UNIVERSITE DE GENEVE
CENTRE UNIVERSITAIRE
D'INFORMATIQUE
ARTIFICIAL INTELLIGENCE
LABORATORY

Date: June, 2006

TECHNICAL REPORT

Bayes Risk Minimization in Natural Language Parsing

Ivan Titov
University of Geneva
24 rue Général Dufour
CH-1211 Genève 4, Switzerland

James Henderson
University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW, UK

e-mail: ivan.titov@cui.unige.ch james.henderson@ed.ac.uk

Abstract

Candidate selection from n-best lists is a widely used approach in natural language parsing. Instead of attempting to select the most probable candidate, we focus on prediction of a new structure which minimizes an approximation to Bayes risk. Our approach does not place any restrictions on the probabilistic model used. We show how this approach can be applied in both dependency and constituent tree parsing with loss functions standard for these tasks. We evaluate these methods empirically on the Wall Street Journal parsing task.

1 Introduction

Much of the recent work in the field of natural language parsing has been focused on the candidate reranking scenario, where a baseline probabilistic model provides a n-best list of candidate parse trees and then a discriminative classifier selects one candidate from the list [1, 5, 12, 3]. The discriminative classifier can often be regarded as a probabilistic model, with the most probable candidate according to this model being selected (i.e. maximum a-posteriori probability decoding).

A different use of the n-best lists has been considered in the machine translation and speech processing areas. Here candidate lists are often used to approximate conditional risk [13, 18]. Then either the candidate with the smallest approximate conditional risk is selected, or a search is performed for an optimal structure under this approximation. The latter is known as Minimum Bayes Risk (MBR) decoding.

MBR decoding differs from Maximum A-posteriori Probability (MAP) decoding only when the loss function employed differs from 0-1 loss, as is the standard situation in complex structure prediction tasks and, particularly, in parsing. However, the only attempt until recently to optimize Bayes risk in parsing [7] only considered the constituent tree parsing task with binarized unlexicalized probabilistic context-free grammars (PCFG). They achieved very low accuracy with their over-simplistic model.

In this paper we focus on MBR decoding on the basis of n-best lists, without placing any constraints on what probabilistic model is used. We consider both the constituent and dependency tree parsing tasks. A novel algorithm is proposed for prediction of a Minimum Bayes risk (MBR) tree given a candidate list and a standard measure of accuracy, F_1 measure on labeled constituents. In dependency parsing, we show that MBR decoding with standard accuracy measures can be done using previously proposed algorithms. The MBR decoding approach is evaluated on the standard Wall Street Journal (WSJ) parsing task with different generative and discriminative models.[1] In all the cases a significant improvement over MAP decoding is achieved. When we use the best parsing method for this task [1], our approach for MBR decoding demonstrates the best published result for WSJ constituent parsing task, a F_1 score of 91.7%.

2 Bayes risk minimization

The Bayes risk of a model $y = h(x)$ is defined as:

$$R(h) = E_{x,y}\Delta(y, h(x)),$$

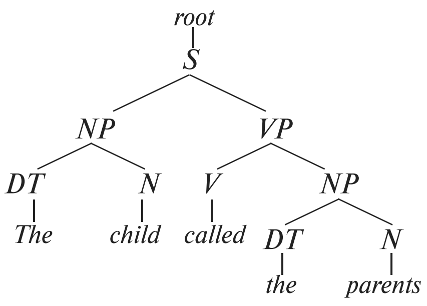
where the expectation is taken over all the possible sentences x and trees y and $\Delta(y, y')$ denotes a loss incurred by selecting a tree y' for a sentence x when the correct tree is y. It follows that an optimal classifier $h^\star$ is one which chooses the tree y that minimizes the conditional risk:

$$h^\star(x) = \arg\min_{y'} \sum_{y} P(y|x)\Delta(y, y'),$$

where minimization is performed over all possible trees for sentence x.

