# Simple Semi-supervised Dependency Parsing

**Terry Koo, Xavier Carreras, and Michael Collins**
MIT CSAIL, Cambridge, MA 02139, USA
{maestro,carreras,mcollins}@csail.mit.edu

## Abstract

We present a simple and effective semi-supervised method for training dependency parsers. We focus on the problem of lexical representation, introducing features that incorporate word clusters derived from a large unannotated corpus. We demonstrate the effectiveness of the approach in a series of dependency parsing experiments on the Penn Treebank and Prague Dependency Treebank, and we show that the cluster-based features yield substantial gains in performance across a wide range of conditions. For example, in the case of English unlabeled second-order parsing, we improve from a baseline accuracy of 92.02% to 93.16%, and in the case of Czech unlabeled second-order parsing, we improve from a baseline accuracy of 86.13% to 87.13%. In addition, we demonstrate that our method also improves performance when small amounts of training data are available, and can roughly halve the amount of supervised data required to reach a desired level of performance.

## 1 Introduction

In natural language parsing, lexical information is seen as crucial to resolving ambiguous relationships, yet lexicalized statistics are sparse and difficult to estimate directly. It is therefore attractive to consider intermediate entities which exist at a coarser level than the words themselves, yet capture the information necessary to resolve the relevant ambiguities.

In this paper, we introduce lexical intermediaries via a simple two-stage semi-supervised approach. First, we use a large unannotated corpus to define word clusters, and then we use that clustering to construct a new cluster-based feature mapping for a discriminative learner. We are thus relying on the ability of discriminative learning methods to identify and exploit informative features while remaining agnostic as to the origin of such features. To demonstrate the effectiveness of our approach, we conduct experiments in dependency parsing, which has been the focus of much recent research—e.g., see work in the CoNLL shared tasks on dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007).

The idea of combining word clusters with discriminative learning has been previously explored by Miller et al. (2004), in the context of named-entity recognition, and their work directly inspired our research. However, our target task of dependency parsing involves more complex structured relationships than named-entity tagging; moreover, it is not at all clear that word clusters should have any relevance to syntactic structure. Nevertheless, our experiments demonstrate that word clusters can be quite effective in dependency parsing applications.

In general, semi-supervised learning can be motivated by two concerns: first, given a fixed amount of supervised data, we might wish to leverage additional unlabeled data to facilitate the utilization of the supervised corpus, increasing the performance of the model in absolute terms. Second, given a fixed target performance level, we might wish to use unlabeled data to reduce the amount of annotated data necessary to reach this target.

We show that our semi-supervised approach yields improvements for fixed datasets by performing parsing experiments on the Penn Treebank (Marcus et al., 1993) and Prague Dependency Treebank (Hajič, 1998; Hajič et al., 2001) (see Sections 4.1 and 4.3). By conducting experiments on datasets of varying sizes, we demonstrate that for fixed levels of performance, the cluster-based approach can reduce the need for supervised data by roughly half, which is a substantial savings in data-annotation costs (see Sections 4.2 and 4.4).

The remainder of this paper is divided as follows:



*Proceedings of ACL-08: HLT*, pages 595–603,
Columbus, Ohio, USA, June 2008. 

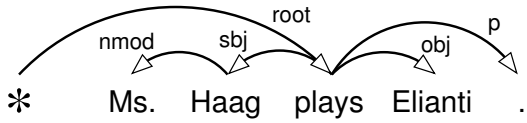


Figure 1: An example of a labeled dependency tree. The tree contains a special token "*" which is always the root of the tree. Each arc is directed from head to modifier and has a label describing the function of the attachment.

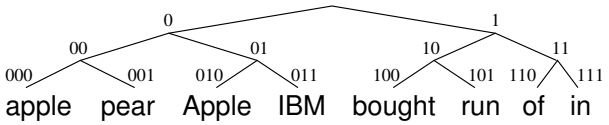


Figure 2: An example of a Brown word-cluster hierarchy. Each node in the tree is labeled with a bit-string indicating the path from the root node to that node, where 0 indicates a left branch and 1 indicates a right branch.

Section 2 gives background on dependency parsing and clustering, Section 3 describes the cluster-based features, Section 4 presents our experimental results, Section 5 discusses related work, and Section 6 concludes with ideas for future research.

## 2 Background

### 2.1 Dependency parsing

Recent work (Buchholz and Marsi, 2006; Nivre et al., 2007) has focused on dependency parsing. Dependency syntax represents syntactic information as a network of head-modifier dependency arcs, typically restricted to be a directed tree (see Figure 1 for an example). Dependency parsing depends critically on predicting head-modifier relationships, which can be difficult due to the statistical sparsity of these word-to-word interactions. Bilexical dependencies are thus ideal candidates for the application of coarse word proxies such as word clusters.

