Simple Training of Dependency Parsers via Structured Boosting

Qin Iris Wang
Department of Computing Science
University of Alberta
wqin@cs.ualberta.ca

Dekang Lin
Google Inc.
lindek@google.com

Dale Schuurmans
Department of Computing Science
University of Alberta
dale@cs.ualberta.ca

Abstract

Recently, significant progress has been made on learning structured predictors via coordinated training algorithms such as conditional random fields and maximum margin Markov networks. Unfortunately, these techniques are based on specialized training algorithms, are complex to implement, and expensive to run. We present a much simpler approach to training structured predictors by applying a boosting-like procedure to standard supervised training methods. The idea is to learn a local predictor using standard methods, such as logistic regression or support vector machines, but then achieve improved structured classification by ”boosting” the influence of misclassified components *after* structured prediction, re-training the local predictor, and repeating. Further improvement in structured prediction accuracy can be achieved by incorporating ”dynamic” features—i.e. an extension whereby the features for one predicted component can depend on the predictions already made for some other components.

We apply our techniques to the problem of learning dependency parsers from annotated natural language corpora. By using logistic regression as an efficient base classifier (for predicting dependency links between word pairs), we are able to efficiently train a dependency parsing model, via structured boosting, that achieves state of the art results in English, and surpasses state of the art in Chinese.

1 Introduction

Recently, a significant amount of progress has been made on developing training algorithms for learning *structured* predictors from data [Tsochantaridis *et al.*, 2004; Altun *et al.*, 2003; Taskar *et al.*, 2003]. Structured prediction learning extends the standard supervised learning framework beyond the univariate setting, where a single output variable is considered, to the multivariate setting, where complex, non-scalar predictions $\hat{\mathbf{y}}$ must be produced for inputs $\mathbf{x}$. The challenge is that each component $\hat{y}_i$ of $\hat{\mathbf{y}}$ should not depend only on the input $\mathbf{x}$, but instead should take into account correlations between $\hat{y}_i$ and its neighboring components $\hat{y}_j \in \hat{\mathbf{y}}$. It has been shown in many application areas that structured prediction models that directly capture the relationships between output components perform better than models that do not directly enforce these relationships [Lafferty *et al.*, 2001; Tsochantaridis *et al.*, 2004; Altun *et al.*, 2003; Taskar *et al.*, 2003; 2004]. In particular, these ideas have started to have a significant impact in the field of natural language parsing [McDonald *et al.*, 2005; McDonald and Pereira, 2006; Corston-Oliver *et al.*, 2006], where state of the art results have recently been achieved through the use of structured training algorithms. Parsing is a large-scale structured prediction problem where multiple predictions must be coordinated to achieve an accurate parse for a given input sentence. Parsing is a particularly challenging and important task for machine learning, since it involves complex outputs (parse trees) and limited training data (manually parsed treebanks), and yet machine learning methods have proved to provide the best approach to obtaining robust parsers for real data.

One drawback with current structured prediction training algorithms, however, is that they involve new, specialized parameter optimization algorithms, that are complex, non-trivial to implement, and usually require far more computation than standard classification learning methods [Lafferty *et al.*, 2001; Taskar *et al.*, 2003]. The main reason for increased complexity is the fact that some form of structured prediction algorithm, such as a parser or a Viterbi decoder, must be considered in the underlying training principle, which causes the structured inference of output predictions to be tightly coupled with the parameter optimization process during training.

In this paper, we demonstrate the somewhat surprising result that state of the art performance on natural language parsing can be achieved through the use of conventional, local classification methods. In particular, we show how a simple form of structured boosting can be used to improve the training of standard local classification methods, in the structured case, without modifying the underlying training method. The advantage of this approach is that one can use off-the-shelf classification techniques, such as support vector machines or logistic regression, to achieve competitive structured prediction results with little additional effort. We achieve this through the use of two simple ideas. First, we introduce a very simple form of “structured boosting”, where a structured predictor, such as a parser, is used to modify the predictions of the local, weak learning algorithm, which then influences

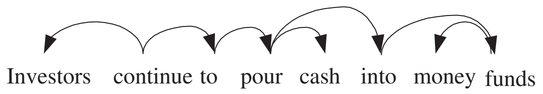

Figure 1: A dependency tree

the example weightings and subsequent hypotheses, implicitly encouraging more accurate structured predictions. Second, we exploit an old idea from natural language parsing, but highlight it here more generally, that "dynamic" features can be used in the local classifiers, which take into account the previous classifications of a restricted set of neighboring examples.

