Data-Defined Kernels for Parse Reranking Derived from Probabilistic Models

James Henderson
School of Informatics
University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW, United Kingdom
james.henderson@ed.ac.uk

Ivan Titov
Department of Computer Science
University of Geneva
24, rue Général Dufour
CH-1211 Genève 4, Switzerland
ivan.titov@cui.unige.ch

Abstract

Previous research applying kernel methods to natural language parsing have focussed on proposing kernels over parse trees, which are hand-crafted based on domain knowledge and computational considerations. In this paper we propose a method for defining kernels in terms of a probabilistic model of parsing. This model is then trained, so that the parameters of the probabilistic model reflect the generalizations in the training data. The method we propose then uses these trained parameters to define a kernel for reranking parse trees. In experiments, we use a neural network based statistical parser as the probabilistic model, and use the resulting kernel with the Voted Perceptron algorithm to rerank the top 20 parses from the probabilistic model. This method achieves a significant improvement over the accuracy of the probabilistic model.

1 Introduction

Kernel methods have been shown to be very effective in many machine learning problems. They have the advantage that learning can try to optimize measures related directly to expected testing performance (i.e. "large margin" methods), rather than the probabilistic measures used in statistical models, which are only indirectly related to expected testing performance. Work on kernel methods in natural language has focussed on the definition of appropriate kernels for natural language tasks. In particular, most of the work on parsing with kernel methods has focussed on kernels over parse trees (Collins and Duffy, 2002; Shen and Joshi, 2003; Shen et al., 2003; Collins and Roark, 2004). These kernels have all been hand-crafted to try reflect properties of parse trees which are relevant to discriminating correct parse trees from incorrect ones, while at the same time maintaining the tractability of learning.

Some work in machine learning has taken an alternative approach to defining kernels, where the kernel is derived from a probabilistic model of the task (Jaakkola and Haussler, 1998; Tsuda et al., 2002). This way of defining kernels has two advantages. First, linguistic knowledge about parsing is reflected in the design of the probabilistic model, not directly in the kernel. Designing probabilistic models to reflect linguistic knowledge is a process which is currently well understood, both in terms of reflecting generalizations and controlling computational cost. Because many NLP problems are unbounded in size and complexity, it is hard to specify all possible relevant kernel features without having so many features that the computations become intractable and/or the data becomes too sparse.[1] Second, the kernel is defined using the trained parameters of the probabilistic model. Thus the kernel is in part determined by the training data, and is automatically tailored to reflect properties of parse trees which are relevant to parsing.

[1] For example, see (Henderson, 2004) for a discussion of why generative models are better than models parameterized to estimate the a posteriori probability directly.

Proceedings of the 43rd Annual Meeting of the ACL, pages 181–188,
Ann Arbor, June 2005.

In this paper, we propose a new method for deriving a kernel from a probabilistic model which is specifically tailored to reranking tasks, and we apply this method to natural language parsing. For the probabilistic model, we use a state-of-the-art neural network based statistical parser (Henderson, 2003). The resulting kernel is then used with the Voted Perceptron algorithm (Freund and Schapire, 1998) to reranking the top 20 parses from the probabilistic model. This method achieves a significant improvement over the accuracy of the probabilistic model alone.

2 Kernels Derived from Probabilistic Models

In recent years, several methods have been proposed for constructing kernels from trained probabilistic models. As usual, these kernels are then used with linear classifiers to learn the desired task. As well as some empirical successes, these methods are motivated by theoretical results which suggest we should expect some improvement with these classifiers over the classifier which chooses the most probable answer according to the probabilistic model (i.e. the maximum a posteriori (MAP) classifier). There is guaranteed to be a linear classifier for the derived kernel which performs at least as well as the MAP classifier for the probabilistic model. So, assuming a large-margin classifier can optimize a more appropriate criteria than the posterior probability, we should expect the derived kernel's classifier to perform better than the probabilistic model's classifier, although empirical results on a given task are never guaranteed.