In order to estimate the conditional risk for a tree y on the basis of a candidate list, we have to make the assumption that the risk is proportional to the average loss in respect to the elements

[1]Software for MBR decoding both for constituent and for dependency tree parsing will be available on the authors' webpages.



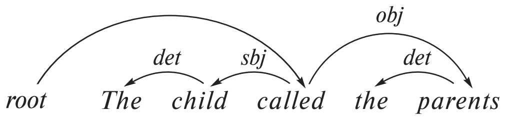


Figure 1: An example constituent tree (left), and a dependency tree (right).

of the candidate list:

$$\sum_{y'} P(y|x)\Delta(y,y') \approx \alpha \frac{\sum_{y' \in G(x)} P(y'|x)\Delta(y,y')}{\sum_{y' \in G(x)} P(y'|x)},$$

where $G(x)$ denotes a candidate list provided by a baseline model for the input x and α is some constant value. Replacing the true probabilities with their estimates $P_\theta(y'|x)$, we can define the model

$$\hat{h}(x) = \arg\min_{y} \sum_{y' \in G(x)} P_\theta(y'|x)\Delta(y,y'). \tag{1}$$

Probability estimates can be provided by the baseline model, which can be either conditional $P_\theta(y|x)$ or joint $P_\theta(x,y)$, since the normalization factor does not affect $\hat{h}(x)$. As will be discussed in section 5, the probability estimates can also be computed from the most widely used non-probabilistic discriminative reranking models.

In the following two sections, we will consider how MBR decoding (equation 1) can be performed for standard loss functions in the constituent and dependency tree parsing tasks. For more complex loss function or very tight constraints on the running time, a simpler approach can be used, where a parse tree is selected from the candidate list:

$$\bar{h}(x) = \arg\min_{y \in G(x)} \sum_{y' \in G(x)} P_\theta(y'|x)\Delta(y,y'). \tag{2}$$

We will refer to this approach in equation 2 as MBR reranking, and to the complete decoding approach in equation 1 as MBR prediction.

3 Constituent tree parsing

Parsing is the task of mapping from sequences of words to trees. The most common type of tree used is the constituent tree, illustrated on the left in Figure 1. A constituent tree defines a hierarchical decomposition of a sentence into phrases. We can think of a constituent tree as a set of non-crossing labeled brackets, i.e. triples (X, s, e), where X is a label of an internal node, s and e are indexes of the first and the last words spanned by the node. Accuracy of the model is evaluated by comparison of a tree y predicted by the model with a 'gold standard' tree $y^\star$ provided by a human annotator. Standard measures of accuracy for constituent parsing involve the number of equally labeled brackets $|y \bigcap y^\star|$, where for bracket equality both the spans and the labels should be the same. The Labeled Recall (LR) measure is defined as $|y \bigcap y^\star|/|y^\star|$, Labeled Precision (LP) as $|y \bigcap y^\star|/|y|$ and their harmonic mean $F_1(y^\star, y) = 2|y \bigcap y^\star|/(|y^\star| + |y|)$. Individual minimization of the LP measure is not possible and individual minimization of the LR measure will make a strong preference towards large trees.[2] It follows that the natural choice is to use the loss function

[2]In [7] optimization of the LR measure was considered, but because they only allow binary trees the denominator in $|y \bigcap y^\star|/|y^\star|$ does not depend on the selected tree y.

$\Delta(y, y') = (1 - F_1(y, y'))$ in MBR decoding methods. Then the expression (1) for this loss function can be rewritten as

$$\hat{h}(x) = \arg\max_{y} \sum_{y' \in G(x)} P_\theta(y'|x) \frac{|y \bigcap y'|}{|y'| + |y|}.$$

We can further transform this into:

$$\hat{h}(x) = \arg\max_{y} \sum_{b \in y} w_{|y|}(b), \tag{3}$$

where the bracket weight $w_{|y|}(b)$ is defined as

$$w_m(b) = \sum_{y' \in G(x)} I_{y'}(b) \frac{P_\theta(y'|x)}{|y'| + m},$$

and I_X denotes the indicator function for a set X. Thus, for a fixed number of brackets m, the task of finding of an optimal tree with m brackets is equivalent to the search for a set of m non-crossing brackets with maximum weight, where the weight of a set is the sum of weights of its elements $w_m(b)$.[3] We can solve this task with dynamic programming. Then the complete maximization in expression 3 can be done by considering an appropriate range of values for m.