We demonstrate these ideas concretely on the problem of learning natural language dependency parsers from labeled (treebank) data. Although dependency parsing is a very complex problem, we are able to achieve state of the art results by training a local "link predictor" that merely attempts to predict the existence and orientation of a link between two words given input features encoding context—without worrying about coordinating the predictions in a coherent global parse. Instead, a wrapper approach, based on structured boosting, is used to successively modify the training data so that the training algorithm is implicitly encouraged to facilitate improved global parsing accuracy.

The remainder of the paper is organized as follows. First, in Section 2 we briefly describe the dependency parsing problem and introduce some of our notation and terminology. We then explain the relationship between structured prediction learning and traditional classification learning in Section 3, and point out opportunities for connecting these problems. Next, in Section 4 we briefly explain, in general terms, the idea of "dynamic" features in classification learning and how these can be used to improve structured prediction. Then in Section 5 we introduce the main technique proposed in this paper, structured boosting, and explain its relation to standard boosting approaches. Finally, in Sections 6 and 7, we describe our experiments in learning dependency parsers from treebank data, and show how competitive results can be obtained through the use of standard learning methods. In fact, our results surpass state of the art accuracy in Chinese parsing, and are competitive with state of the art in English. Section 8 concludes the paper with a discussion of future work.

2 Dependency Parsing

Since our main application involves learning dependency parsers from labeled data, we briefly introduce the problem and some of the issues it creates. In natural language parsing, a dependency tree specifies which words in a sentence are directly related. That is, the dependency structure of a natural language sentence is a directed tree where the nodes are the words in the sentence and links represent the direct dependency relationships between the words; see Figure 1. Generally speaking, a dependency tree is much easier to understand and annotate than constituency trees that involving part of speech and phrase labels (e.g., Figure 2). Consequently, there has been a growing interest in dependency parsing in recent years. Dependency relations have been playing important roles in machine translation [Fox, 2002; Cherry and Lin, 2003; Ding and Palmer, 2005], coreference resolution [Bergsma and Lin, 2006], and information extraction [Culotta and Sorensen, 2004].

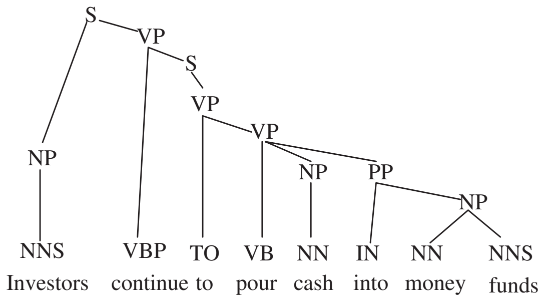

Figure 2: A constituency tree

We consider the problem of automatically learning a dependency parser for a given language from a treebank: a collection of manually parsed sentences. Treebanks usually consist of constituency parses, but a dependency parse can be automatically extracted from a constituency parse by a few simple rules [Bikel, 2004]. Thus, the training data consists of a set of sentences, each annotated with a directed spanning tree over words (Figure 1). For most languages, like English and Chinese, this tree is planar (often called "projective").

Although a dependency tree is a complex object, parsing can conceptually be reduced to a set of local classifications: each word pair in a sentence can be classified into one of three categories; no link, left link, or right link. One aspect of the problem is *local*: for a given pair of words in a given sentence context, what should their link label be? The other aspect of the problem is *global*: how should the local link predictions be coordinated globally to produce a consistent and accurate dependency tree? Both aspects of the problem—local link prediction versus global link coordination—are crucial to achieving a state of the art dependency parser. Locally, the problem is amenable to standard classification learning approaches. Globally, the problem requires coordination—that is, interaction with a parsing algorithm.