In this section, we first present two previous kernels and then propose a new kernel specifically for reranking tasks. In each of these discussions we need to characterize the parsing problem as a classification task. Parsing can be regarded as a mapping from an input space of sentences $x \in \mathcal{X}$ to a structured output space of parse trees $y \in \mathcal{Y}$. On the basis of training sentences, we learn a discriminant function $F : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{R}$. The parse tree y with the largest value for this discriminant function $F(x, y)$ is the output parse tree for the sentence x. We focus on the linear discriminant functions:

$$F_w(x, y) = <w, \phi(x, y)>,$$

where $\phi(x, y)$ is a feature vector for the sentence-tree pair, w is a parameter vector for the discriminant function, and $<a, b>$ is the inner product of vectors a and b. In the remainder of this section, we will characterize the kernel methods we consider in terms of the feature extractor $\phi(x, y)$.

2.1 Fisher Kernels

The Fisher kernel (Jaakkola and Haussler, 1998) is one of the best known kernels belonging to the class of probability model based kernels. Given a generative model of $P(z|\hat{\theta})$ with smooth parameterization, the Fisher score of an example z is a vector of partial derivatives of the log-likelihood of the example with respect to the model parameters:

$$\phi_{\hat{\theta}}(z) = (\frac{\partial \log P(z|\hat{\theta})}{\partial \theta_1}, \ldots, \frac{\partial \log P(z|\hat{\theta})}{\partial \theta_l}).$$

This score can be regarded as specifying how the model should be changed in order to maximize the likelihood of the example z. Then we can define the similarity between data points as the inner product of the corresponding Fisher scores. This kernel is often referred to as the practical Fisher kernel. The theoretical Fisher kernel depends on the Fisher information matrix, which is not feasible to compute for most practical tasks and is usually omitted.

The Fisher kernel is only directly applicable to binary classification tasks. We can apply it to our task by considering an example z to be a sentence-tree pair (x, y), and classifying the pairs into correct parses versus incorrect parses. When we use the Fisher score $\phi_{\hat{\theta}}(x, y)$ in the discriminant function F, we can interpret the value as the confidence that the tree y is correct, and choose the y in which we are the most confident.

2.2 TOP Kernels

Tsuda (2002) proposed another kernel constructed from a probabilistic model, called the Tangent vectors Of Posterior log-odds (TOP) kernel. Their TOP kernel is also only for binary classification tasks, so, as above, we treat the input z as a sentence-tree pair and the output category $c \in \{-1, +1\}$ as incorrect/correct. It is assumed that the true probability distribution is included in the class of probabilistic models and that the true parameter vector $\theta^\star$ is unique. The feature extractor of the TOP kernel for

the input z is defined by:

$$\phi_{\hat{\theta}}(z) = (v(z,\hat{\theta}), \frac{\partial v(z,\hat{\theta})}{\partial \theta_1}, \ldots, \frac{\partial v(z,\hat{\theta})}{\partial \theta_l}),$$

where $v(z,\hat{\theta}) = \log P(c{=}{+}1|z,\hat{\theta}) - \log P(c{=}{-}1|z,\hat{\theta})$.

In addition to being at least as good as the MAP classifier, the choice of the TOP kernel feature extractor is motivated by the minimization of the binary classification error of a linear classifier $<w, \phi_{\hat{\theta}}(z)> + b$. Tsuda (2002) demonstrates that this error is closely related to the estimation error of the posterior probability $P(c{=}{+}1|z,\theta^\star)$ by the estimator $g(<w, \phi_{\hat{\theta}}(z)> + b)$, where g is the sigmoid function $g(t) = 1/(1+\exp(-t))$.

The TOP kernel isn't quite appropriate for structured classification tasks because $\phi_{\hat{\theta}}(z)$ is motivated by binary classificaton error minimization. In the next subsection, we will adapt it to structured classification.

2.3 A TOP Kernel for Reranking

We define the reranking task as selecting a parse tree from the list of candidate trees suggested by a probabilistic model. Furthermore, we only consider learning to rerank the output of a particular probabilistic model, without requiring the classifier to have good performance when applied to a candidate list provided by a different model. In this case, it is natural to model the probability that a parse tree is the best candidate given the list of candidate trees:

$$P(y_k|x, y_1, \ldots, y_s) = \frac{P(x,y_k)}{\sum_t P(x,y_t)},$$

where $y_1, \ldots, y_s$ is the list of candidate parse trees.