Let $C[p, l, s]$ be a dynamic programming table that stores the score of the maximal subforest with p internal nodes (brackets) spanning a subsequence from position s to position $s + l - 1$. We denote as v_j a word at position j in the sentence x. For each split of the subsequence v_s,..., v_{s+l-1} defined by index k, we can distinguish 3 sets of brackets: brackets in the forest spanning the left subsequence v_s,..., v_{s+k-1}, brackets in the forest spanning the right subsequence v_{s+k},..., v_{s+l-1} and brackets of the form $(X, s, s + l - 1)$ spanning the whole subsequence v_s,..., v_{s+l-1}. All the possible sizes of these sets which sum up to p should be considered. If we consider outputting t brackets of the form $(X, s, s+l-1)$, then the optimal choice is to select from the candidates trees the t brackets with the largest weights $w_m(b)$. The recursive computation of $C[p, l, s]$ can be summarized in the following expression:

$$C[p, l, s] = \max_{\substack{0 \le t \le min(N[l,s], p) \\ 0 \le u \le p - t \\ 1 \le k \le l - 1}} \Big(\sum_{1 \le i \le t} w_m(b[l, s, i]) + C[u, k, s] + C[p - u - t, l - k, s + k] \Big),$$

where u is the number of brackets in the a subforest spanning the left subsequence, $b[l, s, 1]$,..., $b[l, s, N[l, s]]$ is the set of brackets of the form $(X, s, s + l - 1)$ sorted in decreasing order of their weights, and $N[l, s]$ is the size of this set. The weight of the optimal forest of m elements is given by $C[m, n, 1]$ where n is a length of the sentence. Pseudo-code for the algorithm is given in Figure 2. A simple modification of this algorithm that keeps track of optimal values for k, t and u, can be used to recover a tree. If we note that $N[l, s]$ is bounded by the number of possible node labels, then it is easy to see that the algorithm has a complexity of $O(m^2 n^3)$.

It can be shown that to perform maximization across values of m in 3 only a fixed range of values needs to be considered:

$$\lceil \frac{\beta}{2 - \beta} \min_{y' \in G(x)} |y'| \rceil \le m \le \lfloor \frac{(2 - \beta)}{\beta} \max_{y' \in G(x)} |y'| \rfloor, \quad \beta = \frac{\max_{y' \in G(x)} P_\theta(y'|x)}{\sum_{y' \in G(x)} P_\theta(y'|x)}. \tag{4}$$

Therefore, the total runtime of the algorithm for finding an optimal tree is $O(n^6 \log^3 n)$. In practice this algorithm is quite tractable, because a multiplicative constant is small. This complexity would be prohibitive for a parser with a large number of node labels, but MBR decoding does not need to choose them from the full set for each span but only from a small set of labels appearing in the candidates. Also, in practice a much smaller range of m values than stated in 4 can be considered. In all the experiments discussed in section 6, average MBR decoding time on a standard desktop PC was always below 0.2 seconds per sentence.