3 From Local to Coordinated Training

The problem of learning a structured predictor can generally be formulated as follows. We are given a set of annotated objects $(\mathbf{s}_1, \mathbf{t}_1), ..., (\mathbf{s}_T, \mathbf{t}_T)$, where each object $\mathbf{s}_i$ consists of a composite observation (i.e. a sentence consisting of a word string) and each annotation $\mathbf{t}_i$ is a complete labeling of the relevant subcomponents in $\mathbf{s}_i$ (i.e. a link label for every pair of words in the sentence). Typically, a single composite example $(\mathbf{s}_i, \mathbf{t}_i)$ can be broken down into a (coordinated) set of *local* examples $(\mathbf{s}_{i,1}, t_{i,1})|_{(\mathbf{s}_i,\mathbf{t}_i)}, ...(\mathbf{s}_{i,N}, t_{i,N})|_{(\mathbf{s}_i,\mathbf{t}_i)}$, where each label $t_{ij} \in \mathbf{t}_i$ is an atomic classification of a subcomponent $\mathbf{s}_{i,j} \in \mathbf{s}_i$, taken in context $(\mathbf{s}_i, \mathbf{t}_i)$. For example, a sentence $\mathbf{s}_i = w_{i,1}...w_{i,n}$ and a dependency tree labeling $\mathbf{t}_i = t_{i,1,2}...t_{i,n-1,n}$ can be decomposed into local examples $(w_{i,1}, w_{i,2}; t_{i,1,2})|_{(\mathbf{s}_i,\mathbf{t}_i)}, ...(w_{i,n-1}, w_{i,n}; t_{i,n-1,n})|_{(\mathbf{s}_i,\mathbf{t}_i)}$, consisting of arbitrary (not necessarily adjacent) word pairs and their link label (none, left, right) in context $(\mathbf{s}_i, \mathbf{t}_i)$. The context is important because accurately predicting a component label, even locally, requires the consideration of more than just the subcomponent (word pair) itself, but also the surrounding components, and possibly even the labels of some of the surrounding components. (We will discuss the latter point in more detail below.)

This decomposition facilitates a purely local approach to the learning problem. Given the original composite data $(\mathbf{s}_1, \mathbf{t}_1), ..., (\mathbf{s}_T, \mathbf{t}_T)$ (e.g. sentences and their parses) one can first break the data up into local examples $(w_{1,1}, w_{1,2}; t_{1,1,2})|_{(\mathbf{s}_1,\mathbf{t}_1)}, ...(w_{i,j}, w_{i,k}; t_{i,j,k})|_{(\mathbf{s}_i,\mathbf{t}_i)}, ...$, ignore the relationships between examples, and use a standard supervised learning algorithm to learn a local predictor $w_{i,j}, w_{i,k} \mapsto t_{i,j,k}$ in context $(\mathbf{s}_i, \mathbf{t}_i)$. For example, if we restrict attention to linear predictors (support vector machines or logistic regression models), we only need to learn a weight vector $\boldsymbol{\theta}$ over a set of features defined on the local examples $\mathbf{f}(w_{i,j}, w_{i,k}, t_{i,j,k}; \mathbf{s}_i, \mathbf{t}_i)$. Here, each feature f_m computes its value based on the component $(w_{i,j}, w_{i,k})$, the label $t_{i,j,k}$, in their context $(\mathbf{s}_i, \mathbf{t}_i)$. In this case, a multiclass support vector machine [Crammer and Singer, 2001] or logistic regression model [Hastie *et al.*, 2001] could be trained in a conventional manner to achieve an accurate local prediction model.

The only question that remains is how to perform valid structured prediction on composite test objects using a local prediction model. The problem is that structured classification requires that *constraints* be respected between the classifications of different local components. For example, in dependency parsing, the predicted word pair labels (no link, left link, right link) must form a valid directed spanning tree, which in the case of some languages like English and Chinese, should also be a planar tree. This form of global consistency is achieved in practice simply by combining a local link classifier with a parsing algorithm. A dependency parsing algorithm is, in effect, a dynamic programming algorithm that has the goal of producing a maximum weight spanning tree subject to the constraints [Eisner, 1996]. The output of the local predictor can be used as a numerical weight to score a potential link, in context, which the parser can then use to make decisions about which link labels to choose. In this sense, the role of a local predictor is just to supply a learned scoring function to a pre-existing parsing algorithm. Exactly this approach to combining local link predictors with dependency parsing algorithms has been tried, with some success, by many researchers—using support vector machines [Yamada and Matsumoto, 2003], logistic regression (aka. maximum entropy models) [Ratnaparkhi, 1999; Charniak, 2000], and generative probability models [Collins, 1997; Wang *et al.*, 2005]—to learn local scoring functions.