In this paper, we take a part-factored structured classification approach to dependency parsing. For a given sentence $\mathbf{x}$, let $\mathcal{Y}(\mathbf{x})$ denote the set of possible dependency structures spanning $\mathbf{x}$, where each $y \in \mathcal{Y}(\mathbf{x})$ decomposes into a set of "parts" $r \in y$. In the simplest case, these parts are the dependency arcs themselves, yielding a first-order or "edge-factored" dependency parsing model. In higher-order parsing models, the parts can consist of interactions between more than two words. For example, the parser of McDonald and Pereira (2006) defines parts for sibling interactions, such as the trio "plays", "Elianti", and "." in Figure 1. The Carreras (2007) parser has parts for both sibling interactions and grandparent interactions, such as the trio "*", "plays", and "Haag" in Figure 1. These kinds of higher-order factorizations allow dependency parsers to obtain a limited form of context-sensitivity.

Given a factorization of dependency structures into parts, we restate dependency parsing as the following maximization:

$$\textsc{Parse}(\mathbf{x}; \mathbf{w}) = \operatorname*{argmax}_{y \in \mathcal{Y}(\mathbf{x})} \sum_{r \in y} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, r)$$

Above, we have assumed that each part is scored by a linear model with parameters $\mathbf{w}$ and feature-mapping $\mathbf{f}(\cdot)$. For many different part factorizations and structure domains $\mathcal{Y}(\cdot)$, it is possible to solve the above maximization efficiently, and several recent efforts have concentrated on designing new maximization algorithms with increased context-sensitivity (Eisner, 2000; McDonald et al., 2005b; McDonald and Pereira, 2006; Carreras, 2007).

### 2.2 Brown clustering algorithm

In order to provide word clusters for our experiments, we used the Brown clustering algorithm (Brown et al., 1992). We chose to work with the Brown algorithm due to its simplicity and prior success in other NLP applications (Miller et al., 2004; Liang, 2005). However, we expect that our approach can function with other clustering algorithms (as in, e.g., Li and McCallum (2005)). We briefly describe the Brown algorithm below.

The input to the algorithm is a vocabulary of words to be clustered and a corpus of text containing these words. Initially, each word in the vocabulary is considered to be in its own distinct cluster. The algorithm then repeatedly merges the pair of clusters which causes the smallest decrease in the likelihood of the text corpus, according to a class-based bigram language model defined on the word clusters. By tracing the pairwise merge operations, one obtains a hierarchical clustering of the words, which can be represented as a binary tree as in Figure 2.

Within this tree, each word is uniquely identified by its path from the root, and this path can be compactly represented with a bit string, as in Figure 2. In order to obtain a clustering of the words, we select all nodes at a certain depth from the root of the

hierarchy. For example, in Figure 2 we might select the four nodes at depth 2 from the root, yielding the clusters {apple,pear}, {Apple,IBM}, {bought,run}, and {of,in}. Note that the same clustering can be obtained by truncating each word's bit-string to a 2-bit prefix. By using prefixes of various lengths, we can produce clusterings of different granularities (Miller et al., 2004).

For all of the experiments in this paper, we used the Liang (2005) implementation of the Brown algorithm to obtain the necessary word clusters.

## 3 Feature design

Key to the success of our approach is the use of features which allow word-cluster-based information to assist the parser. The feature sets we used are similar to other feature sets in the literature (McDonald et al., 2005a; Carreras, 2007), so we will not attempt to give a exhaustive description of the features in this section. Rather, we describe our features at a high level and concentrate on our methodology and motivations. In our experiments, we employed two different feature sets: a baseline feature set which draws upon "normal" information sources such as word forms and parts of speech, and a cluster-based feature set that also uses information derived from the Brown cluster hierarchy.

### 3.1 Baseline features

Our first-order baseline feature set is similar to the feature set of McDonald et al. (2005a), and consists of indicator functions for combinations of words and parts of speech for the head and modifier of each dependency, as well as certain contextual tokens.[1] Our second-order baseline features are the same as those of Carreras (2007) and include indicators for triples of part of speech tags for sibling interactions and grandparent interactions, as well as additional bigram features based on pairs of words involved these higher-order interactions. Examples of baseline features are provided in Table 1.

