Multilingual Dependency Parsing using Bayes Point Machines

Simon Corston-Oliver
Microsoft Research
One Microsoft Way
Redmond, WA 98052
simonco@microsoft.com

Anthony Aue
Microsoft Research
One Microsoft Way
Redmond, WA 98052
anthaue@microsoft.com

Kevin Duh
Dept. of Electrical Eng.
Univ. of Washington
Seattle, WA 98195
duh@ee.washington.edu

Eric Ringger
Computer Science Dept.
Brigham Young Univ.
Provo, UT 84602
ringger@cs.byu.edu

Abstract

We develop dependency parsers for Arabic, English, Chinese, and Czech using Bayes Point Machines, a training algorithm which is as easy to implement as the perceptron yet competitive with large margin methods. We achieve results comparable to state-of-the-art in English and Czech, and report the first directed dependency parsing accuracies for Arabic and Chinese. Given the multilingual nature of our experiments, we discuss some issues regarding the comparison of dependency parsers for different languages.

1 Introduction

Dependency parsing is an alternative to constituency analysis with a venerable tradition going back at least two millenia. The last century has seen attempts to formalize dependency parsing, particularly in the Prague School approach to linguistics (Tesnière, 1959; Melčuk, 1988).

In a dependency analysis of syntax, words directly modify other words. Unlike constituency analysis, there are no intervening non-lexical nodes. We use the terms child and parent to denote the dependent term and the governing term respectively.

Parsing has many potential applications, ranging from question answering and information retrieval to grammar checking. Our intended application is machine translation in the Microsoft Research Treelet Translation System (Quirk et al., 2005; Menezes and Quirk, 2005). This system expects an analysis of the source language in which words are related by directed, unlabeled dependencies. For the purposes of developing machine translation for several language pairs, we are interested in dependency analyses for multiple languages.

The contributions of this paper are two-fold: First, we present a training algorithm called Bayes Point Machines (Herbrich et al., 2001; Harrington et al., 2003), which is as easy to implement as the perceptron, yet competitive with large margin methods. This algorithm has implications for anyone interested in implementing discriminative training methods for any application. Second, we develop parsers for English, Chinese, Czech, and Arabic and probe some linguistic questions regarding dependency analyses in different languages. To the best of our knowledge, the Arabic and Chinese results are the first reported results to date for directed dependencies. In the following, we first describe the data (Section 2) and the basic parser architecture (Section 3). Section 4 introduces the Bayes Point Machine while Section 5 describes the features for each language. We conclude with experimental results and discussions in Sections 6 and 7.

2 Data

We utilize publicly available resources in Arabic, Chinese, Czech, and English for training our dependency parsers.

For Czech we used the Prague Dependency Treebank version 1.0 (LDC2001T10). This is a corpus of approximately 1.6 million words. We divided the data into the standard splits for training, devel-

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 160–167,
New York, June 2006. ©2006 Association for Computational Linguistics

opment test and blind test. The Prague Czech Dependency Treebank is provided with human-edited and automatically-assigned morphological information, including part-of-speech labels. Training and evaluation was performed using the automatically-assigned labels.

For Arabic we used the Prague Arabic Dependency Treebank version 1.0 (LDC2004T23). Since there is no standard split of the data into training and test sections, we made an approximate 70%/15%/15% split for training/development test/blind test by sampling whole files. The Arabic Dependency Treebank is considerably smaller than that used for the other languages, with approximately 117,000 tokens annotated for morphological and syntactic relations. The relatively small size of this corpus, combined with the morphological complexity of Arabic and the heterogeneity of the corpus (it is drawn from five different newspapers across a three-year time period) is reflected in the relatively low dependency accuracy reported below. As with the Czech data, we trained and evaluated using the automatically-assigned part-of-speech labels provided with the data.

Both the Czech and the Arabic corpora are annotated in terms of syntactic dependencies. For English and Chinese, however, no corpus is available that is annotated in terms of dependencies. We therefore applied head-finding rules to treebanks that were annotated in terms of constituency.

For English, we used the Penn Treebank version 3.0 (Marcus et al., 1993) and extracted dependency relations by applying the head-finding rules of (Yamada and Matsumoto, 2003). These rules are a simplification of the head-finding rules of (Collins, 1999). We trained on sections 02-21, used section 24 for development test and evaluated on section 23. The English Penn Treebank contains approximately one million tokens. Training and evaluation against the development test set was performed using human-annotated part-of-speech labels. Evaluation against the blind test set was performed using part-of-speech labels assigned by the tagger described in (Toutanova et al., 2003).