[3]Note that any set of m non-crossing brackets can be used to define a tree by adding a root bracketing which spans the whole sentence. The root bracketing is ignored in the standard loss measures, and we ignore it when measuring the size of a tree.

```
$C[p,l,s] = 0 \; \forall \; p,l,s$
for $p = 1$ to $m$ do
  for $l = 1$ to $n$ do
    for $s = 1$ to $n - l + 1$ do
      $w_{loc} = 0$, $t = 0$
      while $t \leq N[l,s]$ and $t \leq p$ do
        $w_{loc} := w_{loc} + w_m(b[l,s,t])$
        for $k = 1$ to $l - 1$ do
          $C[p,l,s] := \max_{0 \leq u \leq p-t} (C[p,l,s], w_{loc} + C[u,k,s] + C[p-u-t, l-k, s+k])$
        end for
        $t := t + 1$
      end while
    end for
  end for
end for
return $C[m,n,1]$
```

Figure 2: An algorithm for finding the weight of an optimal forest with m internal nodes.

4 Dependency tree parsing

Another common way to define parse trees is as a dependency tree [11], illustrated on the right in Figure 1. There is an increasing interest in dependency parsing because learning to predict dependency trees is a simpler task than learning to predict constituent trees, and because for a number of applications the dependency representation is more appropriate than hierarchical phrase structure. In this section we show how the complete decoding approach in expression 1 can be computed efficiently for dependency trees.

Dependency trees represent dependences of words in sentences as an acyclic directed graphs with labeled edges, where each label represents the type of the relation. As we did before for constituent trees, we augment the dependency trees with a *root* node which serves as a root for the tree. For each edge a parent node is called a head and a child node is called a dependent. In Figure 1, the tree is drawn with the original order of the words and without any crossing edges, making it a projective dependency tree. A tree is projective iff for any edge $v \rightarrow u$ and any word w between u and v, v is an ancestor of w. Even though the projectivity requirement is too strict for many languages, it is widely used in practice and often viewed as a reasonable assumption. In this section we will briefly discuss how Bayes risk minimization can be performed both with projectivity requirement and without it.

Standard measures of accuracy for dependency parsing are labeled and unlabeled relation accuracies. The labeled accuracy is the fraction of relations where both edges and labels match, and the unlabeled accuracy is defined similarly but without requiring labels to match. We will consider optimization of unlabeled accuracy but it is trivial to optimize the labeled accuracy in the same way. If we consider a tree as a set of edges, then expression 1 can be rewritten as

$$\hat{h}(x) = \arg\max_{y} \sum_{y' \in G(x)} P_\theta(y'|x) |y \bigcap y'|.$$

We can represent a score of a dependency tree y as a linear function

$$\hat{h}(x) = \arg\max_{y} \sum_{e \in y} w(e),$$

where $w(e)$, a score for a relation e, is defined as

$$w(e) = \sum_{y' \in G(x)} P_\theta(y'|x) I_{y'}(e).$$

Consequently, the score for a tree y is decomposed into a sum of scores of individual relations. The case of linear models with feature representation decomposable into a sum over individual relations was studied in [15]. For non-projective dependency trees, decoding is equivalent to searching for a Minimum Spanning Tree in directed graphs, a problem for which $O(n^2)$ algorithms are known [19]. For projective dependency trees, as pointed out in [15], the parsing algorithm of [6], with a runtime of $O(n^3)$, can be used.

5 Probability estimation with discriminative classifiers

As we mentioned in the introduction, n-best lists in parsing are often used to learn a discriminative classifier, called a reranker, to predict the best candidate in a list. Consequently, we might expect better performance if we used probability estimates from rerankers in 1, rather than using estimates from baseline models. This is trivial when a discriminative reranker defines a probabilistic model, but less so if voted models or maximal margin classifiers are used. In this section we will briefly review applicable techniques.

If an SVM is used to learn the classier, we suggest using the normalized exponential form to estimate probabilities of candidates in this list. This form can be viewed as a direct generalization of the approach proposed in [17], where the logistic sigmoid of the SVM output is used as the probability estimator for binary classification problems. Therefore, the appropriate form of the probability estimate for parsing and other structured classification problems is the following:

$$\hat{h}(x) = \arg\min_{y} \sum_{y' \in G(x)} \exp(A\hat{w}^T \phi(y'))\Delta(y, y'), \tag{5}$$

where $\hat{w}$ is a decision vector learned during classifier training, $\phi(y')$ is a feature representation of the tree y' and the scalar parameter A should be tuned on a development set.