[1]We augment the McDonald et al. (2005a) feature set with backed-off versions of the "Surrounding Word POS Features" that include only one neighboring POS tag. We also add binned distance features which indicate whether the number of tokens between the head and modifier of a dependency is greater than 2, 5, 10, 20, 30, or 40 tokens.

| Baseline | Cluster-based |
|---|---|
| ht,mt | hc4,mc4 |
| hw,mw | hc6,mc6 |
| hw,ht,mt | hc⋆,mc⋆ |
| hw,ht,mw | hc4,mt |
| ht,mw,mt | ht,mc4 |
| hw,mw,mt | hc6,mt |
| hw,ht,mw,mt | ht,mc6 |
| ... | hc4,mw |
| | hw,mc4 |
| | ... |
| ht,mt,st | hc4,mc4,sc4 |
| ht,mt,gt | hc6,mc6,sc6 |
| ... | ht,mc4,sc4 |
| | hc4,mc4,gc4 |
| | ... |

Table 1: Examples of baseline and cluster-based feature templates. Each entry represents a class of indicators for tuples of information. For example, "ht,mt" represents a class of indicator features with one feature for each possible combination of head POS-tag and modifier POS-tag. Abbreviations: ht = head POS, hw = head word, hc4 = 4-bit prefix of head, hc6 = 6-bit prefix of head, hc⋆ = full bit string of head; mt,mw,mc4,mc6,mc⋆ = likewise for modifier; st,gt,sc4,gc4,... = likewise for sibling and grandchild.

### 3.2 Cluster-based features

The first- and second-order cluster-based feature sets are supersets of the baseline feature sets: they include all of the baseline feature templates, and add an additional layer of features that incorporate word clusters. Following Miller et al. (2004), we use prefixes of the Brown cluster hierarchy to produce clusterings of varying granularity. We found that it was nontrivial to select the proper prefix lengths for the dependency parsing task; in particular, the prefix lengths used in the Miller et al. (2004) work (between 12 and 20 bits) performed poorly in dependency parsing.[2] After experimenting with many different feature configurations, we eventually settled on a simple but effective methodology.

First, we found that it was helpful to employ two different types of word clusters:

1. Short bit-string prefixes (e.g., 4–6 bits), which we used as replacements for parts of speech.

[2]One possible explanation is that the kinds of distinctions required in a named-entity recognition task (e.g., "Alice" versus "Intel") are much finer-grained than the kinds of distinctions relevant to syntax (e.g., "apple" versus "eat").

2. Full bit strings,[3] which we used as substitutes for word forms.

Using these two types of clusters, we generated new features by mimicking the template structure of the original baseline features. For example, the baseline feature set includes indicators for word-to-word and tag-to-tag interactions between the head and modifier of a dependency. In the cluster-based feature set, we correspondingly introduce new indicators for interactions between pairs of short bit-string prefixes and pairs of full bit strings. Some examples of cluster-based features are given in Table 1.

Second, we found it useful to concentrate on "hybrid" features involving, e.g., one bit-string and one part of speech. In our initial attempts, we focused on features that used cluster information exclusively. While these cluster-only features provided some benefit, we found that adding hybrid features resulted in even greater improvements. One possible explanation is that the clusterings generated by the Brown algorithm can be noisy or only weakly relevant to syntax; thus, the clusters are best exploited when "anchored" to words or parts of speech.

Finally, we found it useful to impose a form of vocabulary restriction on the cluster-based features. Specifically, for any feature that is predicated on a word form, we eliminate this feature if the word in question is *not* one of the top-$N$ most frequent words in the corpus. When $N$ is between roughly 100 and 1,000, there is little effect on the performance of the cluster-based feature sets.[4] In addition, the vocabulary restriction reduces the size of the feature sets to managable proportions.

## 4 Experiments

In order to evaluate the effectiveness of the cluster-based feature sets, we conducted dependency parsing experiments in English and Czech. We test the features in a wide range of parsing configurations, including first-order and second-order parsers, and labeled and unlabeled parsers.[5]

The English experiments were performed on the Penn Treebank (Marcus et al., 1993), using a standard set of head-selection rules (Yamada and Matsumoto, 2003) to convert the phrase structure syntax of the Treebank to a dependency tree representation.[6] We split the Treebank into a training set (Sections 2–21), a development set (Section 22), and several test sets (Sections 0,[7] 1, 23, and 24). The data partition and head rules were chosen to match previous work (Yamada and Matsumoto, 2003; McDonald et al., 2005a; McDonald and Pereira, 2006). The part of speech tags for the development and test data were automatically 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 10-way jackknifing.[8] English word clusters were derived from the BLLIP corpus (Charniak et al., 2000), which contains roughly 43 million words of Wall Street Journal text.[9]

