Structural Correspondence Learning for Dependency Parsing

Nobuyuki Shimizu
Information Technology Center
University of Tokyo
Tokyo, Japan
shimizu@r.dl.itc.u-tokyo.ac.jp

Hiroshi Nakagawa
Information Technology Center
University of Tokyo
Tokyo, Japan
nakagawa@dl.itc.u-tokyo.ac.jp

Abstract

Following (Blitzer et al., 2006), we present an application of structural correspondence learning to non-projective dependency parsing (McDonald et al., 2005). To induce the correspondences among dependency edges from different domains, we looked at every two tokens in a sentence and examined whether or not there is a preposition, a determiner or a helping verb between them. Three binary linear classifiers were trained to predict the existence of a preposition, etc, on unlabeled data and we used singular value decomposition to induce new features. During the training, the parser was trained with these additional features in addition to these described in (McDonald et al., 2005). We discriminatively trained our parser in an on-line fashion using a variant of the voted perceptron (Collins, 2002; Collins and Roark, 2004; Crammer and Singer, 2003).

1 Introduction

We have recently seen growing popularity of dependency parsing. It is no longer rare to see dependency relations used as features, in tasks such as machine translation (Ding and Palmer, 2005) and relation extraction (Bunescu and Mooney, 2005). However, there is one factor that prevents the use of dependency parsing: sparseness of annotated corpora outside Wall Street Journal. In many situations we need to parse sentences from a target domain with no labeled data, which is a different distribution from a source domain where plentiful labeled training data is available.

In this paper, we investigate the effectiveness of structural correspondence learning (SCL) (Blitzer et al., 2006) in the domain adaptation task given by the CoNLL 2007. They hypothesize that a model trained in the source domain using this common feature representation will generalize better to the target domain, and focus on using unlabeled data from both the source and target domains to learn a common feature representation that is meaningful across both domains.

The paper is structured as follows: in section 2, we review the decoding and learning aspects of (McDonald et al., 2005), in section 3, structural correspondence learning applied to dependency parsing, and in section 4, we describe the experiments and the features needed for the CoNLL 2006 shared task.

2 Non-Projective Dependency Parsing

2.1 Dependency Structure

Let us define x to be a generic sequence of input tokens together with their POS tags and other morphological features, and y to be a generic dependency structure, that is, a set of edges for x.

A labeled edge is a tuple $\langle \mathit{DEPREL}, i \rightarrow j \rangle$ where i is the start point of the edge, j is the end point, and *DEPREL* is the label of the edge. The token at i is the head of the token at j.

Table 1 shows our formulation of a structured prediction problem. Given x, the input tokens and their features (column 2 and 3, Table 1), the task is to pre-

Index	Token	POS	Labeled Edge
1	John	NN	$\langle \mathit{SUBJ}, 2 \rightarrow 1 \rangle$
2	saw	VBD	$\langle \mathit{PRED}, 0 \rightarrow 2 \rangle$
3	a	DT	$\langle \mathit{DET}, 4 \rightarrow 3 \rangle$
4	dog	NN	$\langle \mathit{OBJ}, 2 \rightarrow 4 \rangle$
5	yesterday	RB	$\langle \mathit{ADJU}, 2 \rightarrow 5 \rangle$
6	which	WDT	$\langle \mathit{MODWH}, 7 \rightarrow 6 \rangle$
7	was	VBD	$\langle \mathit{MODPRED}, 4 \rightarrow 7 \rangle$
8	a	DT	$\langle \mathit{DET}, 10 \rightarrow 8 \rangle$
9	Yorkshire	NN	$\langle \mathit{MODN}, 10 \rightarrow 9 \rangle$
10	Terrier	NN	$\langle \mathit{OBJ}, 7 \rightarrow 10 \rangle$
11	.	.	$\langle ., 10 \rightarrow 11 \rangle$

Table 1: Example Edges

dict y, the set of labeled edges (column 4, Table 1).

In this paper we use the common method of factoring the score of the dependency structure as the sum of the scores of all the labeled edges. A dependency structure is characterized by its labeled edges, and for each labeled edge, we have features and corresponding weights. The score of a dependency structure is the sum of these weights.

For example, let us say we would like to find the score of the labeled edge $\langle \mathit{OBJ}, 2 \rightarrow 4 \rangle$. This is the edge going to the 4th token "dog" in Table 1. The features for this edge could be:

- There is an edge starting at saw, with the POS tag VBD, and the distance between the head and the child is 2. ($head = word_j, head_{POS} = pos_j, dist(i,j) = |i-j|$)
- There is an edge ending at dog, with the POS tag NN, and the distance between the head and the child is 2. ($child = word_i, child_{POS} = pos_i, dist(i,j) = |i-j|$)

In the upcoming section, we explain a decoding algorithm for the dependency structures, and later we give a method for learning the weight vector used in the decoding.

