An Ensemble Model that Combines Syntactic and Semantic Clustering for Discriminative Dependency Parsing

Gholamreza Haffari
Faculty of Information Technology
Monash University
Melbourne, Australia
reza@monash.edu

Marzieh Razavi and **Anoop Sarkar**
School of Computing Science
Simon Fraser University
Vancouver, Canada
{mrazavi,anoop}@cs.sfu.ca

Abstract

We combine multiple word representations based on semantic clusters extracted from the (Brown et al., 1992) algorithm and syntactic clusters obtained from the Berkeley parser (Petrov et al., 2006) in order to improve discriminative dependency parsing in the MSTParser framework (McDonald et al., 2005). We also provide an ensemble method for combining diverse cluster-based models. The two contributions together significantly improves unlabeled dependency accuracy from 90.82% to 92.13%.

1 Introduction

A simple method for using unlabeled data in discriminative dependency parsing was provided in (Koo et al., 2008) which involved clustering the labeled and unlabeled data and then each word in the dependency treebank was assigned a cluster identifier. These identifiers were used to augment the feature representation of the edge-factored or second-order features, and this extended feature set was used to discriminatively train a dependency parser.

The use of clusters leads to the question of how to integrate various types of clusters (possibly from different clustering algorithms) in discriminative dependency parsing. Clusters obtained from the (Brown et al., 1992) clustering algorithm are typically viewed as "semantic", e.g. one cluster might contain *plan, letter, request, memo,* ... while another may contain *people, customers, employees, students,* .... Another clustering view that is more "syntactic" in nature comes from the use of state-splitting in PCFGs. For instance, we could extract a syntactic cluster *loss, time, profit, earnings, performance, rating,* ...: all head words of noun phrases corresponding to cluster of direct objects of verbs like *improve*. In this paper, we obtain syntactic clusters from the Berkeley parser (Petrov et al., 2006). This paper makes two contributions: 1) We combine together multiple word representations based on semantic and syntactic clusters in order to improve discriminative dependency parsing in the MSTParser framework (McDonald et al., 2005), and 2) We provide an ensemble method for combining diverse clustering algorithms that is the discriminative parsing analog to the generative product of experts model for parsing described in (Petrov, 2010). These two contributions combined significantly improves unlabeled dependency accuracy: 90.82% to 92.13% on Sec. 23 of the Penn Treebank, and we see consistent improvements across all our test sets.

2 Dependency Parsing

A dependency tree represents the syntactic structure of a sentence with a directed graph (Figure 1), where nodes correspond to the words, and arcs indicate head-modifier pairs (Mel'čuk, 1987). Graph-based dependency parsing searches for the highest-scoring tree according to a *part*-factored scoring function. In the first-order parsing models, the parts are individual head-modifier arcs in the dependency tree (McDonald et al., 2005). In the higher-order models, the parts consist of arcs together with some context, e.g. the parent or the sister arcs (McDonald and Pereira, 2006; Carreras, 2007; Koo and Collins, 2010). With a linear scoring function, the parse for a sentence $\mathbf{s}$ is:

$$\text{PARSE}(\mathbf{s}) = \arg\max_{\mathbf{t}\in\mathcal{T}(\mathbf{s})} \sum_{r\in\mathbf{t}} \mathbf{w}\cdot\mathbf{f}(\mathbf{s},r) \quad (1)$$

where $\mathcal{T}(\mathbf{s})$ is the space of dependency trees for $\mathbf{s}$, and $\mathbf{f}(\mathbf{s},r)$ is the feature vector for the part r which is linearly combined using the model parameter $\mathbf{w}$ to give the part score. The above $\arg\max$ search for non-projective dependency parsing is accom-