The Czech experiments were performed on the Prague Dependency Treebank 1.0 (Hajič, 1998; Hajič et al., 2001), which is directly annotated with dependency structures. To facilitate comparisons with previous work (McDonald et al., 2005b; McDonald and Pereira, 2006), we used the training/development/test partition defined in the corpus and we also used the automatically-assigned part of speech tags provided in the corpus.[10] Czech word clusters were derived from the raw text section of the PDT 1.0, which contains about 39 million words of newswire text.[11]

We trained the parsers using the averaged perceptron (Freund and Schapire, 1999; Collins, 2002), which represents a balance between strong performance and fast training times. To select the number

[3] As in Brown et al. (1992), we limit the clustering algorithm so that it recovers at most 1,000 distinct bit-strings; thus full bit strings are not equivalent to word forms.

[4] We used $N = 800$ for all experiments in this paper.

[5] In an "unlabeled" parser, we simply ignore dependency label information, which is a common simplification.

[6] We used Joakim Nivre's "Penn2Malt" conversion tool (http://w3.msi.vxu.se/ nivre/research/Penn2Malt.html). Dependency labels were obtained via the "Malt" hard-coded setting.

[7] For computational reasons, we removed a single 249-word sentence from Section 0.

[8] That is, we tagged each fold with the tagger trained on the other 9 folds.

[9] We ensured that the sentences of the Penn Treebank were excluded from the text used for the clustering.

[10] Following Collins et al. (1999), we used a coarsened version of the Czech part of speech tags; this choice also matches the conditions of previous work (McDonald et al., 2005b; McDonald and Pereira, 2006).

[11] This text was disjoint from the training and test corpora.

| Sec | dep1 | dep1c | MD1 | dep2 | dep2c | MD2 | dep1-L | dep1c-L | dep2-L | dep2c-L |
|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 90.48 | 91.57 (+1.09) | — | 91.76 | 92.77 (+1.01) | — | 90.29 | 91.03 (+0.74) | 91.33 | 92.09 (+0.76) |
| 01 | 91.31 | 92.43 (+1.12) | — | 92.46 | 93.34 (+0.88) | — | 90.84 | 91.73 (+0.89) | 91.94 | 92.65 (+0.71) |
| 23 | 90.84 | 92.23 (+1.39) | 90.9 | 92.02 | 93.16 (+1.14) | 91.5 | 90.32 | 91.24 (+0.92) | 91.38 | 92.14 (+0.76) |
| 24 | 89.67 | 91.30 (+1.63) | — | 90.92 | 91.85 (+0.93) | — | 89.55 | 90.06 (+0.51) | 90.42 | 91.18 (+0.76) |

Table 2: Parent-prediction accuracies on Sections 0, 1, 23, and 24. Abbreviations: dep1/dep1c = first-order parser with baseline/cluster-based features; dep2/dep2c = second-order parser with baseline/cluster-based features; MD1 = McDonald et al. (2005a); MD2 = McDonald and Pereira (2006); suffix -L = labeled parser. Unlabeled parsers are scored using unlabeled parent predictions, and labeled parsers are scored using labeled parent predictions. Improvements of cluster-based features over baseline features are shown in parentheses.

of iterations of perceptron training, we performed up to 30 iterations and chose the iteration which optimized accuracy on the development set. Our feature mappings are quite high-dimensional, so we eliminated all features which occur only once in the training data. The resulting models still had very high dimensionality, ranging from tens of millions to as many as a billion features.[12]

All results presented in this section are given in terms of parent-prediction accuracy, which measures the percentage of tokens that are attached to the correct head token. For labeled dependency structures, both the head token and dependency label must be correctly predicted. In addition, in English parsing we ignore the parent-predictions of punctuation tokens,[13] and in Czech parsing we retain the punctuation tokens; this matches previous work (Yamada and Matsumoto, 2003; McDonald et al., 2005a; McDonald and Pereira, 2006).

### 4.1 English main results

In our English experiments, we tested eight different parsing configurations, representing all possible choices between baseline or cluster-based feature sets, first-order (Eisner, 2000) or second-order (Carreras, 2007) factorizations, and labeled or unlabeled parsing.

Table 2 compiles our final test results and also includes two results from previous work by McDonald et al. (2005a) and McDonald and Pereira (2006), for the purposes of comparison. We note a few small differences between our parsers and the parsers evaluated in this previous work. First, the MD1 and MD2 parsers were trained via the MIRA algorithm (Crammer and Singer, 2003; Crammer et al., 2004), while we use the averaged perceptron. In addition, the MD2 model uses only sibling interactions, whereas the dep2/dep2c parsers include both sibling and grandparent interactions.

