Max-Margin Parsing

Ben Taskar
Computer Science Dept.
Stanford University
btaskar@cs.stanford.edu

Dan Klein
Computer Science Dept.
Stanford University
klein@cs.stanford.edu

Michael Collins
CS and AI Lab
MIT
mcollins@csail.mit.edu

Daphne Koller
Computer Science Dept.
Stanford University
koller@cs.stanford.edu

Christopher Manning
Computer Science Dept.
Stanford University
manning@cs.stanford.edu

Abstract

We present a novel discriminative approach to parsing inspired by the large-margin criterion underlying support vector machines. Our formulation uses a factorization analogous to the standard dynamic programs for parsing. In particular, it allows one to efficiently learn a model which discriminates among the entire space of parse trees, as opposed to reranking the top few candidates. Our models can condition on arbitrary features of input sentences, thus incorporating an important kind of lexical information without the added algorithmic complexity of modeling headedness. We provide an efficient algorithm for learning such models and show experimental evidence of the model's improved performance over a natural baseline model and a lexicalized probabilistic context-free grammar.

1 Introduction

Recent work has shown that discriminative techniques frequently achieve classification accuracy that is superior to generative techniques, over a wide range of tasks. The empirical utility of models such as logistic regression and support vector machines (SVMs) in flat classification tasks like text categorization, word-sense disambiguation, and relevance routing has been repeatedly demonstrated. For sequence tasks like part-of-speech tagging or named-entity extraction, recent top-performing systems have also generally been based on discriminative sequence models, like conditional Markov models (Toutanova et al., 2003) or conditional random fields (Lafferty et al., 2001).

A number of recent papers have considered discriminative approaches for natural language parsing (Johnson et al., 1999; Collins, 2000; Johnson, 2001; Geman and Johnson, 2002; Miyao and Tsujii, 2002; Clark and Curran, 2004; Kaplan et al., 2004; Collins, 2004). Broadly speaking, these approaches fall into two categories, *reranking* and *dynamic programming* approaches. In reranking methods (Johnson et al., 1999; Collins, 2000; Shen et al., 2003), an initial parser is used to generate a number of candidate parses. A discriminative model is then used to choose between these candidates. In dynamic programming methods, a large number of candidate parse trees are represented compactly in a parse tree forest or chart. Given sufficiently "local" features, the decoding and parameter estimation problems can be solved using dynamic programming algorithms. For example, (Johnson, 2001; Geman and Johnson, 2002; Miyao and Tsujii, 2002; Clark and Curran, 2004; Kaplan et al., 2004) describe approaches based on conditional log-linear (maximum entropy) models, where variants of the inside-outside algorithm can be used to efficiently calculate gradients of the log-likelihood function, despite the exponential number of trees represented by the parse forest.

In this paper, we describe a dynamic programming approach to discriminative parsing that is an alternative to maximum entropy estimation. Our method extends the max-margin approach of Taskar et al. (2003) to the case of context-free grammars. The present method has several compelling advantages. Unlike reranking methods, which consider only a pre-pruned selection of "good" parses, our method is an end-to-end discriminative model over the full space of parses. This distinction can be very significant, as the set of n-best parses often does not contain the true parse. For example, in the work of Collins (2000), 41% of the correct parses were not in the candidate pool of ∼30-best parses. Unlike previous dynamic programming approaches, which were based on maximum entropy estimation, our method incorporates an articulated loss function which penalizes larger tree discrepancies more severely than smaller ones.[1]

Moreover, like perceptron-based learning, it requires only the calculation of Viterbi trees, rather than expectations over all trees (for example using the inside-outside algorithm). In practice, it converges in many fewer iterations than CRF-like approaches. For example, while our approach generally converged in 20-30 iterations, Clark and Curran (2004) report experiments involving 479 iterations of training for one model, and 1550 iterations for another.

The primary contribution of this paper is the extension of the max-margin approach of Taskar et al. (2003) to context free grammars. We show that this framework allows high-accuracy parsing in cubic time by exploiting novel kinds of lexical information.