To construct a new TOP kernel for reranking, we apply an approach similar to that used for the TOP kernel (Tsuda et al., 2002), but we consider the probability $P(y_k|x, y_1, \ldots, y_s, \theta^\star)$ instead of the probability $P(c{=}{+}1|z,\theta^\star)$ considered by Tsuda. The resulting feature extractor is given by:

$$\phi_{\hat{\theta}}(x, y_k) = (v(x, y_k, \hat{\theta}), \frac{\partial v(x,y_k,\hat{\theta})}{\partial \theta_1}, \ldots, \frac{\partial v(x,y_k,\hat{\theta})}{\partial \theta_l}),$$

where $v(x, y_k, \hat{\theta}) = \log P(y_k|y_1, \ldots, y_s, \hat{\theta}) - \log \sum_{t \neq k} P(y_t|y_1, \ldots, y_s, \hat{\theta})$. We will call this kernel the *TOP reranking kernel*.

3 The Probabilistic Model

To complete the definition of the kernel, we need to choose a probabilistic model of parsing. For this we use a statistical parser which has previously been shown to achieve state-of-the-art performance, namely that proposed in (Henderson, 2003). This parser has two levels of parameterization. The first level of parameterization is in terms of a history-based generative probability model, but this level is not appropriate for our purposes because it defines an infinite number of parameters (one for every possible partial parse history). When parsing a given sentence, the bounded set of parameters which are relevant to a given parse are estimated using a neural network. The weights of this neural network form the second level of parameterization. There is a finite number of these parameters. Neural network training is applied to determine the values of these parameters, which in turn determine the values of the probability model's parameters, which in turn determine the probabilistic model of parse trees.

We do not use the complete set of neural network weights to define our kernels, but instead we define a third level of parameterization which only includes the network's output layer weights. These weights define a normalized exponential model, with the network's hidden layer as the input features. When we tried using the complete set of weights in some small scale experiments, training the classifier was more computationally expensive, and actually performed slightly worse than just using the output weights. Using just the output weights also allows us to make some approximations in the TOP reranking kernel which makes the classifier learning algorithm more efficient.

3.1 A History-Based Probability Model

As with many other statistical parsers (Ratnaparkhi, 1999; Collins, 1999; Charniak, 2000), Henderson (2003) uses a history-based model of parsing. He defines the mapping from phrase structure trees to parse sequences using a form of left-corner parsing strategy (see (Henderson, 2003) for more details). The parser actions include: introducing a new constituent with a specified label, attaching one constituent to another, and predicting the next word of the sentence. A complete parse consists of a sequence of these actions, $d_1,\ldots,d_m$, such that performing $d_1,\ldots,d_m$ results in a complete phrase structure tree.

Because this mapping to parse sequences is

one-to-one, and the word prediction actions in a complete parse $d_1,...,d_m$ specify the sentence, $P(d_1,...,d_m)$ is equivalent to the joint probability of the output phrase structure tree and the input sentence. This probability can be then be decomposed into the multiplication of the probabilities of each action decision d_i conditioned on that decision's prior parse history $d_1,...,d_{i-1}$.

$$P(d_1,...,d_m) = \Pi_i P(d_i|d_1,...,d_{i-1})$$

3.2 Estimating Decision Probabilities with a Neural Network

The parameters of the above probability model are the $P(d_i|d_1,...,d_{i-1})$. There are an infinite number of these parameters, since the parse history $d_1,...,d_{i-1}$ grows with the length of the sentence. In other work on history-based parsing, independence assumptions are applied so that only a finite amount of information from the parse history can be treated as relevant to each parameter, thereby reducing the number of parameters to a finite set which can be estimated directly. Instead, Henderson (2003) uses a neural network to induce a finite representation of this unbounded history, which we will denote $h(d_1,...,d_{i-1})$. Neural network training tries to find such a history representation which preserves all the information about the history which is relevant to estimating the desired probability.

$$P(d_i|d_1,...,d_{i-1}) \approx P(d_i|h(d_1,...,d_{i-1}))$$

Using a neural network architecture called Simple Synchrony Networks (SSNs), the history representation $h(d_1,...,d_{i-1})$ is incrementally computed from features of the previous decision d_{i-1} plus a finite set of previous history representations $h(d_1,...,d_j)$, $j < i-1$. Each history representation is a finite vector of real numbers, called the network's hidden layer. As long as the history representation for position $i-1$ is always included in the inputs to the history representation for position i, any information about the entire sequence could be passed from history representation to history representation and be used to estimate the desired probability. However, learning is biased towards paying more attention to information which passes through fewer history representations.