There are some clear trends in the results of Table 2. First, performance increases with the order of the parser: edge-factored models (dep1 and MD1) have the lowest performance, adding sibling relationships (MD2) increases performance, and adding grandparent relationships (dep2) yields even better accuracies. Similar observations regarding the effect of model order have also been made by Carreras (2007).

Second, note that the parsers using cluster-based feature sets consistently outperform the models using the baseline features, regardless of model order or label usage. Some of these improvements can be quite large; for example, a first-order model using cluster-based features generally performs as well as a second-order model using baseline features. Moreover, the benefits of cluster-based feature sets combine additively with the gains of increasing model order. For example, consider the unlabeled parsers in Table 2: on Section 23, increasing the model order from dep1 to dep2 results in a relative reduction in error of roughly $13\%$, while introducing cluster-based features from dep2 to dep2c yields an additional relative error reduction of roughly $14\%$. As a final note, all 16 comparisons between cluster-based features and baseline features shown in Table 2 are statistically significant.[14]

[12] Due to the sparsity of the perceptron updates, however, only a small fraction of the possible features were active in our trained models.

[13] A punctuation token is any token whose gold-standard part of speech tag is one of {`` '' : , .}.

[14] We used the sign test at the sentence level. The comparison between dep1-L and dep1c-L is significant at $p < 0.05$, and all other comparisons are significant at $p < 0.0005$.

Tagger always trained on full Treebank

| Size | dep1 | dep1c | Δ | dep2 | dep2c | Δ |
|---|---|---|---|---|---|---|
| 1k | 84.54 | 85.90 | 1.36 | 86.29 | 87.47 | 1.18 |
| 2k | 86.20 | 87.65 | 1.45 | 87.67 | 88.88 | 1.21 |
| 4k | 87.79 | 89.15 | 1.36 | 89.22 | 90.46 | 1.24 |
| 8k | 88.92 | 90.22 | 1.30 | 90.62 | 91.55 | 0.93 |
| 16k | 90.00 | 91.27 | 1.27 | 91.27 | 92.39 | 1.12 |
| 32k | 90.74 | 92.18 | 1.44 | 92.05 | 93.36 | 1.31 |
| All | 90.89 | 92.33 | 1.44 | 92.42 | 93.30 | 0.88 |

Tagger trained on reduced dataset

| Size | dep1 | dep1c | Δ | dep2 | dep2c | Δ |
|---|---|---|---|---|---|---|
| 1k | 80.49 | 84.06 | 3.57 | 81.95 | 85.33 | 3.38 |
| 2k | 83.47 | 86.04 | 2.57 | 85.02 | 87.54 | 2.52 |
| 4k | 86.53 | 88.39 | 1.86 | 87.88 | 89.67 | 1.79 |
| 8k | 88.25 | 89.94 | 1.69 | 89.71 | 91.37 | 1.66 |
| 16k | 89.66 | 91.03 | 1.37 | 91.14 | 92.22 | 1.08 |
| 32k | 90.78 | 92.12 | 1.34 | 92.09 | 93.21 | 1.12 |
| All | 90.89 | 92.33 | 1.44 | 92.42 | 93.30 | 0.88 |

Table 3: Parent-prediction accuracies of unlabeled English parsers on Section 22. Abbreviations: Size = #sentences in training corpus; Δ = difference between cluster-based and baseline features; other abbreviations are as in Table 2.

### 4.2 English learning curves

We performed additional experiments to evaluate the effect of the cluster-based features as the amount of training data is varied. Note that the dependency parsers we use require the input to be tagged with parts of speech; thus the quality of the part-of-speech tagger can have a strong effect on the performance of the parser. In these experiments, we consider two possible scenarios:

1. The tagger has a large training corpus, while the parser has a smaller training corpus. This scenario can arise when tagged data is cheaper to obtain than syntactically-annotated data.

2. The same amount of labeled data is available for training both tagger and parser.

Table 3 displays the accuracy of first- and second-order models when trained on smaller portions of the Treebank, in both scenarios described above. Note that the cluster-based features obtain consistent gains regardless of the size of the training set. When the tagger is trained on the reduced-size datasets, the gains of cluster-based features are more pronounced, but substantial improvements are obtained even when the tagger is accurate.

It is interesting to consider the amount by which cluster-based features reduce the need for supervised data, given a desired level of accuracy. Based on Table 3, we can extrapolate that cluster-based features reduce the need for supervised data by roughly a factor of 2. For example, the performance of the dep1c and dep2c models trained on 1k sentences is roughly the same as the performance of the dep1 and dep2 models, respectively, trained on 2k sentences. This approximate data-halving effect can be observed throughout the results in Table 3.