2 Discriminative Parsing

In the discriminative parsing task, we want to learn a function $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is a set of sentences, and $\mathcal{Y}$ is a set of valid parse trees according to a fixed grammar $\mathbf{G}$. $\mathbf{G}$ maps an input $x \in \mathcal{X}$ to a set of candidate parses $\mathbf{G}(x) \subseteq \mathcal{Y}$.[2]

We assume a loss function $L : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$. The function $L(x, y, \hat{y})$ measures the penalty for proposing the parse $\hat{y}$ for x when y is the true parse. This penalty may be defined, for example, as the number of labeled spans on which the two trees do not agree. In general we assume that $L(x, y, \hat{y}) = 0$ for $y = \hat{y}$. Given labeled training examples (x_i, y_i) for $i = 1 \dots n$, we seek a function f with small expected loss on unseen sentences.

The functions we consider take the following linear discriminant form:

$$f_{\mathbf{w}}(x) = \arg \max_{y \in \mathbf{G}(x)} \langle \mathbf{w}, \Phi(x, y) \rangle,$$

where $\langle \cdot, \cdot \rangle$ denotes the vector inner product, $\mathbf{w} \in \mathbb{R}^d$ and Φ is a feature-vector representation of a parse tree $\Phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ (see examples below).[3]

Note that this class of functions includes Viterbi PCFG parsers, where the feature-vector consists of the counts of the productions used in the parse, and the parameters $\mathbf{w}$ are the log-probabilities of those productions.

2.1 Probabilistic Estimation

The traditional method of estimating the parameters of PCFGs assumes a generative grammar that defines $P(x, y)$ and maximizes the joint log-likelihood $\sum_i \log P(x_i, y_i)$ (with some regularization). A alternative probabilistic approach is to estimate the parameters discriminatively by maximizing conditional log-likelihood. For example, the maximum entropy approach (Johnson, 2001) defines a conditional log-linear model:

$$P_{\mathbf{w}}(y \mid x) = \frac{1}{Z_{\mathbf{w}}(x)} \exp\{\langle \mathbf{w}, \Phi(x, y) \rangle\},$$

where $Z_{\mathbf{w}}(x) = \sum_{y \in \mathbf{G}(x)} \exp\{\langle \mathbf{w}, \Phi(x, y) \rangle\}$, and maximizes the conditional log-likelihood of the sample, $\sum_i \log P(y_i \mid x_i)$, (with some regularization).

2.2 Max-Margin Estimation

In this paper, we advocate a different estimation criterion, inspired by the max-margin principle of SVMs. Max-margin estimation has been used for parse reranking (Collins, 2000). Recently, it has also been extended to graphical models (Taskar et al., 2003; Altun et al., 2003) and shown to outperform the standard max-likelihood methods. The main idea is to forego the probabilistic interpretation, and directly ensure that

$$y_i = \arg \max_{y \in \mathbf{G}(x_i)} \langle \mathbf{w}, \Phi(x_i, y) \rangle,$$

for all i in the training data. We define the margin of the parameters $\mathbf{w}$ on the example i and parse y as the difference in value between the true parse y_i and y:

$$\langle \mathbf{w}, \Phi(x_i, y_i) \rangle - \langle \mathbf{w}, \Phi(x_i, y) \rangle = \langle \mathbf{w}, \Phi_{i,y_i} - \Phi_{i,y} \rangle,$$

[1] This articulated loss is supported by empirical success and theoretical generalization bound in Taskar et al. (2003).

[2] For all x, we assume here that $\mathbf{G}(x)$ is finite. The space of parse trees over many grammars is naturally infinite, but can be made finite if we disallow unary chains and empty productions.