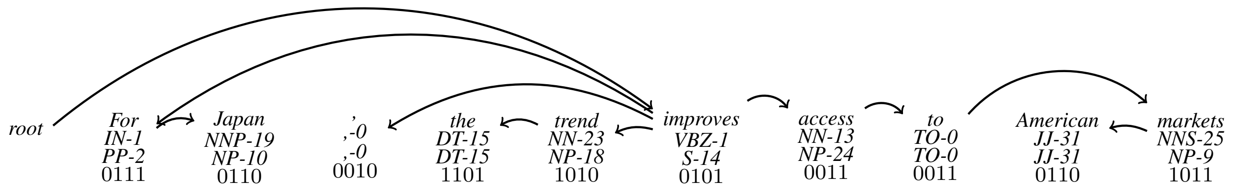

Figure 1: Dependency tree with cluster identifiers obtained from the split non-terminals from the Berkeley parser output. The first row under the words are the split POS tags (Syn-Low), the second row are the split bracketing tags (Syn-High), and the third row is the first 4 bits (to save space in this figure) of the (Brown et al., 1992) clusters.

plished using minimum spanning tree algorithms (West, 2001) or approximate inference algorithms (Smith and Eisner, 2008; Koo et al., 2010). The (Eisner, 1996) algorithm is typically used for projective parsing. The model parameters are trained using a discriminative learning algorithm, e.g. averaged perceptron (Collins, 2002) or MIRA (Crammer and Singer, 2003). In this paper, we work with both first-order and second-order models, we train the models using MIRA, and we use the (Eisner, 1996) algorithm for inference.

The baseline features capture information about the lexical items and their part of speech (POS) tags (as defined in (McDonald et al., 2005)). In this work, following (Koo et al., 2008), we use word cluster identifiers as the source of an additional set of features. The reader is directed to (Koo et al., 2008) for the list of cluster-based feature templates. The clusters inject long distance syntactic or semantic information into the model (in contrast with the use of POS tags in the baseline) and help alleviate the sparse data problem for complex features that include n-grams.

3 The Ensemble Model

A word can have different syntactic or semantic cluster representations, each of which may lead to a different parsing model. We use *ensemble learning* (Dietterich, 2002) in order to combine a collection of diverse and accurate models into a more powerful model. In this paper, we construct the base models based on different syntactic/semantic clusters used in the features in each model. Our ensemble parsing model is a linear combination of the base models:

$$\text{PARSE}(\mathbf{s}) = \arg\max_{\mathbf{t}\in\mathcal{T}(\mathbf{s})} \sum_k \alpha_k \sum_{r\in\mathbf{t}} \mathbf{w}_k \cdot \mathbf{f}_k(\mathbf{s}, r) \quad (2)$$

where α_k is the weight of the kth base model, and each base model has its own feature mapping $\mathbf{f}_k(.)$ based on its cluster annotation. Each expert parsing model in the ensemble contains all of the baseline and the cluster-based feature *templates*; therefore, the experts have in common (at least) the baseline features. The only difference between individual parsing models is the assigned cluster labels, and hence some of the cluster-based features. In a future work, we plan to take the union of all of the feature sets and train a *joint* discriminative parsing model. The ensemble approach seems more scalable though, since we can incrementally add a large number of clustering algorithms into the ensemble.

4 Syntactic and Semantic Clustering

In our ensemble model we use three different clustering methods to obtain three types of word representations that can help alleviate sparse data in a dependency parser. Our first word representation is exactly the same as the one used in (Koo et al., 2008) where words are clustered using the Brown algorithm (Brown et al., 1992). Our two other clusterings are extracted from the split non-terminals obtained from the PCFG-based Berkeley parser (Petrov et al., 2006). Split non-terminals from the Berkeley parser output are converted into cluster identifiers in two different ways: 1) the split POS tags for each word are used as an alternate word representation. We call this representation **Syn-Low**, and 2) head percolation rules are used to label each non-terminal in the parse such that each non-terminal has a unique daughter labeled as head. Each word is assigned a cluster identifier which is defined as the parent split non-terminal of that word if it is not marked as head, else if the parent is marked as head we recursively check its parent until we reach the unique split non-terminal that is not marked as head. This recursion terminates at the start symbol TOP. We call this representation **Syn-High**. We only use cluster identifiers from the Berkeley parser, rather than dependencies, or any other information.