2.2 Maximum Spanning Tree Algorithm

As in (McDonald et al., 2005), we use Chu-Liu-Edmonds (CLE) algorithm (Chu and Liu, 1965; Edmonds, 1967) for decoding. CLE finds the Maximum Spanning Tree in a directed graph. The following is a summary given in (McDonald et al., 2005).

> Informally, the algorithm has each vertex in the graph greedily select the incoming edge with highest weight.

Note that the edge is coming from the parent to the child. That is, given a child node $word_j$, we are finding the parent, or the head $word_i$ such that the edge (i,j) has the highest weight among all $i, i \neq j$.

> If a tree results, then this must be the maximum spanning tree. If not, there must be a cycle. The procedure identifies a cycle and contracts it into a single vertex and recalculates edge weights going into and out of the cycle. It can be shown that a maximum spanning tree on the contracted graph is equivalent to a maximum spanning tree in the original graph (Leonidas, 2003). Hence the algorithm can recursively call itself on the new graph.

2.3 Online Learning

Again following (McDonald et al., 2005), we have used the single best MIRA (Crammer and Singer, 2003), which is a "margin aware" variant of perceptron (Collins, 2002; Collins and Roark, 2004) for structured prediction. In short, the update is executed when the decoder fails to predict the correct parse, and we compare the correct parse y^t and the incorrect parse y' suggested by the decoding algorithm. The weights of the features in y' will be lowered, and the weights of the features in y^t will be increased accordingly.

3 Domain Adaptation

Following (Blitzer et al., 2006), we present an application of structural correspondence learning (SCL) to non-projective dependency parsing (McDonald et al., 2005). SCL is a method for adapting a classifier learned in a source domain to a target domain. We assume that both domains have unlabeled data, but only the source domain has labeled training data.

SCL works as follows: 1. Define a set of pivot features on the unlabeled data from both domains. 2. Use these pivot features to learn a mapping from the original feature spaces of both domains to a shared, low-dimensional real-valued feature space. A high inner product in this new space indicates a high degree of correspondence. 3. Use both the transformed and original features from the source domain. 4. Again using both the transformed and original features, test the samples from the target domain. If we learned a good mapping, then the effectiveness of the classifier in the source domain should transfer to the target domain.

To induce the correspondences among dependency edges in the source domain and the target domain, we looked at every two tokens in a sentence and examined whether or not there is a preposition, a determiner or a helping verb between them. Although no edge is present in unlabeled data, the presence of a preposition indicates that this edge between the tokens, if existed, will not be a noun modifier (in English corpus, this label is NMOD). Thus, this induced feature should correlate with the label of an edge candidate. We postulate that the label of an edge candidate, if known, may allow the supervised learner to choose the correct edge among the edge candidates in the target domain.

In the first step, we chose the presence of a preposition, a determiner or a helping verb between tokens as pivot features. Then three binary linear classifiers were trained to predict the existence of a preposition ($prep$), determiner (det) and helping verb (hv) on unlabeled data and obtained a weight vector for each classifier.

$$classifier_{prep}(e) = sign(w_{prep}\phi(e))$$

$$classifier_{det}(e) = sign(w_{det}\phi(e))$$

$$classifier_{hv}(e) = sign(w_{hv}\phi(e))$$

The input to the above classifiers is an edge e instead of a whole sentence x. ϕ is a mapping from an edge to a feature vector. Since POS tags were not available in unlabeled data, for pivot predictors, we took the subset of the features given by an edge. The features for pivot predictors are listed in Table 2. The reminder of the features are the same as ones used in (McDonald et al., 2005).

Using each weight vector as a column, we created a weight matrix. $W = [w_{prep}|w_{det}|w_{hv}]$. And run a singular value decomposition to induce a lower dimensional feature space. $W = U\Sigma V$. We then took the transpose of the resulting unitary matrix, $U^\top$ which maps the original data to the space spanned by the principal components, and applied it to the feature vector of every potential edge. The original feature vector is $\begin{pmatrix} \mathbf{f}_{subset} \\ \mathbf{f}_{reminder} \end{pmatrix}$. We argument the feature vector with the additional feature induced by $U^\top$. The augmented feature vectors $\begin{pmatrix} \mathbf{f}_{subset} \\ \mathbf{f}_{reminder} \\ U^\top \mathbf{f}_{subset} \end{pmatrix}$ were used throughout the training and testing of the dependency parser.

