ensure that the resulting analysis is still a valid dependency graph.

References

Attardi, G. and Ciaramita, M. (2007). Tree revision learning for dependency parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 388–395.

Attardi, G., Dell'Orletta, F., Simi, M., Chanev, A., and Ciaramita, M. (2007). Multilingual dependency parsing and domain adaptation using DeSR. In *EMNLP-CoNLL*, pages 1112–1118.

Attardi, G., Dell'Orletta, F., Simi, M., and Turian, J. (2009). Accurate dependency parsing with a stacked multilayer perceptron. *Proceedings of EVALITA*, 9:1–8.

Blitzer, J., McDonald, R., and Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 120–128, Sydney, Australia.

Bohnet, B. (2010). Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, pages 89–97, Beijing, China.

Brill, E. (1992). A simple rule-based part of speech tagger. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 112–116, Harriman, NY.

Brill, E. (1993). Automatic grammar induction and parsing free text: A transformation-based approach. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 259–265.

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 24(1):543–565.

Brill, E. and Resnik, P. (1994). A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the 15th Conference on Computational Linguistics*, pages 1198–1204.

Dredze, M., Blitzer, J., Talukdar, P. P., Ganchev, K., Graca, J., and Pereira, F. C. (2007). Frustratingly hard domain adaptation for dependency parsing. In *EMNLP-CoNLL*, pages 1051–1055.

Eberhard, K., Nicholson, H., Kübler, S., Gunderson, S., and Scheutz, M. (2010). The Indiana "Cooperative Remote Search Task" (CReST) Corpus. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC)*, Valetta, Malta.

Francis, W. N. and Kucera, H. (1979). Brown corpus manual. Brown University.

Johansson, R. and Nugues, P. (2007). Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, pages 105–112, Tartu, Estonia.

Kawahara, D. and Uchimoto, K. (2008). Learning reliability of parses for domain adaptation of dependency parsing. In *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP)*, Hyderabad, India.

Marcus, M., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., and Schasberger, B. (1994). The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 114–119, Plainsboro, NJ.

McClosky, D., Charniak, E., and Johnson, M. (2006a). Effective self-training for parsing. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 152–159, New York, New York.

McClosky, D., Charniak, E., and Johnson, M. (2006b). Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 337–344, Sydney, Australia.

McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530.

Sagae, K. and Tsujii, J. (2007). Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1044–1050, Prague, Czech Republic.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*.

Woodley, A. and Geva, S. (2005). Applying transformation-based error-driven learning to structured natural language queries. In *International Conference on Cyberworlds*, pages 8–pp.

Yu, J., Elkaref, M., and Bohnet, B. (2015). Domain adaptation for dependency parsing via self-training. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 1–10.