Improved Large Margin Dependency Parsing via Local Constraints and Laplacian Regularization

Qin Iris Wang **Colin Cherry** **Dan Lizotte** **Dale Schuurmans**
Department of Computing Science
University of Alberta
{wqin,colinc,dlizotte,dale}@cs.ualberta.ca

Abstract

We present an improved approach for learning dependency parsers from treebank data. Our technique is based on two ideas for improving large margin training in the context of dependency parsing. First, we incorporate local constraints that enforce the correctness of each individual link, rather than just scoring the global parse tree. Second, to cope with sparse data, we smooth the lexical parameters according to their underlying word similarities using Laplacian Regularization. To demonstrate the benefits of our approach, we consider the problem of parsing Chinese treebank data using only lexical features, that is, without part-of-speech tags or grammatical categories. We achieve state of the art performance, improving upon current large margin approaches.

1 Introduction

Over the past decade, there has been tremendous progress on learning parsing models from treebank data (Collins, 1997; Charniak, 2000; Wang et al., 2005; McDonald et al., 2005). Most of the early work in this area was based on postulating *generative* probability models of language that included parse structure (Collins, 1997). Learning in this context consisted of estimating the parameters of the model with simple likelihood based techniques, but incorporating various smoothing and back-off estimation tricks to cope with the sparse data problems (Collins, 1997; Bikel, 2004). Subsequent research began to focus more on *conditional* models of parse structure given the input sentence, which allowed discriminative training techniques such as maximum conditional likelihood (i.e. "maximum entropy") to be applied (Ratnaparkhi, 1999; Charniak, 2000). In fact, recently, effective conditional parsing models have been learned using relatively straightforward "plug-in" estimates, augmented with similarity based smoothing (Wang et al., 2005). Currently, the work on conditional parsing models appears to have culminated in large margin training (Taskar et al., 2003; Taskar et al., 2004; Tsochantaridis et al., 2004; McDonald et al., 2005), which currently demonstrates the state of the art performance in English dependency parsing (McDonald et al., 2005).

Despite the realization that maximum margin training is closely related to maximum conditional likelihood for conditional models (McDonald et al., 2005), a sufficiently unified view has not yet been achieved that permits the easy exchange of improvements between the probabilistic and non-probabilistic approaches. For example, smoothing methods have played a central role in probabilistic approaches (Collins, 1997; Wang et al., 2005), and yet they are not being used in current large margin training algorithms. However, as we demonstrate, smoothing can be applied in a large margin training framework, and it leads to generalization improvements in much the same way as probabilistic approaches. The second key observation we make is somewhat more subtle. It turns out that probabilistic approaches pay closer attention to the individual errors made by each component of a parse, whereas the training error minimized in the large margin approach—the "structured margin loss" (Taskar et al., 2003; Tsochantaridis et al., 2004; McDonald et al., 2005)—is a coarse measure that only assesses the total error of an entire parse rather than focusing on the error of any particular component.

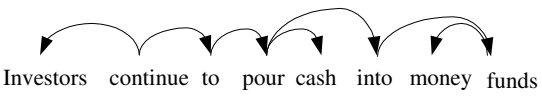

Figure 1: A dependency tree

In this paper, we make two contributions to the large margin approach to learning parsers from supervised data. First, we show that smoothing based on lexical similarity is not only possible in the large margin framework, but more importantly, allows better generalization to new words not encountered during training. Second, we show that the large margin training objective can be significantly refined to assess the error of each component of a given parse, rather than just assess a global score. We show that these two extensions together lead to greater training accuracy and better generalization to novel input sentences than current large margin methods.

To demonstrate the benefit of combining useful learning principles from both the probabilistic and large margin frameworks, we consider the problem of learning a dependency parser for Chinese. This is an interesting test domain because Chinese does not have clearly defined parts-of-speech, which makes lexical smoothing one of the most natural approaches to achieving reasonable results (Wang et al., 2005).

2 Lexicalized Dependency Parsing

A dependency tree specifies which words in a sentence are directly related. That is, the dependency structure of a 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. There has been a growing interest in dependency parsing in recent years. (Fox, 2002) found that the dependency structures of a pair of translated sentences have a greater degree of cohesion than phrase structures. (Cherry and Lin, 2003) exploited such cohesion between the dependency structures to improve the quality of word alignment of parallel sentences. Dependency relations have also been found to be useful in information extraction (Culotta and Sorensen, 2004; Yangarber et al., 2000).