To exploit this learning bias, structural locality is used to determine which history representations are input to which others. First, each history representation is assigned to the constituent which is on the top of the parser's stack when it is computed. Then earlier history representations whose constituents are structurally local to the current representation's constituent are input to the computation of the correct representation. In this way, the number of representations which information needs to pass through in order to flow from history representation i to history representation j is determined by the structural distance between i's constituent and j's constituent, and not just the distance between i and j in the parse sequence. This provides the neural network with a linguistically appropriate inductive bias when it learns the history representations, as explained in more detail in (Henderson, 2003).

Once it has computed $h(d_1,...,d_{i-1})$, the SSN uses a normalized exponential to estimate a probability distribution over the set of possible next decisions d_i given the history:

$$P(d_i|d_1,...,d_{i-1},\theta) \approx \frac{exp(<\theta_{d_i},h(d_1,..,d_{i-1})>)}{\sum_{t\in N(d_{i-1})} exp(<\theta_t,h(d_1,..,d_{i-1})>)},$$

where by θ_t we denote the set of output layer weights, corresponding to the parser action t, $N(d_{i-1})$ defines a set of possible next parser actions after the step d_{i-1} and θ denotes the full set of model parameters.

We trained SSN parsing models, using the on-line version of Backpropagation to perform the gradient descent with a maximum likelihood objective function. This learning simultaneously tries to optimize the parameters of the output computation and the parameters of the mappings $h(d_1,...,d_{i-1})$. With multi-layered networks such as SSNs, this training is not guaranteed to converge to a global optimum, but in practice a network whose criteria value is close to the optimum can be found.

4 Large-Margin Optimization

Once we have defined a kernel over parse trees, general techniques for linear classifier optimization can be used to learn the given task. The most sophisticated of these techniques (such as Support Vector Machines) are unfortunately too computationally expensive to be used on large datasets like the Penn Treebank (Marcus et al., 1993). Instead we use a

method which has often been shown to be virtually as good, the Voted Perceptron (VP) (Freund and Schapire, 1998) algorithm. The VP algorithm was originally applied to parse reranking in (Collins and Duffy, 2002) with the Tree kernel. We modify the perceptron training algorithm to make it more suitable for parsing, where zero-one classification loss is not the evaluation measure usually employed. We also develop a variant of the kernel defined in section 2.3, which is more efficient when used with the VP algorithm.

Given a list of candidate trees, we train the classifier to select the tree with largest constituent F_1 score. The F_1 score is a measure of the similarity between the tree in question and the gold standard parse, and is the standard way to evaluate the accuracy of a parser. We denote the *k*'th candidate tree for the *j*'th sentence x^j by y_k^j. Without loss of generality, let us assume that y_1^j is the candidate tree with the largest F_1 score.

The Voted Perceptron algorithm is an ensemble method for combining the various intermediate models which are produced during training a perceptron. It demonstrates more stable generalization performance than the normal perceptron algorithm when the problem is not linearly separable (Freund and Schapire, 1998), as is usually the case.

We modify the perceptron algorithm by introducing a new classification loss function. This modification enables us to treat differently the cases where the perceptron predicts a tree with an F_1 score much smaller than that of the top candidate and the cases where the predicted and the top candidates have similar score values. The natural choice for the loss function would be $\Delta(y_k^j, y_1^j) = F_1(y_1^j) - F_1(y_k^j)$, where $F_1(y_k^j)$ denotes the F_1 score value for the parse tree y_k^j. This approach is very similar to slack variable rescaling for Support Vector Machines proposed in (Tsochantaridis et al., 2004). The learning algorithm we employed is presented in figure 1.

When applying kernels with a large training corpus, we face efficiency issues because of the large number of the neural network weights. Even though we use only the output layer weights, this vector grows with the size of the vocabulary, and thus can be large. The kernels presented in section 2 all lead to feature vectors without many zero values. This

w = 0
for j = 1 .. n
  for k = 2 .. s
    if $<w, \phi(x^j, y_k^j)>$ > $<w, \phi(x^j, y_1^j)>$
      w = w + $\Delta(y_k^j, y_1^j)(\phi(x^j, y_1^j) - \phi(x^j, y_k^j))$

Figure 1: The modified perceptron algorithm

happens because we compute the derivative of the normalization factor used in the network's estimation of $P(d_i|d_1,...,d_{i-1})$. This normalization factor depends on the output layer weights corresponding to all the possible next decisions (see section 3.2). This makes an application of the VP algorithm infeasible in the case of a large vocabulary.