When combining the effects of model order and cluster-based features, the reductions in the amount of supervised data required are even larger. For example, in scenario 1 the dep2c model trained on 1k sentences is close in performance to the dep1 model trained on 4k sentences, and the dep2c model trained on 4k sentences is close to the dep1 model trained on the entire training set (roughly 40k sentences).

### 4.3 Czech main results

In our Czech experiments, we considered only unlabeled parsing,[15] leaving four different parsing configurations: baseline or cluster-based features and first-order or second-order parsing. Note that our feature sets were originally tuned for English parsing, and except for the use of Czech clusters, we made no attempt to retune our features for Czech.

Czech dependency structures may contain non-projective edges, so we employ a maximum directed spanning tree algorithm (Chu and Liu, 1965; Edmonds, 1967; McDonald et al., 2005b) as our first-order parser for Czech. For the second-order parsing experiments, we used the Carreras (2007) parser. Since this parser only considers projective dependency structures, we "projectivized" the PDT 1.0 training set by finding, for each sentence, the projective tree which retains the most correct dependencies; our second-order parsers were then trained with respect to these projective trees. The development and test sets were *not* projectivized, so our second-order parser is guaranteed to make errors in test sentences containing non-projective dependencies. To overcome this, McDonald and Pereira (2006) use a

[15] We leave labeled parsing experiments to future work.

| dep1 | dep1c | dep2 | dep2c |
|---|---|---|---|
| 84.49 | 86.07 (+1.58) | 86.13 | 87.13 (+1.00) |

Table 4: Parent-prediction accuracies of unlabeled Czech parsers on the PDT 1.0 test set, for baseline features and cluster-based features. Abbreviations are as in Table 2.

| Parser | Accuracy |
|---|---|
| Nivre and Nilsson (2005) | 80.1 |
| McDonald et al. (2005b) | 84.4 |
| Hall and Novák (2005) | 85.1 |
| McDonald and Pereira (2006) | 85.2 |
| dep1c | 86.07 |
| dep2c | 87.13 |

Table 5: Unlabeled parent-prediction accuracies of Czech parsers on the PDT 1.0 test set, for our models and for previous work.

| Size | dep1 | dep1c | Δ | dep2 | dep2c | Δ |
|---|---|---|---|---|---|---|
| 1k | 72.79 | 73.66 | 0.87 | 74.35 | 74.63 | 0.28 |
| 2k | 74.92 | 76.23 | 1.31 | 76.63 | 77.60 | 0.97 |
| 4k | 76.87 | 78.14 | 1.27 | 78.34 | 79.34 | 1.00 |
| 8k | 78.17 | 79.83 | 1.66 | 79.82 | 80.98 | 1.16 |
| 16k | 80.60 | 82.44 | 1.84 | 82.53 | 83.69 | 1.16 |
| 32k | 82.85 | 84.65 | 1.80 | 84.66 | 85.81 | 1.15 |
| 64k | 84.20 | 85.98 | 1.78 | 86.01 | 87.11 | 1.10 |
| All | 84.36 | 86.09 | 1.73 | 86.09 | 87.26 | 1.17 |

Table 6: Parent-prediction accuracies of unlabeled Czech parsers on the PDT 1.0 development set. Abbreviations are as in Table 3.

two-stage approximate decoding process in which the output of their second-order parser is "deprojectivized" via greedy search. For simplicity, we did not implement a deprojectivization stage on top of our second-order parser, but we conjecture that such techniques may yield some additional performance gains; we leave this to future work.

Table 4 gives accuracy results on the PDT 1.0 test set for our unlabeled parsers. As in the English experiments, there are clear trends in the results: parsers using cluster-based features outperform parsers using baseline features, and second-order parsers outperform first-order parsers. Both of the comparisons between cluster-based and baseline features in Table 4 are statistically significant.[16] Table 5 compares accuracy results on the PDT 1.0 test set for our parsers and several other recent papers.