A key aspect of a dependency tree is that it does not necessarily report parts-of-speech or phrase labels. Not requiring parts-of-speech is especially beneficial for languages such as Chinese, where parts-of-speech are not as clearly defined as English. In Chinese, clear indicators of a word's part-of-speech such as suffixes "-ment", "-ous" or function words such as "the", are largely absent. One of our motivating goals is to develop an approach to learning dependency parsers that is strictly lexical. Hence the parser can be trained with a treebank that only contains the dependency relationships, making annotation much easier.

Of course, training a parser with bare word-to-word relationships presents a serious challenge due to data sparseness. It was found in (Bikel, 2004) that Collins' parser made use of bi-lexical statistics only 1.49% of the time. The parser has to compute back-off probability using parts-of-speech in vast majority of the cases. In fact, it was found in (Gildea, 2001) that the removal of bi-lexical statistics from a state of the art PCFG parser resulted in very little change in the output. (Klein and Manning, 2003) presented an unlexicalized parser that eliminated all lexicalized parameters. Its performance was close to the state of the art lexicalized parsers.

Nevertheless, in this paper we follow the recent work of (Wang et al., 2005) and consider a completely lexicalized parser that uses no parts-of-speech or grammatical categories of any kind. Even though a part-of-speech lexicon has always been considered to be necessary in any natural language parser, (Wang et al., 2005) showed that distributional word similarities from a large unannotated corpus can be used to supplant part-of-speech smoothing with word similarity smoothing, to still achieve state of the art dependency parsing accuracy for Chinese.

Before discussing our modifications to large margin training for parsing in detail, we first present the dependency parsing model we use. We then give a brief overview of large margin training, and then present our two modifications. Subsequently, we present our experimental results on fully lexical dependency parsing for Chinese.

3 Dependency Parsing Model

Given a sentence $W = (w_1, ..., w_n)$ we are interested in computing a directed dependency tree,

T, over W. In particular, we assume that a directed dependency tree T consists of ordered pairs $(w_i \rightarrow w_j)$ of words in W such that each word appears in at least one pair and each word has in-degree at most one. Dependency trees are usually assumed to be projective (no crossing arcs), which means that if there is an arc $(w_i \rightarrow w_j)$, then w_i is an ancestor of all the words between w_i and w_j. Let $\Phi(W)$ denote the set of all the directed, projective trees that span W.

Given an input sentence W, we would like to be able to compute the best parse; that is, a projective tree, $T \in \Phi(W)$, that obtains the highest "score". In particular, we follow (Eisner, 1996; Eisner and Satta, 1999; McDonald et al., 2005) and assume that the score of a complete spanning tree T for a given sentence, whether probabilistically motivated or not, can be decomposed as a sum of local scores for each link (a word pair). In which case, the parsing problem reduces to

$$T^* = \arg \max_{T \in \Phi(W)} \sum_{(w_i \rightarrow w_j) \in T} s(w_i \rightarrow w_j) \quad (1)$$

where the score $s(w_i \rightarrow w_j)$ can depend on any measurable property of w_i and w_j within the tree T. This formulation is sufficiently general to capture most dependency parsing models, including probabilistic dependency models (Wang et al., 2005; Eisner, 1996) as well as non-probabilistic models (McDonald et al., 2005). For standard scoring functions, parsing requires an $O(n^3)$ dynamic programming algorithm to compute a projective tree that obtains the maximum score (Eisner and Satta, 1999; Wang et al., 2005; McDonald et al., 2005).

For the purpose of learning, we decompose each link score into a weighted linear combination of features

$$s(w_i \rightarrow w_j) = \boldsymbol{\theta}^\top \mathbf{f}(w_i \rightarrow w_j) \quad (2)$$

where $\boldsymbol{\theta}$ are the weight parameters to be estimated during training.

Of course, the specific features used in any real situation are critical for obtaining a reasonable dependency parser. The natural sets of features to consider in this setting are very large, consisting at the very least of features indexed by all possible lexical items (words). For example, natural features to use for dependency parsing are indicators of each possible word pair

$$f_{uv}(w_i \rightarrow w_j) = 1_{(w_i=u)} 1_{(w_j=v)}$$

which allows one to represent the tendency of two words, u and v, to be directly linked in a parse. In this case, there is a corresponding parameter θ_{uv} to be learned for each word pair, which represents the strength of the possible linkage.