We can address this problem by freezing the normalization factor when computing the feature vector. Note that we can rewrite the model log-probability of the tree as:

$$\log P(y|\theta) = \sum_i \log\left(\frac{exp(<\theta_{d_i}, h(d_1,..,d_{i-1})>)}{\sum_{t\in N(d_{i-1})} exp(<\theta_t, h(d_1,..,d_{i-1})>)}\right) = \sum_i (<\theta_{d_i}, h(d_1,...,d_{i-1})>) - \sum_i \log \sum_{t\in N(d_{i-1})} exp(<\theta_t, h(d_1,...,d_{i-1})>).$$

We treat the parameters used to compute the first term as different from the parameters used to compute the second term, and we define our kernel only using the parameters in the first term. This means that the second term does not effect the derivatives in the formula for the feature vector $\phi(x, y)$. Thus the feature vector for the kernel will contain non-zero entries only in the components corresponding to the parser actions which are present in the candidate derivation for the sentence, and thus in the first vector component. We have applied this technique to the TOP reranking kernel, the result of which we will call the *efficient TOP reranking kernel*.

5 The Experimental Results

We used the Penn Treebank WSJ corpus (Marcus et al., 1993) to perform empirical experiments on the proposed parsing models. In each case the input to the network is a sequence of tag-word pairs.[2] We report results for two different vocabulary sizes, varying in the frequency with which tag-word pairs must

[2]We used a publicly available tagger (Ratnaparkhi, 1996) to provide the tags.

occur in the training set in order to be included explicitly in the vocabulary. A frequency threshold of 200 resulted in a vocabulary of 508 tag-word pairs (including tag-unknown_word pairs) and a threshold of 20 resulted in 4215 tag-word pairs. We denote the probabilistic model trained with the vocabulary of 508 by the SSN-Freq≥200, the model trained with the vocabulary of 4215 by the SSN-Freq≥20.

Testing the probabilistic parser requires using a beam search through the space of possible parses. We used a form of beam search which prunes the search after the prediction of each word. We set the width of this post-word beam to 40 for both testing of the probabilistic model and generating the candidate list for reranking. For training and testing of the kernel models, we provided a candidate list consisting of the top 20 parses found by the generative probabilistic model. When using the Fisher kernel, we added the log-probability of the tree given by the probabilistic model as the feature. This was not necessary for the TOP kernels because they already contain a feature corresponding to the probability estimated by the probabilistic model (see section 2.3).

We trained the VP model with all three kernels using the 508 word vocabulary (Fisher-Freq≥200, TOP-Freq≥200, TOP-Eff-Freq≥200) but only the efficient TOP reranking kernel model was trained with the vocabulary of 4215 words (TOP-Eff-Freq≥20). The non-sparsity of the feature vectors for other kernels led to the excessive memory requirements and larger testing time. In each case, the VP model was run for only one epoch. We would expect some improvement if running it for more epochs, as has been empirically demonstrated in other domains (Freund and Schapire, 1998).

To avoid repeated testing on the standard testing set, we first compare the different models with their performance on the validation set. Note that the validation set wasn't used during learning of the kernel models or for adjustment of any parameters.

Standard measures of accuracy are shown in table 1.[3] Both the Fisher kernel and the TOP kernels show better accuracy than the baseline probabilistic model, but only the improvement of the TOP kernels is statistically significant.[4] For the TOP kernel, the improvement over baseline is about the same with both vocabulary sizes. Also note that the performance of the efficient TOP reranking kernel is the same as that of the original TOP reranking kernel, for the smaller vocabulary.

	LR	LP	$F_{\beta=1}$
SSN-Freq≥200	87.2	88.5	87.8
Fisher-Freq≥200	87.2	88.8	87.9
TOP-Freq≥200	87.3	88.9	88.1
TOP-Eff-Freq≥200	87.3	88.9	88.1
SSN-Freq≥20	88.1	89.2	88.6
TOP-Eff-Freq≥20	88.2	89.7	88.9

Table 1: Percentage labeled constituent recall (LR), precision (LP), and a combination of both ($F_{\beta=1}$) on validation set sentences of length at most 100.