[3] Note that in the case that two members y_1 and y_2 have the same tied value for $\langle \mathbf{w}, \Phi(x, y) \rangle$, we assume that there is some fixed, deterministic way for breaking ties. For example, one approach would be to assume some default ordering on the members of $\mathcal{Y}$.

where $\Phi_{i,y} = \Phi(x_i, y)$, and $\Phi_{i,y_i} = \Phi(x_i, y_i)$. Intuitively, the size of the margin quantifies the confidence in rejecting the mistaken parse y using the function $f_{\mathbf{w}}(x)$, modulo the scale of the parameters $||\mathbf{w}||$. We would like this rejection confidence to be larger when the mistake y is more severe, i.e. $L(x_i, y_i, y)$ is large. We can express this desideratum as an optimization problem:

$$\begin{aligned} \max \quad & \gamma & (1) \\ \text{s.t.} \quad & \langle \mathbf{w}, \Phi_{i,y_i} - \Phi_{i,y} \rangle \geq \gamma L_{i,y} \quad \forall y \in \mathbf{G}(x_i); \\ & ||\mathbf{w}||^2 \leq 1, \end{aligned}$$

where $L_{i,y} = L(x_i, y_i, y)$. This quadratic program aims to separate each $y \in \mathbf{G}(x_i)$ from the target parse y_i by a margin that is proportional to the loss $L(x_i, y_i, y)$. After a standard transformation, in which maximizing the margin is reformulated as minimizing the scale of the weights (for a fixed margin of 1), we get the following program:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_i \xi_i & (2) \\ \text{s.t.} \quad & \langle \mathbf{w}, \Phi_{i,y_i} - \Phi_{i,y} \rangle \geq L_{i,y} - \xi_i \quad \forall y \in \mathbf{G}(x_i). \end{aligned}$$

The addition of non-negative slack variables ξ_i allows one to increase the global margin by paying a local penalty on some outlying examples. The constant C dictates the desired trade-off between margin size and outliers. Note that this formulation has an exponential number of constraints, one for each possible parse y for each sentence i. We address this issue in section 4.

2.3 The Max-Margin Dual

In SVMs, the optimization problem is solved by working with the dual of a quadratic program analogous to Eq. 2. For our problem, just as for SVMs, the dual has important computational advantages, including the "kernel trick," which allows the efficient use of high-dimensional features spaces endowed with efficient dot products (Cristianini and Shawe-Taylor, 2000). Moreover, the dual view plays a crucial role in circumventing the exponential size of the primal problem.

In Eq. 2, there is a constraint for each mistake y one might make on each example i, which rules out that mistake. For each mistake-exclusion constraint, the dual contains a variable $\alpha_{i,y}$. Intuitively, the magnitude of $\alpha_{i,y}$ is proportional to the attention we must pay to that mistake in order not to make it.

The dual of Eq. 2 (after adding additional variables α_{i,y_i} and renormalizing by C) is given by:

$$\begin{aligned} \max \quad & C \sum_{i,y} \alpha_{i,y} L_{i,y} - \frac{1}{2} \left\| C \sum_{i,y} (I_{i,y} - \alpha_{i,y}) \Phi_{i,y} \right\|^2 \\ \text{s.t.} \quad & \sum_y \alpha_{i,y} = 1, \quad \forall i; \quad \alpha_{i,y} \geq 0, \quad \forall i, y, \qquad (3) \end{aligned}$$

where $I_{i,y} = I(x_i, y_i, y)$ indicates whether y is the true parse y_i. Given the dual solution α^*, the solution to the primal problem $\mathbf{w}^*$ is simply a weighted linear combination of the feature vectors of the correct parse and mistaken parses:

$$\mathbf{w}^* = C \sum_{i,y} (I_{i,y} - \alpha^*_{i,y}) \Phi_{i,y}.$$

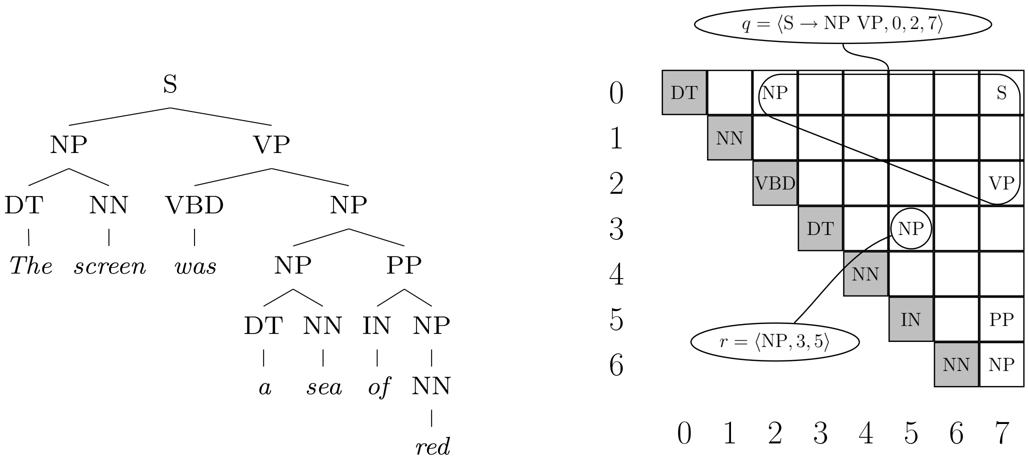
This is the precise sense in which mistakes with large α contribute more strongly to the model.

3 Factored Models

There is a major problem with both the primal and the dual formulations above: since each potential mistake must be ruled out, the number of variables or constraints is proportional to $|\mathbf{G}(x)|$, the number of possible parse trees. Even in grammars without unary chains or empty elements, the number of parses is generally exponential in the length of the sentence, so we cannot expect to solve the above problem without any assumptions about the feature-vector representation Φ and loss function L.

For that matter, for arbitrary representations, to find the best parse given a weight vector, we would have no choice but to enumerate all trees and score them. However, our grammars and representations are generally structured to enable efficient inference. For example, we usually assign scores to local parts of the parse such as PCFG productions. Such factored models have shared substructure properties which permit dynamic programming decompositions. In this section, we describe how this kind of decomposition can be done over the dual α distributions. The idea of this decomposition has previously been used for sequences and other Markov random fields in Taskar et al. (2003), but the present extension to CFGs is novel.

For clarity of presentation, we restrict the grammar to be in Chomsky normal form (CNF), where all rules in the grammar are of the form $\langle A \rightarrow B\ C \rangle$ or $\langle A \rightarrow a \rangle$, where A, B and C are



(a) (b)

Figure 1: Two representations of a binary parse tree: (a) nested tree structure, and (b) grid of labeled spans.

non-terminal symbols, and a is some terminal symbol. For example figure 1(a) shows a tree in this form.

We will represent each parse as a set of two types of parts. Parts of the first type are single constituent tuples $\langle A, s, e, i\rangle$, consisting of a non-terminal A, start-point s and end-point e, and sentence i, such as r in figure 1(b). In this representation, indices s and e refer to positions between words, rather than to words themselves. These parts correspond to the traditional notion of an edge in a tabular parser.

Parts of the second type consist of CF-rule-tuples $\langle A \to B\ C, s, m, e, i\rangle$. The tuple specifies a particular rule $A \to B\ C$, and its position, including split point m, within the sentence i, such as q in figure 1(b), and corresponds to the traditional notion of a traversal in a tabular parser. Note that parts for a basic PCFG model are not just rewrites (which can occur multiple times), but rather anchored items.

Formally, we assume some countable set of parts, $\mathcal{R}$. We also assume a function R which maps each object $(x, y) \in \mathcal{X} \times \mathcal{Y}$ to a finite subset of $\mathcal{R}$. Thus $R(x, y)$ is the set of parts belonging to a particular parse. Equivalently, the function $R(x, y)$ maps a derivation y to the set of parts which it includes. Because all rules are in binary-branching form, $|R(x, y)|$ is constant across different derivations y for the same input sentence x. We assume that the feature vector for a sentence and parse tree (x, y) decomposes into a sum of the feature vectors for its parts:

$$\Phi(x, y) = \sum_{r \in R(x,y)} \phi(x, r).$$

In CFGs, the function $\phi(x, r)$ can be any function mapping a rule production and its position in the sentence x, to some feature vector representation. For example, ϕ could include features which identify the rule used in the production, or features which track the rule identity together with features of the words at positions s, m, e, and neighboring positions in the sentence x.