A large number of features leads to a serious risk of over-fitting due to sparse data problems. The standard mechanisms for mitigating such effects are to combine features via *abstraction* (e.g. using parts-of-speech) or *smoothing* (e.g. using word similarity based smoothing). For abstraction, a common strategy is to use parts-of-speech to compress the feature set, for example by only considering the tag of the parent

$$f_{pv}(w_i \rightarrow w_j) = 1_{(pos(w_i)=p)} 1_{(w_j=v)}$$

However, rather than use abstraction, we will follow a purely lexical approach and only consider features that are directly computable from the words themselves (or statistical quantities that are directly measurable from these words).

In general, the most important aspect of a link feature is simply that it measures something about a candidate word pair that is predictive of whether the words will actually be linked in a given sentence. Thus, many other natural features, beyond parts-of-speech and abstract grammatical categories, immediately suggest themselves as being predictive of link existence. For example, one very useful feature is simply the degree of association between the two words as measured by their pointwise mutual information

$$f_{PMI}(w_i \rightarrow w_j) = PMI(w_i, w_j)$$

(We describe in Section 6 below how we compute this association measure on an auxiliary corpus of unannotated text.) Another useful link feature is simply the distance between the two words in the sentence; that is, how many words they have between them

$$f_{dist}(w_i \rightarrow w_j) = |position(w_i) - position(w_j)|$$

In fact, the likelihood of a direct link between two words diminishes quickly with distance, which motivates using more rapidly increasing functions of distance, such as the square

$$f_{dist2}(w_i \rightarrow w_j) = (position(w_i) - position(w_j))^2$$

In our experiments below, we used only these simple, lexically determined features, $\{f_{uv}\}$, f_{PMI}, f_{dist} and f_{dist2}, *without* the parts-of-speech $\{f_{pv}\}$. Currently, we only use undirected forms of these features, where, for example, $f_{uv} = f_{vu}$ for all pairs (or, put another way, we tie the parameters $\theta_{uv} = \theta_{vu}$ together for all u, v). Ideally, we would like to use directed features, but we have already found that these simple undirected features permit state of the art accuracy in predicting (undirected) dependencies. Nevertheless, extending our approach to directed features and contextual features, as in (Wang et al., 2005), remains an important direction for future research.

4 Large Margin Training

Given a training set of sentences annotated with their correct dependency parses, $(W_1, T_1), ..., (W_N, T_N)$, the goal of learning is to estimate the parameters of the parsing model, $\boldsymbol{\theta}$. In particular, we seek values for the parameters that can accurately reconstruct the training parses, but more importantly, are also able to accurately predict the dependency parse structure on future test sentences.

To train $\boldsymbol{\theta}$ we follow the large margin training approach of (Taskar et al., 2003; Tsochantaridis et al., 2004), which has been applied with great success to dependency parsing (Taskar et al., 2004; McDonald et al., 2005). Large margin training can be expressed as minimizing a regularized loss (Hastie et al., 2004)

$$\min_{\boldsymbol{\theta}} \frac{\beta}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta} \quad + \quad \sum_i \max_{L_i} \Delta(L_i, T_i) - (s(\boldsymbol{\theta}, T_i) - s(\boldsymbol{\theta}, L_i)) \tag{3}$$

where T_i is the target tree for sentence W_i; L_i ranges over all possible alternative trees in $\Phi(W_i)$; $s(\boldsymbol{\theta}, T) = \sum_{(w_i \rightarrow w_j) \in T} \boldsymbol{\theta}^\top \mathbf{f}(w_i \rightarrow w_j)$; and $\Delta(L_i, T_i)$ is a measure of distance between the two trees L_i and T_i.

Using the techniques of (Hastie et al., 2004) one can show that minimizing (4) is equivalent to solving the quadratic program

$$\min_{\boldsymbol{\theta}, \boldsymbol{\xi}} \quad \frac{\beta}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta} + \mathbf{e}^\top\boldsymbol{\xi} \quad \text{subject to} \tag{4}$$
$$\xi_i \geq \Delta(T_i, L_i) + s(\boldsymbol{\theta}, L_i) - s(\boldsymbol{\theta}, T_i)$$
$$\text{for all } i, L_i \in \Phi(W_i)$$

which corresponds to the training problem posed in (McDonald et al., 2005).