Another type of models popular in natural language processing are voted models [4]. For such models, which combine classifiers using votes, the number of votes cast for each candidate can be used to define this discriminative probability. The discriminative probability of a candidate is simply the number of votes cast for that candidate normalized across candidates. Intuitively, we can think of this method as treating the votes as a sample from the discriminative distribution.

6 Experimental evaluation

To perform empirical evaluations of the proposed methods, we considered the task of constituent tree parsing of the Penn Treebank Wall Street Journal corpus [14]. First, we perform experiments with SVM Struct [20] as the learner. However, use of SVM Struct for large scale parsing experiments is computationally expensive[4], so here we use only a small portion of the available training data to perform evaluation. In the other two sets of experiments, described below, we test our best model on the standard Wall Street Journal parsing benchmark [2] with the Voted Perceptron (VP) algorithm [4] and maximum entropy reranker [1] as the the learners.

6.1 Experiments with SVM Struct and a neural network probabilistic model

For our first set of experiments, we choose to use a publicly available neural network based generative model of parsing, Simple Synchrony Networks (SSN) [8], as the baseline model. As a discriminative reranker we use a SVM modification [20] with the TOP Reranking Kernel (TRK) derived from the SSN probabilistic model [9]. This model combination is a good candidate for our experiments because the SSN parser alone achieves state-of-the-art results on the standard Wall Street Journal (WSJ) parsing problem, and data-defined kernels derived from it have recently been used with the Voted Perceptron algorithm on the full scale WSJ parsing task, achieving a significant improvement in accuracy over the neural network parser alone [9].

[4]Proposals have been made for addressing this problem, but since this issue is orthogonal to those addressed in this paper, we do not consider them here.

Table 1: Percentage labeled constituent recall (LR), precision (LP), combination of both (F_1) and percentage complete match (CM) on the testing set.

	SSN	SSN-Rerank	SSN-Predict	TRK	TRK-Rerank	TRK-Predict
F_1	81.3	81.8	82.1	81.7	82.1	82.3
LP	81.7	82.3	82.6	82.4	82.8	83.1
LR	80.9	81.4	81.6	81.1	81.5	81.6
CM	18.3	18.3	18.7	18.2	18.6	18.9

Both the neural network probabilistic model and the discriminative classifier were trained on section 0 (1,921 sentences, 40,930 words). Section 24 (1,346 sentences, 29,125 words) was used as the validation set during the neural network learning and for choosing parameters of the models. Section 23 (2,416 sentences, 54,268 words) was used for the final testing of the models. For training and testing of the kernel models, we provided a candidate list consisting of the top 20 parses found by the probabilistic model. For the testing set, selecting the candidate with an oracle results in an F_1 score of 89.1%.

We used the SVM-Struct software package [20] to train the SVM with the TRK kernel [9], with slack rescaling and linear slack penalty. The loss function is defined as $\Delta(y, y') = 1 - F_1(y, y')$, where F_1 denotes F_1 measure on bracketed constituents. This loss function was used for rescaling the slacks in all the SVM models, as well as in the definition of the Bayes risk. Model parameters, including the parameter A in 5, were adjusted on the basis of validation set accuracy.

Standard measures of parsing accuracy, plus complete match accuracy (0-1 error), on the final testing set are shown in table 1.[5] As the baselines, the table includes the results of the standard SVM classifier with the TRK kernel [9], and the baseline probabilistic model (SSN) [8]. SSN-Rerank and SSN-Predict are decoding methods that use probability estimates from the SSN probabilistic model, whereas TRK-Rerank and TRK-Predict a based on probability estimates from the SVM. Both SSN-Rerank and TRK-Rerank are MBR reranking approaches, as in 2, whereas in SSN-Predict and TRK-Predict full MBR decoding is performed with the dynamic programming algorithm proposed in section 3.