In addition, we assume that the loss function $L(x, y, \hat{y})$ also decomposes into a sum of local loss functions $l(x, y, r)$ over parts, as follows:

$$L(x, y, \hat{y}) = \sum_{r \in R(x,\hat{y})} l(x, y, r).$$

One approach would be to define $l(x, y, r)$ to be 0 only if the non-terminal A spans words $s \ldots e$ in the derivation y and 1 otherwise. This would lead to $L(x, y, \hat{y})$ tracking the number of "constituent errors" in $\hat{y}$, where a constituent is a tuple such as $\langle A, s, e, i\rangle$. Another, more strict definition would be to define $l(x, y, r)$ to be 0 if r of the type $\langle A \to B\ C, s, m, e, i\rangle$ is in the derivation y and 1 otherwise. This definition would lead to $L(x, y, \hat{y})$ being the number of CF-rule-tuples in $\hat{y}$ which are not seen in y.[4]

Finally, we define indicator variables $I(x, y, r)$ which are 1 if $r \in R(x, y)$, 0 otherwise. We also define sets $R(x_i) = \cup_{y \in \mathbf{G}(x_i)} R(x_i, y)$ for the training examples $i = 1 \ldots n$. Thus, $R(x_i)$ is the set of parts that is seen in at least one of the objects $\{(x_i, y) \ : \ y \in \mathbf{G}(x_i)\}$.

4 Factored Dual

The dual in Eq. 3 involves variables $\alpha_{i,y}$ for all $i = 1 \ldots n$, $y \in \mathbf{G}(x_i)$, and the objective is quadratic in these α variables. In addition, it turns out that the set of dual variables $\alpha_i = \{\alpha_{i,y} : y \in \mathbf{G}(x_i)\}$ for each example i is constrained to be non-negative and sum to 1. It is interesting that, while the parameters $\mathbf{w}$ lose their probabilistic interpretation, the dual variables α_i for each sentence actually form a kind of probability distribution. Furthermore, the objective can be expressed in terms of expectations with respect to these distributions:

$$C \sum_i \mathbf{E}_{\alpha_i}\left[L_{i,y}\right] - \frac{1}{2}\left|\left|C \sum_i \Phi_{i,y_i} - \mathbf{E}_{\alpha_i}\left[\Phi_{i,y}\right]\right|\right|^2.$$

We now consider how to efficiently solve the max-margin optimization problem for a factored model. As shown in Taskar et al. (2003), the dual in Eq. 3 can be reframed using "marginal" terms. We will also find it useful to consider this alternative formulation of the dual. Given dual variables α, we define the marginals $\mu_{i,r}(\alpha)$ for all i, r, as follows:

$$\mu_{i,r}(\alpha_i) = \sum_y \alpha_{i,y} I(x_i, y, r) = \mathbf{E}_{\alpha_i}\left[I(x_i, y, r)\right].$$

Since the dual variables α_i form probability distributions over parse trees for each sentence i, the marginals $\mu_{i,r}(\alpha_i)$ represent the proportion of parses that would contain part r if they were drawn from a distribution α_i. Note that the number of such marginal terms is the number of parts, which is polynomial in the length of the sentence.

Now consider the dual objective $Q(\alpha)$ in Eq. 3. It can be shown that the original objective $Q(\alpha)$ can be expressed in terms of these marginals as $Q_m(\mu(\alpha))$, where $\mu(\alpha)$ is the vector with components $\mu_{i,r}(\alpha_i)$, and $Q_m(\mu)$ is defined as:

$$C \sum_{i,r \in R(x_i)} \mu_{i,r} l_{i,r} - \frac{1}{2}\left|\left|C \sum_{i,r \in R(x_i)} (I_{i,r} - \mu_{i,r})\phi_{i,r}\right|\right|^2$$

where $l_{i,r} = l(x_i, y_i, r)$, $\phi_{i,r} = \phi(x_i, r)$ and $I_{i,r} = I(x_i, y_i, r)$.

This follows from substituting the factored definitions of the feature representation Φ and loss function L together with definition of marginals.