Unfortunately, these simple local learning strategies have an obvious shortcoming. The problem is that the training loss being minimized during local parameter optimization has nothing directly to do with the parser. Although it is true that an accurate local predictor is a prerequisite for an accurate parse prediction, the parameters of the local model are not being trained to directly optimize the global accuracy of the parser. That is, a far better choice of parameters might exist within the given space defined by the features that leads to better global parsing accuracy. This is where the advent of recent training algorithms for learning *structured* predictors has been helpful. The main idea behind these training algorithms has been to explicitly incorporate the effects of the structured predictor *directly* into the training algorithm. That is, parameter optimization of a local predictor is performed by directly considering the implied effects on the structured (global rather than local) prediction error. The extension to structured training loss has been developed for both the large margin training principle of support vector machines [Tsochantaridis *et al.*, 2004; Altun *et al.*, 2003; Taskar *et al.*, 2003] and the maximum conditional likelihood principle of logistic regression [Lafferty *et al.*, 2001]. Subsequently, training algorithms based on these principles have been applied to parsing [Taskar *et al.*, 2004; Wang *et al.*, 2006], and recently resulted in state of the art accuracy for English dependency parsing [McDonald *et al.*, 2005; McDonald and Pereira, 2006]. Unfortunately, the main drawback with these structured training techniques is that they are specialized, non-trivial to implement, and require a great deal of refinement and computational resources to apply to a significant task like parsing [McDonald *et al.*, 2005; Corston-Oliver *et al.*, 2006].

In this paper, we pursue a simpler, more general approach that can be applied to any local prediction learning strategy, without requiring that the underlying training algorithm be modified, while still ensuring that the training outcome is directly influenced by the resulting accuracy of the global structured predictor (the parser).

4 Dynamic Features

Before describing our general structured boosting method, we first briefly describe a simple but useful idea for improving the structured classification accuracy of local predictors. The idea is to use so-called "dynamic" features; that is, features that take into account the *labels* of (some) of the surrounding components when predicting the label of a target component. In particular, when predicting the label of a target component $\mathbf{s}_{i,j} \in \mathbf{s}_i$ from a composite object $\mathbf{s}_i$, one can assume that the labels for *some* other components, say $\mathbf{s}_{i,j-1} \in \mathbf{s}_i$, have already been computed. The only constraint is that the neighboring labels used must always be available before attempting to label $\mathbf{s}_{i,j}$. For supervised training, this is never a problem, because the labels are always available for every component. However, the real issue arises during testing—that is, during the structured classification of a test object. Here the labels for each component have to be determined in a systematic order that ensures the required features are always available when the next component needs to be labeled.

The easiest way to illustrate the concept is in a sequential labeling task, like part of speech tagging: Given a sentence $\mathbf{s} = w_1...w_n$, the goal is to predict the corresponding tag sequence $\mathbf{t} = t_1...t_n$. Here the *preceding* tags can be used as features for the current word under consideration—e.g. in a maximum entropy Markov model (MEMM) [McCallum *et al.*, 2000]—while still permitting an efficient Viterbi decoding algorithm to be used for structured prediction. Alternatively, one could use the *following* tags as features or use both the preceding and following tags [Toutanova *et al.*, 2003]. The idea of dynamic features, however, is more general than sequence labeling and maximum conditional likelihood training.

For dependency parsing, dynamic features can also be easily employed. For example, when considering a possible link label that connects a head word to a subordinate word, one will always have access (in any standard parsing algorithm) to the existing children of the head that occur between the two words under consideration. In this case, the number and types of pre-existing subordinate children are valid features that can be used to predict whether the new head-subordinate link should occur, which turns out to be a very informative feature for link prediction in parsing [Collins, 1997; Wang *et al.*, 2005].

Although the idea of using dynamic features is not new in some fields like parsing [Collins, 1997; Magerman, 1995], it is still not as widely appreciated as perhaps it should be. Dynamic features are often a trivial way to improve structured predictors without requiring any modification of the underlying training methods. In fact, even still the possibility is not always used in parsing [McDonald *et al.*, 2005], only to yield immediate improvements when subsequently reintroduced [McDonald and Pereira, 2006]. Below we find that simple dynamic features easily improve structured prediction performance.