First order features

Sec	Baseline	Brown	Syn-Low	Syn-High	Ensemble
00	89.61	90.39	90.01	89.97	**90.82**
	34.68	36.97	34.42	34.94	**37.96**
01	90.44	91.48	90.89	90.76	**91.84**
	36.36	38.62	35.66	36.56	**39.67**
23	90.02	91.13	90.46	90.35	**91.30**
	34.13	**39.64**	36.95	35.00	39.43
24	88.84	90.06	89.44	89.40	**90.33**
	30.85	**34.49**	32.49	31.22	34.05

Second order features

Sec	Baseline	Brown	Syn-Low	Syn-High	Ensemble
00	90.34	90.98	90.89	90.59	**91.41**
	38.02	**41.04**	38.80	39.16	40.93
01	91.48	92.13	91.95	91.72	**92.51**
	41.48	43.84	42.24	41.28	**45.05**
23	90.82	91.84	91.31	91.21	**92.13**
	39.18	43.66	40.84	39.97	**44.28**
24	89.87	90.61	90.28	90.31	**91.18**
	35.53	37.99	37.32	35.61	**39.55**

Table 1: For each test section and model, the number in the first/second row is the unlabeled-accuracy/unlabeled-complete-correct. See the text for more explanation.

```
(TOP
  (S-14
     (PP-2 (IN-1 For)
           (NP-10 (NNP-19 Japan)))
     (,-0 ,)
     (NP-18 (DT-15 the) (NN-23 trend))
     (VP-6 (VBZ-1 improves)
           (NP-24 (NN-13 access))
           (PP-14 (TO-0 to)
                  (NP-9 (JJ-31 American)
                        (NNS-25 markets))))))
```

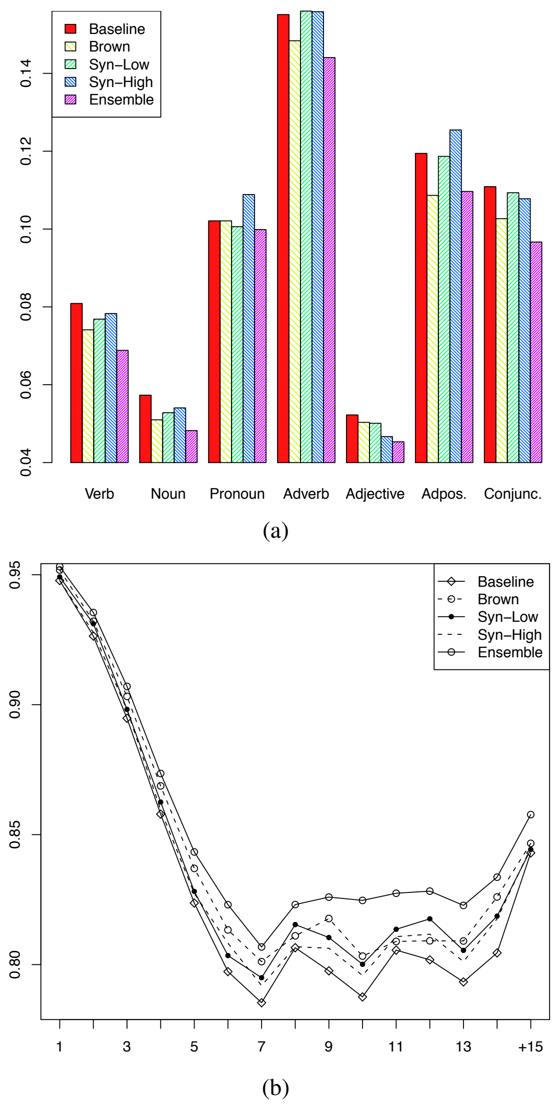

Figure 2: (a) Error rate of the head attachment for different types of modifier categories. (b) F-score for each dependency length.