Having expressed the objective in terms of a polynomial number of variables, we now turn to the constraints on these variables. The feasible set for α is

$$\Delta = \{\alpha : \alpha_{i,y} \geq 0, \quad \forall i, y \quad \sum_y \alpha_{i,y} = 1, \quad \forall i\}.$$

Now let Δ_m be the space of marginal vectors which are feasible:

$$\Delta_m = \{\mu \ : \ \exists \alpha \in \Delta \ \text{ s.t. } \ \mu = \mu(\alpha)\}.$$

Then our original optimization problem can be reframed as $\max_{\mu \in \Delta_m} Q_m(\mu)$.

Fortunately, in case of PCFGs, the domain Δ_m can be described compactly with a polynomial number of linear constraints. Essentially, we need to enforce the condition that the expected proportions of parses having particular parts should be consistent with each other. Our marginals track constituent parts $\langle A, s, e, i\rangle$ and CF-rule-tuple parts $\langle A \rightarrow B \ C, s, m, e, i\rangle$ The consistency constraints are precisely the inside-outside probability relations:

$$\mu_{i,A,s,e} \quad = \quad \sum_{\substack{B,C \\ s<m<e}} \mu_{i,A \rightarrow B\ C,s,m,e}$$

and

$$\mu_{i,A,s,e} \quad = \quad \sum_{\substack{B,C \\ e<m\leq n_i}} \mu_{i,B \rightarrow AC} + \sum_{\substack{B,C \\ 0\leq m<s}} \mu_{i,B \rightarrow CA}$$

where n_i is the length of the sentence. In addition, we must ensure non-negativity and normalization to 1:

$$\mu_{i,r} \geq 0; \quad \sum_A \mu_{i,A,0,n_i} = 1.$$

The number of variables in our factored dual for CFGs is cubic in the length of the sentence,

[4]The constituent loss function does not exactly correspond to the standard scoring metrics, such as F_1 or crossing brackets, but shares the sensitivity to the number of differences between trees. We have not thoroughly investigated the exact interplay between the various loss choices and the various parsing metrics. We used the constituent loss in our experiments.

Model	P	R	F_1
GENERATIVE	87.70	88.06	87.88
BASIC	87.51	88.44	87.98
LEXICAL	88.15	88.62	88.39
LEXICAL+AUX	89.74	90.22	89.98

Figure 2: Development set results of the various models when trained and tested on Penn treebank sentences of length ≤ 15.

Model	P	R	F_1
GENERATIVE	88.25	87.73	87.99
BASIC	88.08	88.31	88.20
LEXICAL	88.55	88.34	88.44
LEXICAL+AUX	89.14	**89.10**	**89.12**
COLLINS 99	**89.18**	88.20	88.69

Figure 3: Test set results of the various models when trained and tested on Penn treebank sentences of length ≤ 15.

while the number of constraints is quadratic. This polynomial size formulation should be contrasted with the earlier formulation in Collins (2004), which has an exponential number of constraints.

5 Factored SMO

We have reduced the problem to a polynomial size QP, which, in principle, can be solved using standard QP toolkits. However, although the number of variables and constraints in the factored dual is polynomial in the size of the data, the number of coefficients in the quadratic term in the objective is very large: quadratic in the number of sentences and dependent on the sixth power of sentence length. Hence, in our experiments we use an online coordinate descent method analogous to the sequential minimal optimization (SMO) used for SVMs (Platt, 1999) and adapted to structured max-margin estimation in Taskar et al. (2003).

We omit the details of the structured SMO procedure, but the important fact about this kind of training is that, similar to the basic perceptron approach, it only requires picking up sentences one at a time, checking what the best parse is according to the current primal and dual weights, and adjusting the weights.

6 Results

We used the Penn English Treebank for all of our experiments. We report results here for each model and setting trained and tested on only the sentences of length ≤ 15 words. Aside from the length restriction, we used the standard splits: sections 2-21 for training (9753 sentences), 22 for development (603 sentences), and 23 for final testing (421 sentences).