5 Structured Boosting

Even though dynamic features can significantly improve the structured prediction performance of local training algorithms, local training is still myopic, and the parameter optimization process remains uninfluenced by the global behavior of the structured predictor. In fact, even though dynamic features capture some weak aspects of global prediction, we still expect proper structured training algorithms to yield significant global accuracy improvements in structured problems like parsing. For example, in simpler sequence labeling tasks, it has already been well observed that structured training algorithms, like conditional random fields [Lafferty *et al.*, 2001], outperform their myopic counterparts, maximum entropy models [McCallum *et al.*, 2000], because MEMM training still fails to consider global prediction accuracy [Lafferty *et al.*, 2001]. However, by attempting to incorporate the global predictor directly into local parameter optimization, one is inevitably led to design new, complex training algorithms that require standard local training methods to be replaced. The problem we seek to avoid is to complicate the training process in this manner.

We now introduce our main proposal, structured boosting, that provides a straightforward way to combine global structured prediction (parsing) with local parameter optimization, without modifying the underlying local training algorithm. In fact, the procedure is a trivial variant of standard boosting algorithms [Freund and Schapire, 1996; Schapire and Singer, 1999; Collins *et al.*, 2002], altered to incorporate a structured prediction algorithm during the classification phase. The procedure is as follows.

- First, a standard predictor is trained on the local labeled components, as discussed in Sec. 3, to produce a "weak" local predictor (e.g. a local link predictor for parsing).
- Then the global structured predictor is used to classify the data using the current weak local predictor. For example, a parsing algorithm is used to re-predict the training labels, in a coordinated global fashion, using the learned link predictor as an internal scoring function.
- Based on the resulting misclassifications of the *structured predictor* (i.e. the parser output), the ensemble weight for the current weak local predictor is calculated, and the local example weights are updated, according to any standard boosting method; for example, either exponential loss Adaboost [Freund and Schapire, 1996; Schapire and Singer, 1999] or logistic regression loss boosting [Collins *et al.*, 2002].
- The above steps are then repeated for some number of boosting rounds.

The resulting ensemble of weak local predictors then provides a combined local predictor that can be used for subsequent global structured prediction on composite test examples.

The advantage of this approach is its simplicity and generality. It can be applied to any standard local training method without requiring any modification of the underlying algorithm, yet via structured boosting, the local learning algorithm is forced to respond to the behavior of the global predictor. In effect, it is a simple training wrapper, where local examples are reweighted, not based on the predictions of a current hypothesis, but instead on the predictions that the local hypothesis forces the global structured predictor to make. Below we find that a structured boosting method of this form can improve the quality of dependency parsers learned from treebank data. Note that only a few boosting rounds are ever feasible in our application, because each round requires the entire corpus to be re-parsed and the local prediction model re-trained. Nevertheless, we still witness some useful improvements and achieve state of the art results.

6 Experimental Design: Dependency Parsing

We applied the above ideas for learning structured predictors to the challenging problem of learning a dependency parser from treebank data. In particular, we considered two languages, English and Chinese.

Data sets We used the English Penn Treebank 3.0 [Marcus *et al.*, 1993] and the Chinese Treebanks 4.0 [Palmer *et al.*, 2004] and 5.0 [Palmer *et al.*, 2005] for our experiments. For English, we converted the constituency structures to dependency trees using the same rules as [Yamada and Matsumoto, 2003] and adopted the standard training/development/test split used throughout the literature. The development and test sets were tagged with the part of speech tagger of [Ratnaparkhi, 1996]. For Chinese, we used the rules in [Bikel, 2004] for conversion and created the same data split as [Wang *et al.*, 2005] on the Chinese Treebank 4.0 data set, and the same data split as [Corston-Oliver *et al.*, 2006] on the Chinese Treebank 5.0 data set. Chinese Treebank 5.0 contains Chinese Treebank 4.0 as a subset, but adds approximately 3000 sentences (100,000 words) of Taiwanese Chinese text.

Static features For both English and Chinese we used a common set of feature templates. In particular, for the static features, we used the same set of features described in [McDonald *et al.*, 2005], except the “In Between POS Features”. Given a target word pair and their context, these static features consisted of indicators of the individual words, their part of speech tags, and also the part of speech tags of words in the surrounding context. In addition to the indicator features used in [McDonald *et al.*, 2005] we also added a distance feature that simply measures how far apart the two words are in the sentence, which is highly predictive of link existence, since most links in a dependency parse are short range.

Dynamic features For dynamic features, we used the number of previous children of a candidate head word, and an indicator of the part of speech, if any, of the previous child word on the same side of the candidate head word. For English, we used one special dynamic feature to try to capture prepositional phrase attachment preference: if a candidate child is tagged PP, then we use a feature that indicates the tag and word of the first grandchild (first child of the child).