For the Berkeley parser output shown above, the resulting word representations and dependency tree is shown in Fig. 1. If we group all the head-words in the training data that project up to split non-terminal NP-24 then we get a cluster: *loss, time, profit, earnings, performance, rating,* . . . which are head words of the noun phrases that appear as direct object of verbs like *improve*.

5 Experimental Results

The experiments were done on the English Penn Treebank, using standard head-percolation rules (Yamada and Matsumoto, 2003) to convert the phrase structure into dependency trees. We split the Treebank into a training set (Sections 2-21), a development set (Section 22), and test sets (Sections 0, 1, 23, and 24). All our experimental settings match previous work (Yamada and Matsumoto, 2003; McDonald et al., 2005; Koo et al., 2008). POS tags for the development and test data were assigned by MXPOST (Ratnaparkhi, 1996), where the tagger was trained on the entire training corpus. To generate part of speech tags for the training data, we used 20-way jackknifing, i.e. we tagged each fold with the tagger trained on the other 19 folds. We set model weights α_k in Eqn (2) to one for all experiments.

Syntactic State-Splitting The sentence-specific word clusters are derived from the parse trees using

Berkeley parser[1], which generates phrase-structure parse trees with split syntactic categories. To generate parse trees for development and test data, the parser is trained on the entire training data to learn a PCFG with latent annotations using split-merge operations for 5 iterations. To generate parse trees for the training data, we used 20-way jackknifing as with the tagger.

Word Clusterings from Brown Algorithm The word clusters were derived using Percy Liang's implementation of the (Brown et al., 1992) algorithm on the BLLIP corpus (Charniak et al., 2000) which contains ∼43M words of Wall Street Journal text.[2] This produces a hierarchical clustering over the words which is then sliced at a certain height to obtain the clusters. In our experiments we use the clusters obtained in (Koo et al., 2008)[3], but were unable to match the accuracy reported there, perhaps due to additional features used in their implementation not described in the paper.[4]

Results Table 1 presents our results for each model on each test set. In this table, the baseline (first column) does not use any cluster-based features, the next three models use cluster-based features using different clustering algorithms, and the last column is our ensemble model which is the linear combination of the three cluster-based models.

As Table 1 shows, the ensemble model has outperformed the baseline and individual models in almost all cases. Among the individual models, the model with Brown semantic clusters clearly outperforms the baseline, but the two models with syntactic clusters perform almost the same as the baseline. The ensemble model outperforms all of the individual models and does so very consistently across both first-order and second-order dependency models.

Error Analysis To better understand the contribution of each model to the ensemble, we take a closer look at the parsing errors for each model and the ensemble. For each dependent to head dependency, Fig. 2(a) shows the error rate for each dependent grouped by a coarse POS tag (c.f. (McDonald and Nivre, 2007)). For most POS categories, the Brown cluster model is the best individual model, but for Adjectives it is Syn-High, and for Pronouns it is Syn-Low that is the best. But the ensemble always does the best in every grammatical category. Fig. 2(b) shows the F-score of the different models for various dependency lengths, where the length of a dependency from word w_i to word w_j is equal to $|i - j|$. We see that different models are experts on different lengths (Syn-Low on 8, Syn-High on 9), while the ensemble model can always combine their expertise and do better at each length.

6 Comparison to Related Work

Several ensemble models have been proposed for dependency parsing (Sagae and Lavie, 2006; Hall et al., 2007; Nivre and McDonald, 2008; Attardi and Dell'Orletta, 2009; Surdeanu and Manning, 2010). Essentially, all of these approaches combine *different* dependency parsing systems, i.e. transition-based and graph-based. Although graph-based models are globally trained and can use exact inference algorithms, their features are defined over a limited history of parsing decisions. Since transition-based parsing models have the opposite characteristics, the idea is to combine these two types of models to exploit their complementary strengths. The base parsing models are either independently trained (Sagae and Lavie, 2006; Hall et al., 2007; Attardi and Dell'Orletta, 2009; Surdeanu and Manning, 2010), or their training is integrated, e.g. using stacking (Nivre and McDonald, 2008; Attardi and Dell'Orletta, 2009; Surdeanu and Manning, 2010).