As a baseline, we trained a CNF transformation of the unlexicalized model of Klein and Manning (2003) on this data. The resulting grammar had 3975 non-terminal symbols and contained two kinds of productions: binary nonterminal rewrites and tag-word rewrites.[5] The scores for the binary rewrites were estimated using unsmoothed relative frequency estimators. The tagging rewrites were estimated with a smoothed model of $P(w|t)$, also using the model from Klein and Manning (2003). Figure 3 shows the performance of this model (GENERATIVE): 87.99 F_1 on the test set.

For the BASIC max-margin model, we used exactly the same set of allowed rewrites (and therefore the same set of candidate parses) as in the generative case, but estimated their weights according to the discriminative method of section 4. Tag-word production weights were fixed to be the log of the generative $P(w|t)$ model. That is, the only change between GENERATIVE and BASIC is the use of the discriminative maximum-margin criterion in place of the generative maximum likelihood one. This change alone results in a small improvement (88.20 vs. 87.99 F_1).

On top of the basic model, we first added lexical features of each span; this gave a LEXICAL model. For a span $\langle s, e\rangle$ of a sentence x, the base lexical features were:

- x_s, the first word in the span
- x_{s-1}, the preceding adjacent word
- x_{e-1}, the last word in the span
- x_e, the following adjacent word
- $\langle x_{s-1}, x_s\rangle$
- $\langle x_{e-1}, x_e\rangle$
- x_{s+1} for spans of length 3

These base features were conjoined with the span length for spans of length 3 and below, since short spans have highly distinct behaviors (see the examples below). The features are lexical in the sense than they allow specific words

[5] Unary rewrites were compiled into a single compound symbol, so for example a subject-gapped sentence would have label like S+VP. These symbols were expanded back into their source unary chain before parses were evaluated.

and word pairs to influence the parse scores, but are distinct from traditional lexical features in several ways. First, there is no notion of headword here, nor is there any modeling of word-to-word attachment. Rather, these features pick up on lexical trends in constituent boundaries, for example the trend that in the sentence *The screen was a sea of red.*, the (length 2) span between the word *was* and the word *of* is unlikely to be a constituent. These non-head lexical features capture a potentially very different source of constraint on tree structures than head-argument pairs, one having to do more with linear syntactic preferences than lexical selection. Regardless of the relative merit of the two kinds of information, one clear advantage of the present approach is that inference in the resulting model remains cubic, since the dynamic program need not track items with distinguished headwords. With the addition of these features, the accuracy jumped past the generative baseline, to 88.44.

As a concrete (and particularly clean) example of how these features can sway a decision, consider the sentence *The Egyptian president said he would visit Libya today to resume the talks.* The generative model incorrectly considers *Libya today* to be a base NP. However, this analysis is counter to the trend of *today* being a one-word constituent. Two features relevant to this trend are: (CONSTITUENT $\wedge$ first-word = *today* $\wedge$ length = 1) and (CONSTITUENT $\wedge$ last-word = *today* $\wedge$ length = 1). These features represent the preference of the word *today* for being the first and and last word in constituent spans of length 1.[6] In the LEXICAL model, however, these features have quite large positive weights: 0.62 each. As a result, this model makes this parse decision correctly.

Another kind of feature that can usefully be incorporated into the classification process is the output of other, auxiliary classifiers. For this kind of feature, one must take care that its reliability on the training not be vastly greater than its reliability on the test set. Otherwise, its weight will be artificially (and detrimentally) high. To ensure that such features are as noisy on the training data as the test data, we split the training into two folds. We then trained the auxiliary classifiers in jacknife fashion on each fold, and using their predictions as features on the other fold. The auxiliary classifiers were then retrained on the entire training set, and their predictions used as features on the development and test sets.

We used two such auxiliary classifiers, giving a prediction feature for each span (these classifiers predicted only the presence or absence of a bracket over that span, not bracket labels). The first feature was the prediction of the generative baseline; this feature added little information, but made the learning phase faster. The second feature was the output of a flat classifier which was trained to predict whether single spans, in isolation, were constituents or not, based on a bundle of features including the list above, but also the following: the preceding, first, last, and following tag in the span, pairs of tags such as preceding-first, last-following, preceding-following, first-last, and the entire tag sequence.