Local training For the local training algorithm we used a standard logistic regression model (aka maximum entropy model) [Hastie *et al.*, 2001] to attempt to learn a predictor for one of three word pair labels (no link, left link, right link), given the features described above. To regularize the parameters in the model, we used the linear, non-negative regularizer described in [Goodman, 2004]. The regularization parameter, λ, was set to 0.5 in our experiments. This parameter was selected with some tuning on the English development set, and then used without modification on the other data sets. Unfortunately, the number of features and number of local examples were both so large that training the logistic regression model, even once, took more than a day. So to accelerate the training process, we employed a further trick: We partitioned the set of local examples (determined by word pairs in each sentence) according to the part of speech tags of the pair. Within each equivalence class, the number of features could then be further reduced by dropping those features that became constant within the class. This partitioning dropped the overall training cost to a few hours on a few computers, since the separate partitions could then be trained in parallel. Interestingly, the quality of the learned model was not significantly affected by this training procedure. This suggests that the part of speech tags of the word pair, which are used to create the partitions, are the most essential piece of information in deciding the existence and orientation of a dependency link.

Parser There are many dependency parsing algorithms available with differing computational cost. They range from Eisner’s $O(n^3)$ time parser, where n is the length of the sentence, to an $O(n^5)$ time CKY parser [McDonald *et al.*, 2005]. The difference between these algorithms has to do with the global constraints they can enforce and the types of features they can use during dynamic programming. Faster algorithms enforce fewer global constraints and need to use a more restricted class of features. In our experiments, we used a standard CKY parser [Jurafsky and Martin, 2000]. which allowed us to use all of the features described above, while also enforcing the planarity constraint.

Boosting method We experimented with a few boosting methods, including a simplified variant where the weights of each mis-parsed local example were simply increased by an additive constant, with other weights kept the same, and only the last hypothesis is kept. In fact, in our experiments below we obtain state of the art results just using this simplified procedure, and so we focus on these initial results here. Comparisons to standard boosting algorithms, such as Adaboost M1, M2 [Freund and Schapire, 1997] and the logistic regression form of boosting described in [Collins *et al.*, 2002] are still in progress.

Table 1: Boosting with static features

Iter	English			Chinese (Treebank 4.0)		
	DA	RA	CM	DA	RA	CM
1	87.77	89.61	27.44	82.20	88.58	19.38
2	88.08	89.61	28.97	82.33	89.62	19.72
3	88.10	89.74	28.81	82.22	89.97	18.69
4	88.49	90.62	29.04	82.79	89.97	19.38

7 Results

To determine the effectiveness and generality of our approach we conducted a number of experiments on each of the data sets (English and Chinese). These results were achieved using only the simplified boosting procedure mentioned above (additive weight updates, keeping only the last hypothesis).

First, to determine the effectiveness of the basic structured boosting idea, we started with a simple local prediction model (static features only) and measured parsing accuracy on the held out test set as a function of the number of boosting rounds. Table 1 shows that parsing accuracy is improved in each round of boosting with static features, on both English and Chinese (using Chinese Treebank 4.0). To explain the improvements more carefully, note that DA (Dependency Accuracy) indicates how many word pairs are correctly linked; RA (Root Accuracy) measures how many sentence roots are correctly identified; and CM (Complete Match) is the number of sentences where the entire tree is correct. Our focus in this work is on improving the *dependency accuracy* scores (DA), rather than the root accuracy and complete match scores. This fact is reflected in the boosting procedure, since instance reweighting is based only on whether each candidate link is predicted correctly, not whether the root is labeled correctly, nor whether the complete sentence is matched correctly.

Not surprisingly, Table 1 shows that the dependency accuracy (DA) improves on each round of boosting for English, and improves on most rounds (and improves overall) for Chinese; while the RA and CM results fluctuate somewhat. Note that although the improvements appear small, the observed DA differences are all statistically significant. For English, the test corpus consists of 564,848 instances (word pairs occurring in a sentence), and differences of 0.02 in the percentages shown in the tables are statistically significant with greater than 99% confidence. For Chinese, the test corpus consists of 99,922 instances, and differences of 0.05 in the percentages shown in the tables are statistically significant with greater than 99% confidence.