For comparison to previous results, table 2 lists the results on the testing set for our best model (TOP-Efficient-Freq≥20) and several other statistical parsers (Collins, 1999; Collins and Duffy, 2002; Collins and Roark, 2004; Henderson, 2003; Charniak, 2000; Collins, 2000; Shen and Joshi, 2004; Shen et al., 2003; Henderson, 2004; Bod, 2003). First note that the parser based on the TOP efficient kernel has better accuracy than (Henderson, 2003), which used the same parsing method as our baseline model, although the trained network parameters were not the same. When compared to other kernel methods, our approach performs better than those based on the Tree kernel (Collins and Duffy, 2002; Collins and Roark, 2004), and is only 0.2% worse than the best results achieved by a kernel method for parsing (Shen et al., 2003; Shen and Joshi, 2004).

6 Related Work

The first application of kernel methods to parsing was proposed by Collins and Duffy (2002). They used the Tree kernel, where the features of a tree are all its connected tree fragments. The VP algorithm was applied to rerank the output of a probabilistic model and demonstrated an improvement over the baseline.

[3] All our results are computed with the evalb program following the standard criteria in (Collins, 1999), and using the standard training (sections 2–22, 39,832 sentences, 910,196 words), validation (section 24, 1346 sentence, 31507 words), and testing (section 23, 2416 sentences, 54268 words) sets (Collins, 1999).

[4] We measured significance with the randomized significance test of (Yeh, 2000).

	LR	LP	$F_{\beta=1*}$
Collins99	88.1	88.3	88.2
Collins&Duffy02	88.6	88.9	88.7
Collins&Roark04	88.4	89.1	88.8
Henderson03	88.8	89.5	89.1
Charniak00	89.6	89.5	89.5
TOP-Eff-Freq≥20	89.1	90.1	89.6
Collins00	89.6	89.9	89.7
Shen&Joshi04	89.5	90.0	89.8
Shen et al.03	89.7	90.0	89.8
Henderson04	89.8	90.4	90.1
Bod03	90.7	90.8	90.7

* $F_{\beta=1}$ for previous models may have rounding errors.

Table 2: Percentage labeled constituent recall (LR), precision (LP), and a combination of both ($F_{\beta=1}$) on the entire testing set.

Shen and Joshi (2003) applied an SVM based voting algorithm with the Preference kernel defined over pairs for reranking. To define the Preference kernel they used the Tree kernel and the Linear kernel as its underlying kernels and achieved state-of-the-art results with the Linear kernel.

In (Shen et al., 2003) it was pointed out that most of the arbitrary tree fragments allowed by the Tree kernel are linguistically meaningless. The authors suggested the use of Lexical Tree Adjoining Grammar (LTAG) based features as a more linguistically appropriate set of features. They empirically demonstrated that incorporation of these features helps to improve reranking performance.

Shen and Joshi (2004) proposed to improve margin based methods for reranking by defining the margin not only between the top tree and all the other trees in the candidate list but between all the pairs of parses in the ordered candidate list for the given sentence. They achieved the best results when training with an uneven margin scaled by the heuristic function of the candidates positions in the list. One potential drawback of this method is that it doesn't take into account the actual F_1 score of the candidate and considers only the position in the list ordered by the F_1 score. We expect that an improvement could be achieved by combining our approach of scaling updates by the F_1 loss with the all pairs approach of (Shen and Joshi, 2004). Use of the F_1 loss function during training demonstrated better performance comparing to the 0-1 loss function when applied to a structured classification task (Tsochantaridis et al., 2004).

All the described kernel methods are limited to the reranking of candidates from an existing parser due to the complexity of finding the best parse given a kernel (i.e. the decoding problem). (Taskar et al., 2004) suggested a method for maximal margin parsing which employs the dynamic programming approach to decoding and parameter estimation problems. The efficiency of dynamic programming means that the entire space of parses can be considered, not just a candidate list. However, not all kernels are suitable for this method. The dynamic programming approach requires the feature vector of a tree to be decomposable into a sum over parts of the tree. In particular, this is impossible with the TOP and Fisher kernels derived from the SSN model. Also, it isn't clear whether the algorithm remains tractable for a large training set with long sentences, since the authors only present results for sentences of length less than or equal to 15.