4 Experiments

Our experiments were conducted on CoNLL-2007 shared task domain adaptation track (Nivre et al., 2007) using treebanks (Marcus et al., 1993; Johansson and Nugues, 2007; Kulick et al., 2004).

Given an edge $\langle DEPREL, i, j\rangle$
$head_{-1} = word_{i-1}$
$head = word_i$
$head_{+1} = word_{i+1}$
$child_{-1} = word_{j-1}$
$child = word_j$
$child_{+1} = word_{j+1}$

Table 2: Binary Features for Pivot Predictors

4.1 Dependency Relation

The CLE algorithm works on a directed graph with unlabeled edges. Since the CoNLL shared task requires the labeling of edges, as a preprocessing stage, we created a directed complete graph. Then we labeled each edge with the highest scoring dependency relation. This complete graph was given to the CLE algorithm and the edge labels were never altered in the course of finding the maximum spanning tree.

4.2 Features

The features we used for pivot predictors to classify each edge $\langle DEPREL, i, j\rangle$ are shown in Table 2. The index i is the position of the parent and j is that of the child.

$word_j$ = the word token at the position j.
pos_j = the coarse part-of-speech at j.

No other features were used beyond the combinations of the word token in Table 2.

The hardware used was an Intel CPU at 3.0 Ghz with 32 GB of memory, and the software was written in C++. While more iterations should help, due to the time constraints, we were unable to complete more training. The parser required a few days to train.

5 Results

Unfortunately, we have discovered a bug in our codes after submitting our results for the blind tests, and the reported results in (Nivre et al., 2007) were not representative of our approach. The current results (closed class) are shown in Table 3.

For the explanations of Labeled Attachment Score, Unlabeled Attachment Score and Label Accuracy, the readers are suggested to refer to the shared task introductory paper (Nivre et al., 2007). WSJ represents the application of the parser without SCL to the source domain test set, and WSJ-SCL the parser with SCL to the same test set. Similarly

Domain	LAS	UAS	Label Accuracy
WSJ	83.01% → 83.43%	86.43% → 86.81%	88.77% → 88.99%
WSJ-SCL	83.43% → 83.59%	86.87% → 86.93%	88.75% → 89.01%
Chem	74.75% → 75.18%	80.74% → 81.24%	82.34% → 82.70%
Chem-SCL	75.04% → 74.91%	81.02% → 80.82%	82.18% → 82.18%

Table 3: Labeled Attachment Score, Unlabeled Attachment Score and Label Accuracy

Chem and Chem-SCL represents the application of the parser without SCL and with SCL to the source domain test set respectively. We did batch learning by running the online algorithm 4 times. An arrow → indicates how the results after 2nd iteration changed at the end of 4th iteration. Contrary to our expectations, we seem to see SCL overfitting to the source domain WSJ in this experiment. Due to the lack of POS tags in unlabeled data, our feature set for pivot predictors uses tokens extensively unlike that for the dependency parser. Since tokens are not as abstract as POS tags, we suspect induced features may have caused overfitting.

6 Conclusion

We presented an application of structural correspondence learning to non-projective dependency parsing. Effectiveness of SCL for domain adaptation is mixed in this experiment perhaps due to the mismatch between feature sets. Future work includes use of more sophisticated features such as POS and other morphological features, possibly a joint domain adaptation of POS tagging and dependency parsing for unlabeled data as well as re-examination of pivot features.

References

J. Blitzer, R. McDonald, and F. Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of Empirical Methods in Natural Language Processing (EMNLP).*

R. Bunescu and R. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proc. of the Joint Conf. on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP).*

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. In *Science Sinica*, page 14:13961400.

M. Collins and B. Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proc. of the 42rd Annual Meeting of the ACL.*

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. of Empirical Methods in Natural Language Processing (EMNLP).*

K. Crammer and Y. Singer. 2003. Ultraconservative online algorithms for multiclass problems. In *JMLR.*

Y. Ding and M. Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proc. of the 43rd Annual Meeting of the ACL.*

J. Edmonds. 1967. Optimum branchings. In *Journal of Research of the National Bureau of Standards*, page 71B:233240.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA).*

S. Kulick, A. Bies, M. Liberman, M. Mandel, R. McDonald, M. Palmer, A. Schein, and L. Ungar. 2004. Integrated annotation for biomedical information extraction. In *Proc. of the Human Language Technology Conference and the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL).*

G. Leonidas. 2003. Arborescence optimization problems solvable by edmonds algorithm. In *Theoretical Computer Science*, page 301:427 437.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of the Joint Conf. on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP).*

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).*