Second, to determine the effectiveness of dynamic fea-

Table 2: Boosting with dynamic features

Iter	English			Chinese (Treebank 4.0)		
	DA	RA	CM	DA	RA	CM
1	89.10	90.36	33.77	86.08	92.39	25.26
2	89.15	89.65	34.56	86.25	92.39	26.64
3	89.20	89.69	34.31	86.45	91.70	28.72
4	89.22	90.20	34.35	86.58	92.04	28.37

tures, we added these additional features to the local prediction model and repeated the previous boosting experiment. Table 2 shows a significant further improvement in parsing accuracy over just using the static features alone (Table 1). Once again, however, boosting provides further improvement over the base model on both English and Chinese with respect to dependency accuracy. In each case the improvement is significant.

Finally, we compare the final results we were able to achieve to the state of the art. Table 3 shows the best results achieved by our method and other researchers on English and Chinese data. Once again, all of the results on English are obtained on the same standard training and test set splits on the English Penn Treebank 3.0 [Marcus *et al.*, 1993]. The results on Chinese are obtained on two different data sets, Chinese Treebank 4.0 [Palmer *et al.*, 2004] and Chinese Treebank 5.0 [Palmer *et al.*, 2005] as noted. In the table, Y&M03 refers to [Yamada and Matsumoto, 2003], N&S04 refers to [Nivre and Scholz, 2004], WSL05 refers to [Wang *et al.*, 2005], MIRA05 refers to [McDonald *et al.*, 2005], MIRA06 refers to [McDonald and Pereira, 2006], BPM06 refers to [Corston-Oliver *et al.*, 2006]. From Table 3 we can see that on English, the results we are able to achieve through the simple boosting method are competitive with the state of the art, but are still behind the best results of [McDonald and Pereira, 2006]. However, perhaps surprisingly, Table 3 shows that the technique we have proposed in this paper actually achieves state of the art accuracy on Chinese parsing for both treebank collections.[1] [2]

Computational complexity Clearly, there is some computational overhead associated with training by boosting, since each round requires the base learning algorithm to be retrained on the re-weighted training data. The training cost scales up proportional to the number of boosting iterations however, and reasonable improvements can be achieved with a small number of rounds. Interestingly, we have found for test complexity, the computational cost of using a composite hypothesis for scoring the local predictions does not add much overhead to the parsing complexity (although we report only single hypothesis results here).

[1] The results on Chinese Treebank 5.0 are generally worse than on Chinese Treebank 4.0, since the former is a superset of the latter, and moreover the additional sentences come entirely from a Taiwanese Chinese source that is more difficult to parse than the rest of the data [Palmer *et al.*, 2005; 2004].

[2] In fact, MIRA has been tried on Chinese Treebank 4.0 with the same data split reported above, obtaining a dependency accuracy score of 82.5, which does not match the 86.6 percent dependency accuracy achieved by the boosting technique on this data (Ryan McDonald, personal communication).

Table 3: Comparison with state of the art

Model	English			Chinese		
	DA	RA	CM	DA	RA	CM
Y&M03	90.3	91.6	38.4	-	-	-
N&S04	87.3	84.3	30.4	-	-	-
WSL05	-	-	-	79.9*	-	-
MIRA05	90.9	94.2	37.5	-	-	-
MIRA06	91.5	-	42.1	-	-	-
BPM06	90.8	93.7	37.6	73.3†	66.2†	18.2†
our model	89.2	90.2	34.4	77.6† 86.6*	60.6† 92.0*	13.5† 28.4*

* Obtained with Chinese Treebank 4.0 using the data split reported in [Wang *et al.*, 2005].
† Obtained with Chinese Treebank 5.0 using the data split reported in [Corston-Oliver *et al.*, 2006].

8 Conclusion

We have addressed the problem of learning structured predictors by using a simple form of boosting to augment the training of standard local predictors. The procedure is general, and allows one to improve performance at structured prediction without modifying the underlying training algorithm, nor implementing a complex training algorithm. Further improvements in structured prediction accuracy are easily obtained by using dynamic features that consider the labels of some neighboring examples. Although our results are very promising, and in fact provide the new state of the art result in Chinese parsing, there remain many directions for future work. One obvious direction is to investigate the effect of using alternative boosting algorithms, and also to investigate the theoretical nature of applying these algorithms to the structured boosting case: under what circumstances do the algorithms converge, and what guarantees can be made about their performance. We would also like to explore further ideas about useful features for parsing, and additional smoothing and regularization techniques for local training.

9 Acknowledgments

Research supported by the Alberta Ingenuity Centre for Machine Learning, NSERC, MITACS, CFI and the Canada Research Chairs program.