Unfortunately, the quadratic program (4) has three problems one must address. First, there are exponentially many constraints—corresponding to each possible parse of each training sentence—which forces one to use alternative training procedures, such as incremental constraint generation, to slowly converge to a solution (McDonald et al., 2005; Tsochantaridis et al., 2004). Second, and related, the original loss (4) is only evaluated at the global parse tree level, and is not targeted at penalizing any specific component in an incorrect parse. Although (McDonald et al., 2005) explicitly describes this as an advantage over previous approaches (Ratnaparkhi, 1999; Yamada and Matsumoto, 2003), below we find that changing the loss to enforce a more detailed set of constraints leads to a more effective approach. Third, given the large number of bi-lexical features $\{f_{uv}\}$ in our model, solving (4) directly will over-fit any reasonable training corpus. (Moreover, using a large β to shrink the $\boldsymbol{\theta}$ values does not mitigate the sparse data problem introduced by having so many features.) We now present our refinements that address each of these issues in turn.

5 Training with Local Constraints

We are initially focusing on training on just an undirected link model, where each parameter in the model is a weight $\theta_{ww'}$ between two words, w and w', respectively. Since links are undirected, these weights are symmetric $\theta_{ww'} = \theta_{w'w}$, and we can also write the score in an undirected fashion as: $s(w, w') = \boldsymbol{\theta}^\top \mathbf{f}(w, w')$. The main advantage of working with the undirected link model is that the constraints needed to ensure correct parses on the training data are *much* easier to specify in this case. Ignoring the projective (no crossing arcs) constraint for the moment, an undirected dependency parse can

be equated with a maximum score spanning tree of a sentence. Given a target parse, the set of constraints needed to ensure the target parse is in fact the maximum score spanning tree under the weights $\boldsymbol{\theta}$, by at least a minimum amount, is a simple set of linear constraints: for any edge $w_1 w_2$ that is *not* in the target parse, one simply adds two constraints

$$\begin{aligned} \boldsymbol{\theta}^\top \mathbf{f}(w_1, w_1') &\geq \boldsymbol{\theta}^\top \mathbf{f}(w_1, w_2) + 1 \\ \boldsymbol{\theta}^\top \mathbf{f}(w_2, w_2') &\geq \boldsymbol{\theta}^\top \mathbf{f}(w_1, w_2) + 1 \end{aligned} \quad (5)$$

where the edges $w_1 w_1'$ and $w_2 w_2'$ are the adjacent edges that actually occur in the target parse that are also on the path between w_1 and w_2. (These would have to be the only such edges, or there would be a loop in the parse tree.) These constraints behave very naturally by forcing the weight of an omitted edge to be smaller than the adjacent included edges that would form a loop, which ensures that the omitted edge would not be added to the maximum score spanning tree before the included edges.

In this way, one can simply accumulate the set of linear constraints (5) for every edge that fails to be included in the target parse for the sentences where it is a candidate. We denote this set of constraints by

$$A = \{\boldsymbol{\theta}^\top \mathbf{f}(w_1, w_1') \geq \boldsymbol{\theta}^\top \mathbf{f}(w_1, w_2) + 1\}$$

Importantly, the constraint set A is *convex* in the link weight parameters $\boldsymbol{\theta}$, as it consists only of linear constraints.

Ignoring the non-crossing condition, the constraint set A is exact. However, because of the non-crossing condition, the constraint set A is more restrictive than necessary. For example, consider the word sequence $...w_i w_{i+1} w_{i+2} w_{i+3}...$, where the edge $w_{i+1} w_{i+3}$ is in the target parse. Then the edge $w_i w_{i+2}$ can be ruled out of the parse in one of two ways: it can be ruled out by making its score less than the adjacent scores as specified in (5), *or* it can be ruled out by making its score smaller than the score of $w_{i+1} w_{i+3}$. Thus, the exact constraint contains a disjunction of two different constraints, which creates a *non*-convex constraint in $\boldsymbol{\theta}$. (The union of two convex sets is not necessarily convex.) This is a weakening of the original constraint set A. Unfortunately, this means that, given a large training corpus, the constraint set A can easily become infeasible.