7 Conclusions

This paper proposes a method for deriving a kernel for reranking from a probabilistic model, and demonstrates state-of-the-art accuracy when this method is applied to parse reranking. Contrary to most of the previous research on kernel methods in parsing, linguistic knowledge does not have to be expressed through a list of features, but instead can be expressed through the design of a probability model. The parameters of this probability model are then trained, so that they reflect what features of trees are relevant to parsing. The kernel is then derived from this trained model in such a way as to maximize its usefulness for reranking.

We performed experiments on parse reranking using a neural network based statistical parser as both the probabilistic model and the source of the list of candidate parses. We used a modification of the Voted Perceptron algorithm to perform reranking with the kernel. The results were amongst the best current statistical parsers, and only 0.2% worse than the best current parsing methods which use kernels. We would expect further improvement if we used different models to derive the kernel and to gener-

ate the candidates, thereby exploiting the advantages of combining multiple models, as do the better performing methods using kernels.

In recent years, probabilistic models have become commonplace in natural language processing. We believe that this approach to defining kernels would simplify the problem of defining kernels for these tasks, and could be very useful for many of them. In particular, maximum entropy models also use a normalized exponential function to estimate probabilities, so all the methods discussed in this paper would be applicable to maximum entropy models. This approach would be particularly useful for tasks where there is less data available than in parsing, for which large-margin methods work particularly well.

References

Rens Bod. 2003. An efficient implementation of a new DOP model. In *Proc. 10th Conf. of European Chapter of the Association for Computational Linguistics*, Budapest, Hungary.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. 1st Meeting of North American Chapter of Association for Computational Linguistics*, pages 132–139, Seattle, Washington.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures and the voted perceptron. In *Proc. 40th Meeting of Association for Computational Linguistics*, pages 263–270.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proc. 42th Meeting of Association for Computational Linguistics*, Barcelona, Spain.

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

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proc. 17th Int. Conf. on Machine Learning*, pages 175–182, Stanford, CA.

Yoav Freund and Robert E. Schapire. 1998. Large margin classification using the perceptron algorithm. In *Proc. of the 11th Annual Conf. on Computational Learning Theory*, pages 209–217, Madisson WI.

James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proc. joint meeting of North American Chapter of the Association for Computational Linguistics and the Human Language Technology Conf.*, pages 103–110, Edmonton, Canada.

James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proc. 42nd Meeting of Association for Computational Linguistics*, Barcelona, Spain.

Tommi S. Jaakkola and David Haussler. 1998. Exploiting generative models in discriminative classifiers. *Advances in Neural Information Processes Systems 11*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proc. Conf. on Empirical Methods in Natural Language Processing*, pages 133–142, Univ. of Pennsylvania, PA.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34:151–175.

Libin Shen and Aravind K. Joshi. 2003. An SVM based voting algorithm with application to parse reranking. In *Proc. of the 7th Conf. on Computational Natural Language Learning*, pages 9–16, Edmonton, Canada.

Libin Shen and Aravind K. Joshi. 2004. Flexible margin selection for reranking with full pairwise samples. In *Proc. of the 1st Int. Joint Conf. on Natural Language Processing*, Hainan Island, China.

Libin Shen, Anoop Sarkar, and Aravind K. Joshi. 2003. Using LTAG based features in parse reranking. In *Proc. of Conf. on Empirical Methods in Natural Language Processing*, Sapporo, Japan.

Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Christopher Manning. 2004. Max-margin parsing. In *Proc. Conf. on Empirical Methods in Natural Language Processing*, Barcelona, Spain.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. 21st Int. Conf. on Machine Learning*, pages 823–830, Banff, Alberta, Canada.

K. Tsuda, M. Kawanabe, G. Ratsch, S. Sonnenburg, and K. Muller. 2002. A new discriminative kernel from probabilistic models. *Neural Computation*, 14(10):2397–2414.

Alexander Yeh. 2000. More accurate tests for the statistical significance of the result differences. In *Proc. 17th International Conf. on Computational Linguistics*, pages 947–953, Saarbruken, Germany.