All the proposed MBR approaches show better F_1 accuracy than the MAP decoding with the corresponding model. Both full MBR decoding methods demonstrate better accuracy than the corresponding MBR reranking approaches (SSN-Predict vs SSN-Rerank, TRK-Predict vs TRK-Rerank). All these differences are statistically significant.[6] It should also be noted that, surprisingly, exact match for SSN-Predict and TRK-Predict is also improved, even though the F_1 loss function was optimized.

These experimental results demonstrate that the MBR decoding approaches considered in this paper demonstrate significant improvement over the baseline MAP decoding approaches. The relative error reduction over MAP decoding approaches is larger than the error reduction that was previously achieved by reranking with data-defined kernels [9] over the baseline SSN model.

6.2 Experiments with the voted perceptron algorithm and tree kernels

The above experiments with SVM Struct demonstrate empirically the viability of our approaches. The aim of experiments on the entire WSJ is to test whether our approaches still achieve significant improvement when more accurate models are used, and also to show that they generalize well to learning methods different from SVMs. We replicated the parse reranking experimental setup used for the evaluation of the Tree Kernel in [4], where the candidate list was provided by the generative probabilistic model [2] (model 2). A list of on average 29 candidates was used, with an oracle F_1 score on the testing set of 95.0%. We trained the VP algorithm using the same parameters for the Tree Kernel and probability feature weighting as described in [4]. A publicly available efficient implementation of the Tree Kernel was utilized to speed up computations [16]. As in the previous section, votes of the perceptron were used to define the probability estimate used in the classifier.

[5] All our results are computed with the evalb program [2].

[6] We measured significance of all the experiments in this paper with the randomized significance test [21].

Table 2: Result on the testing set. Percentage labeled constituent recall (LR), precision (LP), combination of both (F_1), an average number of crossing brackets per sentence (CB), percentage of sentences with 0 and ≤ 2 crossing brackets (0C and 2C, respectively).

	LR	LP	F_1*	CB	0C	2C
CO99	88.1	88.3	88.2	1.06	64.0	85.1
TK	88.6	88.9	88.7	0.99	66.5	86.3
TK-Rerank	89.0	89.5	89.2	0.91	66.6	87.4
TK-Predict	89.1	89.5	89.3	0.89	66.9	87.6

	LR	LP	F_1	CB	0C	2C
ME	91.0	91.8	91.4	0.73	73.3	89.7
ME-Rerank	91.2	92.1	91.6	0.68	73.7	90.9
ME-Predict	91.3	92.1	91.7	0.67	74.5	91.1

* F_1 for previous models may have rounding errors.

The results for our Bayes risk reranking and prediction approaches (TK-Rerank and TK-Predict, respectively), along with the standard Tree Kernel VP results (TK) [4] and the probabilistic baseline [2] (CO99) are presented on the left in table 2. The Bayes risk minimization models improve in F_1 score over the standard VP results. Differences between them and the TK model are statistically significant. The achieved error reduction over the TK model is larger than the error reduction of the standard Tree Kernel VP over MAP decoding with the probabilistic model CO99, and this improvement is achieved without adding any additional linguistic features, but by using only a different decoding method. It is interesting to note that the model improves in other accuracy measures as well.

6.3 Experiments with a maximum entropy reranker

In the last set of experiments we considered the parsing model which achieves the best results on WSJ parsing task [1]. It consists of a 50-best generative parser and a maximum entropy reranker. We used probability estimates given by the reranker to define the risk approximation. The experimental setup of [1] was replicated. However the maximum entropy reranker results appeared to be considerably better than the results published in [1].[7] Selecting the candidate with an oracle results in an F_1 score of 96.8%. Accuracy measure for the maximum entropy reranking (MER) and our MBR decoding approaches (ME-Predict, ME-Rerank) are presented on the right in table 2.