Nevertheless, the constraints in A capture much of the relevant structure in the data, and are easy to enforce. Therefore, we wish to maintain them. However, rather than impose the constraints exactly, we enforce them approximately through the introduction of slack variables $\boldsymbol{\xi}$. The relaxed constraints can then be expressed as

$$\boldsymbol{\theta}^\top \mathbf{f}(w_1, w_1') \geq \boldsymbol{\theta}^\top \mathbf{f}(w_1, w_2) + 1 - \xi_{w_1 w_2, w_1 w_1'} \quad (6)$$

and therefore a maximum soft margin solution can then be expressed as a quadratic program

$$\begin{aligned} \min_{\boldsymbol{\theta}, \boldsymbol{\xi}} \quad & \frac{\beta}{2} \boldsymbol{\theta}^\top \boldsymbol{\theta} + \boldsymbol{\xi}^\top \mathbf{e} \quad \text{subject to} \\ & \{\boldsymbol{\theta}^\top \mathbf{f}(w_1, w_1') \geq \boldsymbol{\theta}^\top \mathbf{f}(w_1, w_2) + 1 - \xi_{w_1 w_2, w_1 w_1'}\} \\ & \text{for all constraints in } A \end{aligned} \quad (7)$$

where $\mathbf{e}$ denotes the vector of all 1's.

Even though the slacks are required because we have slightly over-constrained the parameters, given that there are so many parameters and a sparse data problem as well, it seems desirable to impose a stronger set of constraints. A set of solution parameters achieved in this way will allow maximum weight spanning trees to correctly parse nearly all of the training sentences, even without the non-crossing condition (see the results in Section 8).

This quadratic program has the advantage of producing link parameters that will correctly parse most of the training data. Unfortunately, the main drawback of this method thus far is that it does not offer any mechanism by which the link weights $\theta_{ww'}$ can be *generalized* to new or rare words. Given the sparse data problem, some form of generalization is necessary to achieve good test results. We achieve this by exploiting distributional similarities between words to smooth the parameters.

6 Distributional Word Similarity

Treebanks are an extremely precious resource. The average cost of producing a treebank parse can run as high as 30 person-minutes per sentence (20 words on average). Similarity-based smoothing, on the other hand, allows one to tap into auxiliary sources of raw unannotated text, which is practically unlimited. With this extra data, one can estimate parameters for words that have never appeared in the training corpus.

The basic intuition behind similarity smoothing is that words that tend to appear in the same contexts tend to have similar meanings. This is known as the Distributional Hypothesis in linguistics (Harris, 1968). For example, the words test and exam are similar because both of them can follow verbs such as administer, cancel, cheat on, conduct, etc.

Many methods have been proposed to compute distributional similarity between words, e.g., (Hindle, 1990; Pereira et al., 1993; Grefenstette, 1994; Lin, 1998). Almost all of the methods represent a word by a feature vector where each feature corresponds to a type of context in which the word appeared. They differ in how the feature vectors are constructed and how the similarity between two feature vectors is computed.

In our approach below, we define the features of a word w to be the set of words that occurred within a small window of w in a large corpus. The context window of w consists of the closest non-stop-word on each side of w and the stop-words in between. The value of a feature w' is defined as the pointwise mutual information between the w' and w: $\text{PMI}(w', w) = \log(\frac{P(w,w')}{P(w)P(w')})$. The similarity between two words, $S(w_1, w_2)$, is then defined as the cosine of the angle between their feature vectors.

We use this similarity information both in training and in parsing. For training, we smooth the parameters according to their underlying word-*pair* similarities by introducing a Laplacian regularizer, which will be introduced in the next section. For parsing, the link scores in (1) are smoothed by word similarities (similar to the approach used by (Wang et al., 2005)) before the maximum score projective dependency tree is computed.

7 Laplacian Regularization

We wish to incorporate similarity based smoothing in large margin training, while using the more refined constraints outlined in Section 5.

Recall that most of the features we use, and therefore most of the parameters we need to estimate are based on bi-lexical parameters $\theta_{ww'}$ that serve as undirected link weights between words w and w' in our dependency parsing model (Section 3). Here we would like to ensure that two different link weights, $\theta_{w_1w_1'}$ and $\theta_{w_2w_2'}$, that involve similar words also take on similar values. The previous optimization (7) needs to be modified to take this into account.