Our work is distinguished from the aforementioned works in two dimensions. Firstly, we combine various *graph-based* models, constructed using different syntactic/semantic clusters. Secondly, we do *exact* inference on the *shared* hypothesis space of the base models. This is in contrast to previous work which combine the best parse trees suggested by the individual base-models to generate a final parse tree, i.e. a two-phase inference scheme.

7 Conclusion

We presented an ensemble of different dependency parsing models, each model corresponding to a dif-

[1]*code.google.com/p/berkeleyparser*

[2]Sentences of the Penn Treebank were excluded from the text used for the clustering.

[3]*people.csail.mit.edu/maestro/papers/bllip-clusters.gz*

[4]Terry Koo was kind enough to share the source code for the (Koo et al., 2008) paper with us, and we plan to incorporate all the features in our future work.

ferent syntactic/semantic word clustering annotation. The ensemble obtains consistent improvements in unlabeled dependency parsing, e.g. from 90.82% to 92.13% for Sec. 23 of the Penn Treebank. Our error analysis has revealed that each syntactic/semantic parsing model is an expert in capturing different dependency lengths, and the ensemble model can always combine their expertise and do better at each dependency length. We can incrementally add a large number models using different clustering algorithms, and our preliminary results show increased improvement in accuracy when more models are added into the ensemble.

Acknowledgements

This research was partially supported by NSERC, Canada (RGPIN: 264905). We would like to thank Terry Koo for his help with the cluster-based features for dependency parsing and Ryan McDonald for the MSTParser source code which we modified and used for the experiments in this paper.

References

G. Attardi and F. Dell'Orletta. 2009. Reverse revision and linear tree combination for dependency parsing. In *Proc. of NAACL-HLT*.

P. F. Brown, P. V. deSouza, R. L. Mercer, T. J. Watson, V. J. Della Pietra, and J. C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4).

X. Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proc. of EMNLP-CoNLL Shared Task*.

E. Charniak, D. Blaheta, N. Ge, K. Hall, and M. Johnson. 2000. *BLLIP 1987-89 WSJ Corpus Release 1, LDC No. LDC2000T43*, Linguistic Data Consortium.

M. Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proc. of EMNLP*.

K. Crammer and Y. Singer. 2003. Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.*, 3:951–991.

T. Dietterich. 2002. *Ensemble learning*. In The Handbook of Brain Theory and Neural Networks, Second Edition.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: an exploration. In *COLING*.

J. Hall, J. Nilsson, J. Nivre, G. Eryigit, B. Megyesi, M. Nilsson, and M. Saers. 2007. Single malt or blended? a study in multilingual parser optimization. In *Proc. of CoNLL Shared Task*.

T. Koo and M. Collins. 2010. Efficient third-order dependency parsers. In *Proc. of ACL*.

T. Koo, X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *Proc. of ACL/HLT*.

T. Koo, A. Rush, M. Collins, T. Jaakkola, and D. Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proc. of EMNLP*.

R. McDonald and J. Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proc. of EMNLP-CONLL*.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of ACL*.

I. Mel'čuk. 1987. *Dependency syntax: theory and practice*. State University of New York Press.

J. Nivre and R. McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proc. of ACL*.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. COLING-ACL*.

S. Petrov. 2010. Products of random latent variable grammars. In *Proc. of NAACL-HLT*.

A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proc. of EMNLP*.

K. Sagae and A. Lavie. 2006. Parser combination by reparsing. In *Proc. of NAACL-HLT*.

D. A. Smith and J. Eisner. 2008. Dependency parsing by belief propagation. In *Proc. of EMNLP*.

M. Surdeanu and C. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Proc. of NAACL*.

D. West. 2001. *Introduction to Graph Theory*. Prentice Hall, 2nd editoin.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of IWPT*.