[16]We used the sign test at the sentence level; both comparisons are significant at $p < 0.0005$.

| $N$ | dep1 | dep1c | dep2 | dep2c |
|---|---|---|---|---|
| 100 | 89.19 | 92.25 | 90.61 | 93.14 |
| 200 | 90.03 | 92.26 | 91.35 | 93.18 |
| 400 | 90.31 | 92.32 | 91.72 | 93.20 |
| 800 | 90.62 | 92.33 | 91.89 | 93.30 |
| 1600 | 90.87 | — | 92.20 | — |
| All | 90.89 | — | 92.42 | — |

Table 7: Parent-prediction accuracies of unlabeled English parsers on Section 22. Abbreviations: $N$ = threshold value; other abbreviations are as in Table 2. We did not train cluster-based parsers using threshold values larger than 800 due to computational limitations.

| dep1-P | dep1c-P | dep1 | dep2-P | dep2c-P | dep2 |
|---|---|---|---|---|---|
| 77.19 | 90.69 | 90.89 | 86.73 | 91.84 | 92.42 |

Table 8: Parent-prediction accuracies of unlabeled English parsers on Section 22. Abbreviations: suffix -P = model without POS; other abbreviations are as in Table 2.

### 4.4 Czech learning curves

As in our English experiments, we performed additional experiments on reduced sections of the PDT; the results are shown in Table 6. For simplicity, we did not retrain a tagger for each reduced dataset, so we always use the (automatically-assigned) part of speech tags provided in the corpus. Note that the cluster-based features obtain improvements at all training set sizes, with data-reduction factors similar to those observed in English. For example, the dep1c model trained on 4k sentences is roughly as good as the dep1 model trained on 8k sentences.

### 4.5 Additional results

Here, we present two additional results which further explore the behavior of the cluster-based feature sets. In Table 7, we show the development-set performance of second-order parsers as the threshold for lexical feature elimination (see Section 3.2) is varied. Note that the performance of cluster-based features is fairly insensitive to the threshold value, whereas the performance of baseline features clearly degrades as the vocabulary size is reduced.

In Table 8, we show the development-set performance of the first- and second-order parsers when features containing part-of-speech-based information are eliminated. Note that the performance obtained by using clusters *without* parts of speech is close to the performance of the baseline features.

## 5 Related Work

As mentioned earlier, our approach was inspired by the success of Miller et al. (2004), who demonstrated the effectiveness of using word clusters as features in a discriminative learning approach. Our research, however, applies this technique to dependency parsing rather than named-entity recognition.

In this paper, we have focused on developing new representations for lexical information. Previous research in this area includes several models which incorporate hidden variables (Matsuzaki et al., 2005; Koo and Collins, 2005; Petrov et al., 2006; Titov and Henderson, 2007). These approaches have the advantage that the model is able to learn different usages for the hidden variables, depending on the target problem at hand. Crucially, however, these methods do not exploit unlabeled data when learning their representations.

Wang et al. (2005) used distributional similarity scores to smooth a generative probability model for dependency parsing and obtained improvements in a Chinese parsing task. Our approach is similar to theirs in that the Brown algorithm produces clusters based on distributional similarity, and the cluster-based features can be viewed as being a kind of "backed-off" version of the baseline features. However, our work is focused on discriminative learning as opposed to generative models.

Semi-supervised phrase structure parsing has been previously explored by McClosky et al. (2006), who applied a reranked parser to a large unsupervised corpus in order to obtain additional training data for the parser; this self-training appraoch was shown to be quite effective in practice. However, their approach depends on the usage of a high-quality parse reranker, whereas the method described here simply augments the features of an existing parser. Note that our two approaches are compatible in that we could also design a reranker and apply self-training techniques on top of the cluster-based features.

## 6 Conclusions

In this paper, we have presented a simple but effective semi-supervised learning approach and demonstrated that it achieves substantial improvement over a competitive baseline in two broad-coverage dependency parsing tasks. Despite this success, there are several ways in which our approach might be improved.

To begin, recall that the Brown clustering algorithm is based on a bigram language model. Intuitively, there is a "mismatch" between the kind of lexical information that is captured by the Brown clusters and the kind of lexical information that is modeled in dependency parsing. A natural avenue for further research would be the development of clustering algorithms that reflect the syntactic behavior of words; e.g., an algorithm that attempts to maximize the likelihood of a treebank, according to a probabilistic dependency model. Alternately, one could design clustering algorithms that cluster entire head-modifier arcs rather than individual words.

Another idea would be to integrate the clustering algorithm into the training algorithm in a limited fashion. For example, after training an initial parser, one could parse a large amount of unlabeled text and use those parses to improve the quality of the clusters. These improved clusters can then be used to retrain an improved parser, resulting in an overall algorithm similar to that of McClosky et al. (2006).

Setting aside the development of new clustering algorithms, a final area for future work is the extension of our method to new domains, such as conversational text or other languages, and new NLP problems, such as machine translation.