Smoothing the link parameters requires us to first extend the notion of word similarity to word-*pair* similarities, since each link involves two words. Given similarities between individual words, computed above, we then define the similarity between word pairs by the geometric mean of the similarities between corresponding words.

$$S(w_1w_1', w_2w_2') = \sqrt{S(w_1, w_2)S(w_1', w_2')} \quad (8)$$

where $S(w_1, w_2)$ is defined as in Section 6 above. Then, instead of just solving the constraint system (7) we can also ensure that similar links take on similar parameter values by introducing a penalty on their deviations that is weighted by their similarity value. Specifically, we use

$$\sum_{w_1w_1'} \sum_{w_2w_2'} S(w_1w_1', w_2w_2')(\theta_{w_1w_1'} - \theta_{w_2w_2'})^2 = 2\boldsymbol{\theta'}^\top L(S)\boldsymbol{\theta'} \quad (9)$$

Here $L(S)$ is the *Laplacian matrix* of S, which is defined by $L(S) = D(S) - S$ where $D(S)$ is a diagonal matrix such that $D_{w_1w_1',w_1w_1'} = \sum_{w_2w_2'} S(w_1w_1', w_2w_2')$. Also, $\boldsymbol{\theta'}$ corresponds to the vector of bi-lexical parameters. In this penalty function, if two edges w_1w_1' and w_2w_2' have a high similarity value, their parameters will be encouraged to take on similar values. By contrast, if two edges have low similarity, then there will be little mutual attraction on their parameter values.

Note, however, that we do not smooth the parameters, θ_{PMI}, θ_{dist}, θ_{dist2}, corresponding to the pointwise mutual information, distance, and squared distance features described in Section 5, respectively. We only apply similarity smoothing to the bi-lexical parameters.

The Laplacian regularizer (9) provides a natural smoother for the bi-lexical parameter estimates that takes into account valuable word similarity information computed as above. The Laplacian regularizer also has a significant computational advantage: it is guaranteed to be a *convex* quadratic function of the parameters (Zhu et al., 2001). Therefore, by combining the constraint system (7) with the Laplacian smoother (9), we can obtain a convex optimization

Table 1: Accuracy Results on CTB Test Set

Features used	Trained w/ local loss	Trained w/ global loss
Pairs	0.6426	0.6184
+ Lap	0.6506	0.5622
+ Dist	0.6546	0.6466
+ Lap + Dist	0.6586	0.5542
+ MI + Dist	0.6707	0.6546
+ Lap + MI + Dist	0.6827	n/a

Table 2: Accuracy Results on CTB Dev Set

Features used	Trained w/ local loss	Trained w/ global loss
Pairs	0.6130	0.5688
+ Lap	0.6390	0.4935
+ Dist	0.6364	0.6130
+ Lap + Dist	0.6494	0.5299
+ MI + Dist	0.6312	0.6182
+ Lap + MI + Dist	0.6571	n/a

procedure for estimating the link parameters

$$\min_{\boldsymbol{\theta},\boldsymbol{\xi}} \quad \frac{\beta}{2}\boldsymbol{\theta}^\top \tilde{L}(S)\boldsymbol{\theta} + \boldsymbol{\xi}^\top \mathbf{e} \quad \text{subject to} \tag{10}$$
$$\{\boldsymbol{\theta}^\top \mathbf{f}(w_1, w_1') \geq \boldsymbol{\theta}^\top \mathbf{f}(w_1, w_2) + 1 - \xi_{w_1 w_2, w_1 w_1'}\}$$
$$\text{for all constraints in } A$$

where $\tilde{L}(S)$ does not apply smoothing to θ_{PMI}, θ_{dist}, θ_{dist2}.

Clearly, (10) describes a large margin training program for dependency parsing, but one which uses word similarity smoothing for the bi-lexical parameters, and a more refined set of constraints developed in Section 5. Although the constraints are more refined, they are fewer in number than (4). That is, we now only have a polynomial number of constraints corresponding to each word pair in (5), rather than the exponential number over every possible parse tree in (4). Thus, we obtain a polynomial size quadratic program that can be solved for moderately large problems using standard software packages. We used CPLEX in our experiments below. As before, once optimized, the solution parameters $\boldsymbol{\theta}$ can be introduced into the dependency model (1) according to (2).

8 Experimental Results

We tested our method experimentally on the Chinese Treebank (CTB) (Xue et al., 2004). The parse trees in CTB are constituency structures. We converted them into dependency trees using the same method and head-finding rules as in (Bikel, 2004). Following (Bikel, 2004), we used Sections 1-270 for training, Sections 271-300 for testing and Sections 301-325 for development. We experimented with two sets of data: CTB-10 and CTB-15, which contains sentences with no more than 10 and 15 words respectively. Table 1, Table 2 and Table 3 show our experimental results trained and evaluated on Chinese Treebank sentences of length no more than 10, using the standard split. For any unseen link in the new sentences, the weight is computed as the similarity weighted average of similar links seen in the training corpus. The regularization parameter β was set by 5-fold cross-validation on the training set.