Tag features on the test sets were taken from a pretagging of the sentence by the tagger described in Toutanova et al. (2003). While the flat classifier alone was quite poor (P 78.77 / R 63.94 / F_1 70.58), the resulting max-margin model (LEXICAL+AUX) scored 89.12 F_1. To situate these numbers with respect to other models, the parser in Collins (1999), which is generative, lexicalized, and intricately smoothed scores 88.69 over the same train/test configuration.

It is worth considering the cost of this kind of method. At training time, discriminative methods are inherently expensive, since they all involve iteratively checking current model performance on the training set, which means parsing the training set (usually many times). In our experiments, 10-20 iterations were generally required for convergence (except the BASIC model, which took about 100 iterations.) There are several nice aspects of the approach described here. First, it is driven by the repeated extraction, over the training examples, of incorrect parses which the model currently prefers over the true parses. The procedure that provides these parses need not sum over all parses, nor even necessarily find the Viterbi parses, to function. This allows a range of optimizations not possible for CRF-like approaches which must extract feature expectations from the entire set of parses.[7] Nonetheless, generative approaches

[6]In this length 1 case, these are the same feature. Note also that the features are conjoined with only one generic label class "constituent" rather than specific constituent types.

[7]One tradeoff is that this approach is more inherently sequential and harder to parallelize.

are vastly cheaper to train, since they must only collect counts from the training set.

On the other hand, the max-margin approach does have the potential to incorporate many new kinds of features over the input, and the current feature set allows limited lexicalization in cubic time, unlike other lexicalized models (including the Collins model which it outperforms in the present limited experiments).

7 Conclusion

We have presented a maximum-margin approach to parsing, which allows a discriminative SVM-like objective to be applied to the parsing problem. Our framework permits the use of a rich variety of input features, while still decomposing in a way that exploits the shared substructure of parse trees in the standard way. On a test set of ≤ 15 word sentences, the feature-rich model outperforms both its own natural generative baseline and the Collins parser on F_1. While like most discriminative models it is compute-intensive to train, it allows fast parsing, remaining cubic despite the incorporation of lexical features. This trade-off between the complexity, accuracy and efficiency of a parsing model is an important area of future research.

Acknowledgements

This work was supported in part by the Department of the Interior/DARPA under contract number NBCHD030010, a Microsoft Graduate Fellowship to the second author, and National Science Foundation grant 0347631 to the third author.

References

Y. Altun, I. Tsochantaridis, and T. Hofmann. 2003. Hidden markov support vector machines. In *Proc. ICML*.

S. Clark and J. R. Curran. 2004. Parsing the wsj using ccg and log-linear models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL '04)*.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

M. Collins. 2000. Discriminative reranking for natural language parsing. In *ICML 17*, pages 175–182.

M. Collins. 2004. Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In H. Bunt, J. Carroll, and G. Satta, editors, *New Developments in Parsing Technology*. Kluwer.

N. Cristianini and J. Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press.

S. Geman and M. Johnson. 2002. Dynamic programming for parsing and estimation of stochastic unification-based grammars. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

M. Johnson, S. Geman, S. Canon, Z. Chi, and S. Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proceedings of ACL 1999*.

M. Johnson. 2001. Joint and conditional estimation of tagging and parsing models. In *ACL 39*.

R. Kaplan, S. Riezler, T. King, J. Maxwell, A. Vasserman, and R. Crouch. 2004. Speed and accuracy in shallow and deep stochastic parsing. In *Proceedings of HLT-NAACL'04*.

D. Klein and C. D. Manning. 2003. Accurate unlexicalized parsing. In *ACL 41*, pages 423–430.

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

Y. Miyao and J. Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proceedings of Human Language Technology Conference (HLT 2002)*.

J. Platt. 1999. Using sparseness and analytic QP to speed training of support vector machines. In *NIPS*.

L. Shen, A. Sarkar, and A. K. Joshi. 2003. Using ltag based features in parse reranking. In *Proc. EMNLP*.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max margin Markov networks. In *NIPS*.

K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL 3*, pages 252–259.