References

[Altun *et al.*, 2003] Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Markov support vector machines. In *Proc. ICML*, 2003.

[Bergsma and Lin, 2006] Shane Bergsma and Dekang Lin. Bootstrapping path-based pronoun resolution. In *Proc. COLING/ACL-06*, 2006.

[Bikel, 2004] Dan Bikel. Intricacies of collins' parsing model. *Computational Linguistics*, 30(4), 2004.

[Charniak, 2000] E. Charniak. A maximum entropy inspired parser. In *Proc. NAACL*, pages 132–139, 2000.

[Cherry and Lin, 2003] C. Cherry and D. Lin. A probability model to improve word alignment. In *Proc. ACL*, pages 88–95, 2003.

[Collins *et al.*, 2002] Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, adaboost and bregman distances. In *Machine Learning 48*, 2002.

[Collins, 1997] M. J. Collins. Three generative, lexicalized models for statistical parsing. In *Proc. ACL*, pages 16–23, 1997.

[Corston-Oliver *et al.*, 2006] S. Corston-Oliver, Aue A., Duh K., and Ringger E. Multilingual dependency parsing using bayes' point machines. In *Proceedings HLT/NAACL*, 2006.

[Crammer and Singer, 2001] K. Crammer and Y. Singer. On the algorithmic interpretation of multiclass kernel-based vector machines. *JMLR*, 2, 2001.

[Culotta and Sorensen, 2004] A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In *Pro. ACL*, 2004.

[Ding and Palmer, 2005] Yuan Ding and Martha Palmer. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proc. ACL*, 2005.

[Eisner, 1996] J. Eisner. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING*, 1996.

[Fox, 2002] Heidi J. Fox. Phrasal cohesion and statistical machine translation. In *Proceedings of EMNLP-02*, pages 304–311, 2002.

[Freund and Schapire, 1996] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, 1996.

[Freund and Schapire, 1997] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Computer and System Sciences*, 55(1), 1997.

[Goodman, 2004] Joshua Goodman. Exponential priors for maximum entropy models. In *Proc. NAACL*, 2004.

[Hastie *et al.*, 2001] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[Jurafsky and Martin, 2000] D. Jurafsky and J. Martin. *Speech and Language Processing*. Prentice Hall, 2000.

[Lafferty *et al.*, 2001] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*, 2001.

[Magerman, 1995] David Magerman. Statistical decision-tree model for parsing. In *Proc. ACL*, pages 276–283, 1995.

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

[McCallum *et al.*, 2000] Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum entropy markov methods for information extraction and segmentation. In *Proc. of ICML*, 2000.

[McDonald and Pereira, 2006] R. McDonald and F. Pereira. Online learning of approximate dependency parsing algorithmsx. In *Proceedings of EACL*, 2006.

[McDonald *et al.*, 2005] Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *Proceedings of ACL*, Ann Arbor, Michigan, 2005.

[Nivre and Scholz, 2004] J. Nivre and M. Scholz. Deterministic dependency parsing of english text. In *Proc. COLING*, 2004.

[Palmer *et al.*, 2004] M. Palmer *et al. Chinese Treebank 4.0*. Linguistic Data Consortium, 2004.

[Palmer *et al.*, 2005] M. Palmer *et al. Chinese Treebank 5.0*. Linguistic Data Consortium, 2005.

[Ratnaparkhi, 1996] Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP-96*, 1996.

[Ratnaparkhi, 1999] Adwait Ratnaparkhi. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175, 1999.

[Schapire and Singer, 1999] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *Machine Learning 37*, 1999.

[Taskar *et al.*, 2003] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Proceedings NIPS 16*, 2003.

[Taskar *et al.*, 2004] B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. Max-margin parsing. In *Proc. EMNLP*, 2004.

[Toutanova *et al.*, 2003] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL*, 2003.

[Tsochantaridis *et al.*, 2004] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proc. ICML*, 2004.

[Wang *et al.*, 2005] Q. Wang, D. Schuurmans, and D. Lin. Strictly lexical dependency parsing. In *Proc. IWPT*, pages 152–159, 2005.

[Wang *et al.*, 2006] Q. Wang, C. Cherry, D. Lizotte, and D. Schuurmans. Improved large margin dependency parsing via local constraints and laplacian regularization. In *Proc. of CoNLL*, pages 21–28, 2006.

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