Table 3: Accuracy Results on CTB Training Set

Features used	Trained w/ local loss	Trained w/ global loss
Pairs	0.9802	0.8393
+ Lap	0.9777	0.7216
+ Dist	0.9755	0.8376
+ Lap + Dist	0.9747	0.7216
+ MI + Dist	0.9768	0.7985
+ Lap + MI + Dist	0.9738	n/a

We evaluate parsing accuracy by comparing the undirected dependency links in the parser outputs against the undirected links in the treebank. We define the accuracy of the parser to be the percentage of correct dependency links among the total set of dependency links created by the parser.

Table 1 and Table 2 show that training based on the more refined *local loss* is far superior to training with the *global loss* of standard large margin training, on both the test and development sets. Parsing accuracy also appears to increase with the introduction of each new feature. Notably, the pointwise mutual information and distance features significantly improve parsing accuracy—and yet we know of no other research that has investigated these features in this context. Finally, we note that Laplacian regularization improved performance as expected, but not for the *global loss*, where it appears to systematically degrade performance (n/a results did not complete in time). It seems that the global loss model may have been over-regularized (Table 3). However, we have picked the β parameter which gave us the

best resutls in our experiments. One possible explanation for this phenomenon is that the interaction between the Laplician regularization in training and the similarity smoothing in parsing, since distributional word similarities are used in both cases.

Finally, we compared our results to the probabilistic parsing approach of (Wang et al., 2005), which on this data obtained accuracies of 0.7631 on the CTB test set and 0.6104 on the development set. However, we are using a much simpler feature set here.

9 Conclusion

We have presented two improvements to the standard large margin training approach for dependency parsing. To cope with the sparse data problem, we smooth the parameters according to their underlying word similarities by introducing a Laplacian regularizer. More significantly, we use more refined local constraints in the large margin criterion, rather than the global parse-level losses that are commonly considered. We achieve state of the art parsing accuracy for predicting undirected dependencies in test data, competitive with previous large margin and previous probabilistic approaches in our experiments.

Much work remains to be done. One extension is to consider directed features, and contextual features like those used in current probabilistic parsers (Wang et al., 2005). We would also like to apply our approach to parsing English, investigate the confusion showed in Table 3 more carefully, and possibly re-investigate the use of parts-of-speech features in this context.

References

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

Eugene Charniak. 2000. A maximum entropy inspired parser. In *Proceedings of NAACL-2000*, pages 132–139.

Colin Cherry and Dekang Lin. 2003. A probability model to improve word alignment. In *Proceedings of ACL-2003*.

M. J. Collins. 1997. Three generative, lexicalized models for statistical parsing. In *Proceedings of ACL-1997*.

Aron Culotta and Jeffery Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL-2004*.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head-automaton grammars. In *Proceedings of ACL-1999*.

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

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of EMNLP-2002*.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of EMNLP-2001*, Pittsburgh, PA.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Press, Boston, MA.

Zelig S. Harris. 1968. *Mathematical Structures of Language*. Wiley, New York.

T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. 2004. The entire regularization path for the support vector machine. *JMLR*, 5.

Donald Hindle. 1990. Noun classification from predicate-argument structures. In *Proceedings of ACL-1990*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL-2003*.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING/ACL-1998*.

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

F. Pereira, N. Tishby, and L. Lee. 1993. Distributional clustering of english words. In *Proceedings of ACL-1993*.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3).

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin markov networks. In *Proc. of NIPS-2003*.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *Proceedings of EMNLP*.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of ICML-2004*.

Q. Wang, D. Schuurmans, and D. Lin. 2005. Strictly lexical dependency parsing. In *Proceedings of IWPT-2005*.

N. Xue, F. Xia, F. Chiou, and M. Palmer. 2004. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 10(4):1–30.

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

R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. 2000. Unsupervised discovery of scenario-level patterns for information extraction. In *Proceedings of ANLP/NAACL-2000*.

Xiaojin Zhu, John Lafferty, and Zoublin Ghahramani. 2001. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of ICML-2003*.