## Acknowledgments

The authors thank the anonymous reviewers for their insightful comments. Many thanks also to Percy Liang for providing his implementation of the Brown algorithm, and Ryan McDonald for his assistance with the experimental setup. The authors gratefully acknowledge the following sources of support. Terry Koo was funded by NSF grant DMS-0434222 and a grant from NTT, Agmt. Dtd. 6/21/1998. Xavier Carreras was supported by the Catalan Ministry of Innovation, Universities and Enterprise, and a grant from NTT, Agmt. Dtd. 6/21/1998. Michael Collins was funded by NSF grants 0347631 and DMS-0434222.

## References


P.F. Brown, V.J. Della Pietra, P.V. deSouza, J.C. Lai, and R.L. Mercer. 1992. Class-Based n-gram Models of Natural Language. *Computational Linguistics*, 18(4):467–479.

S. Buchholz and E. Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proceedings of CoNLL*, pages 149–164.

X. Carreras. 2007. Experiments with a Higher-Order Projective Dependency Parser. In *Proceedings of EMNLP-CoNLL*, pages 957–961.

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.

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

M. Collins, J. Hajič, L. Ramshaw, and C. Tillmann. 1999. A Statistical Parser for Czech. In *Proceedings of ACL*, pages 505–512.

M. Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of EMNLP*, pages 1–8.

K. Crammer and Y. Singer. 2003. Ultraconservative Online Algorithms for Multiclass Problems. *Journal of Machine Learning Research*, 3:951–991.

K. Crammer, O. Dekel, S. Shalev-Shwartz, and Y. Singer. 2004. Online Passive-Aggressive Algorithms. In S. Thrun, L. Saul, and B. Schölkopf, editors, *NIPS 16*, pages 1229–1236.

J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

J. Eisner. 2000. Bilexical Grammars and Their Cubic-Time Parsing Algorithms. In H. Bunt and A. Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers.

Y. Freund and R. Schapire. 1999. Large Margin Classification Using the Perceptron Algorithm. *Machine Learning*, 37(3):277–296.

J. Hajič, E. Hajičová, P. Pajas, J. Panevova, and P. Sgall. 2001. *The Prague Dependency Treebank 1.0, LDC No. LDC2001T10*. Linguistics Data Consortium.

J. Hajič. 1998. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. In E. Hajičová, editor, *Issues of Valency and Meaning. Studies in Honor of Jarmila Panevová*, pages 12–19.

K. Hall and V. Novák. 2005. Corrective Modeling for Non-Projective Dependency Parsing. In *Proceedings of IWPT*, pages 42–52.

T. Koo and M. Collins. 2005. Hidden-Variable Models for Discriminative Reranking. In *Proceedings of HLT-EMNLP*, pages 507–514.

W. Li and A. McCallum. 2005. Semi-Supervised Sequence Modeling with Syntactic Topic Models. In *Proceedings of AAAI*, pages 813–818.

P. Liang. 2005. Semi-Supervised Learning for Natural Language. Master's thesis, Massachusetts Institute of Technology.

M.P. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with Latent Annotations. In *Proceedings of ACL*, pages 75–82.

D. McClosky, E. Charniak, and M. Johnson. 2006. Effective Self-Training for Parsing. In *Proceedings of HLT-NAACL*, pages 152–159.

R. McDonald and F. Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *Proceedings of EACL*, pages 81–88.

R. McDonald, K. Crammer, and F. Pereira. 2005a. Online Large-Margin Training of Dependency Parsers. In *Proceedings of ACL*, pages 91–98.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005b. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of HLT-EMNLP*, pages 523–530.

S. Miller, J. Guinness, and A. Zamanian. 2004. Name Tagging with Word Clusters and Discriminative Training. In *Proceedings of HLT-NAACL*, pages 337–342.

J. Nivre and J. Nilsson. 2005. Pseudo-Projective Dependency Parsing. In *Proceedings of ACL*, pages 99–106.

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 *Proceedings of EMNLP-CoNLL 2007*, pages 915–932.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of COLING-ACL*, pages 433–440.

A. Ratnaparkhi. 1996. A Maximum Entropy Model for Part-Of-Speech Tagging. In *Proceedings of EMNLP*, pages 133–142.

I. Titov and J. Henderson. 2007. Constituent Parsing with Incremental Sigmoid Belief Networks. In *Proceedings of ACL*, pages 632–639.

Q.I. Wang, D. Schuurmans, and D. Lin. 2005. Strictly Lexical Dependency Parsing. In *Proceedings of IWPT*, pages 152–159.

H. Yamada and Y. Matsumoto. 2003. Statistical Dependency Analysis With Support Vector Machines. In *Proceedings of IWPT*, pages 195–206.