Both MBR decoding methods improve over MAP decoding (ME) and ME-Predict improvement is statistically significant and corresponds to about 3% error reduction.[8] We would expect a larger improvement in practice, because normally a parser trained on the WSJ is applied to the data from a different domain. In this case a probabilistic model is generally more uncertain, i.e. the entropy of the distribution in the list is larger. Therefore, the difference between MAP and MBR predictions is also expected to be larger.

7 Conclusions

This paper considers the application of MBR decoding approaches to the natural language parsing task. We proposed techniques for full MBR decoding on the basis of risk estimates from candidate lists in both the constituent and dependency parsing tasks. Our approaches do not place any constraints on the probabilistic models used, thus allowing it to be used with any generative or discriminative parsing method. The proposed decoding methods achieve significant improvement over MAP decoding.

We would expect even better results with MBR-decoding if larger n-best lists were used. The n-best parsing algorithm [10] can be used to efficiently produce candidate lists as large as 10^6 parse trees with the model of [2]. It is also worth noting that runtime of MBR-decoding algorithms is not significantly affected by the size of candidate lists.

[7]The software implementation of [1] is publicly available. We used the release of May, 2006.

[8]If we use the upper bound of 97% suggested in [1], rather than 100%, then the error reduction is 5%. 97% is motivated by inter-annotator agreement for the WSJ dataset.

References

[1] Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proc. 43rd Meeting of Association for Computational Linguistics*, pages 173–180, Ann Arbor, MI, 2005.

[2] Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing.* PhD thesis, University of Pennsylvania, Philadelphia, PA, 1999.

[3] Michael Collins and Nigel Duffy. Convolution Kernels for Natural Language. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Adv. Neural. Inform. Process Syst.*, volume 14, pages 625–632. MIT Press, 2001.

[4] Michael Collins and Nigel Duffy. 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, Philadelphia, PA, 2002.

[5] Michael Collins and Terry Koo. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–69, 2005.

[6] Jason Eisner. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, Denmark, 1996.

[7] Joshua Goodman. Parsing algorithms and metrics. In *Proc. 34th Meeting of the Association for Computational Linguistics*, pages 177–183, Santa Cruz, CA, 1996.

[8] James Henderson. 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, 2003.

[9] James Henderson and Ivan Titov. Data-defined kernels for parse reranking derived from probabilistic models. In *Proc. 43rd Meeting of Association for Computational Linguistics*, Ann Arbor, MI, 2005.

[10] Liang Huang and David Chiang. Better k-best parsing. In *Proc. 9th Int. Workshop on Parsing Technologies*, Vancouver, Canada, 2005.

[11] R. Hudson. *Word Grammar.* Basil Blackwell, Oxford, 1984.

[12] Terry Koo and Michael Collins. Hidden-variable models for discriminative reranking. In *Proc. Conf. on Empirical Methods in Natural Language Processing*, Vancouver, B.C., Canada, 2005.

[13] Shankar Kumar and William Byrne. Minimum bayes-risk decoding for statistical machine translation. In *Proceedings of the Human Language Technology Conference and Meeting of the North American Chapter of the Association for Computational Linguistics*, Boston, MA, 2004.

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

[15] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, October 2005.

[16] Alessandro Moschitti. A study on convolutional kernels for shallow semantic parsing. In *Proc. 42nd Meeting of the Association for Computational Linguistics*, Barcelona, Spain, 2004.

[17] John C. Platt. Probabilistic outputs for support vector machines and comparision to regularized likelihood methods. In A. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[18] Andreas Stolcke, Yochai Konig, and Mitchel Weintraub. Explicit word error minimization in n-best list rescoring. In *Proc. of 5th European Conference on Speech Communication and Technology*, pages 163–165, Rhodes, Greece, 1997.

[19] R. E. Tarjan. Finding optimal branchings. *Networks*, 7:25–35, 1977.

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

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