For Chinese, we used the Chinese Treebank version 5.0 (Xue et al., 2005). This corpus contains approximately 500,000 tokens. We made an approximate 70%/15%/15% split for training/development test/blind test by sampling whole files. As with the English Treebank, training and evaluation against the development test set was performed using human-annotated part-of-speech labels. For evaluation against the blind test section, we used an implementation of the tagger described in (Toutanova et al., 2003). Trained on the same training section as that used for training the parser and evaluated on the development test set, this tagger achieved a token accuracy of 92.2% and a sentence accuracy of 63.8%.

The corpora used vary in homogeneity from the extreme case of the English Penn Treebank (a large corpus drawn from a single source, the Wall Street Journal) to the case of Arabic (a relatively small corpus–approximately 2,000 sentences–drawn from multiple sources). Furthermore, each language presents unique problems for computational analysis. Direct comparison of the dependency parsing results for one language to the results for another language is therefore difficult, although we do attempt in the discussion below to provide some basis for a more direct comparison. A common question when considering the deployment of a new language for machine translation is whether the natural language components available are of sufficient quality to warrant the effort to integrate them into the machine translation system. It is not feasible in every instance to do the integration work first and then to evaluate the output.

Table 1 summarizes the data used to train the parsers, giving the number of tokens (excluding traces and other empty elements) and counts of sentences.[1]

3 Parser Architecture

We take as our starting point a re-implementation of McDonald's state-of-the-art dependency parser (McDonald et al., 2005a). Given a sentence x, the goal of the parser is to find the highest-scoring parse $\hat{y}$ among all possible parses $y \in Y$:

$$\hat{y} = \arg\max_{y \in Y} s(x, y) \qquad (1)$$

[1]The files in each partition of the Chinese and Arabic data are given at http://research.microsoft.com/˜simonco/HLTNAACL2006.

Language	Total Tokens	Training Sentences	Development Sentences	Blind Sentences
Arabic	116,695	2,100	446	449
Chinese	527,242	14,735	1,961	2,080
Czech	1,595,247	73,088	7,319	7,507
English	1,083,159	39,832	1,346	2,416

Table 1: Summary of data used to train parsers.

For a given parse y, its score is the sum of the scores of all its dependency links $(i, j) \in y$:

$$s(x, y) = \sum_{(i,j)\in y} d(i, j) = \sum_{(i,j)\in y} \mathbf{w} \cdot \mathbf{f}(i, j) \quad (2)$$

where the link (i, j) indicates a head-child dependency between the token at position i and the token at position j. The score $d(i, j)$ of each dependency link (i, j) is further decomposed as the weighted sum of its features $\mathbf{f}(i, j)$.

This parser architecture naturally consists of three modules: (1) a decoder that enumerates all possible parses y and computes the argmax; (2) a training algorithm for adjusting the weights $\mathbf{w}$ given the training data; and (3) a feature representation $\mathbf{f}(i, j)$. Two decoders will be discussed here; the training algorithm and feature representation are discussed in the following sections.

A good decoder should satisfy several properties: ideally, it should be able to search through all valid parses of a sentence and compute the parse scores efficiently. Efficiency is a significant issue since there are usually an exponential number of parses for any given sentence, and the discriminative training methods we will describe later require repeated decoding at each training iteration. We re-implemented Eisner's decoder (Eisner, 1996), which searches among all *projective* parse trees, and the Chu-Liu-Edmonds' decoder (Chu and Liu, 1965; Edmonds, 1967), which searches in the space of both projective and non-projective parses. (A projective tree is a parse with no crossing dependency links.) For the English and Chinese data, the head-finding rules for converting from Penn Treebank analyses to dependency analyses creates trees that are guaranteed to be projective, so Eisner's algorithm suffices. For the Czech and Arabic corpora, a non-projective decoder is necessary. Both algorithms are $O(N^3)$, where N is the number of words in a sentence.[2] Refer to (McDonald et al., 2005b) for a detailed treatment of both algorithms.

4 Training: The Bayes Point Machine

In this section, we describe an *online* learning algorithm for training the weights $\mathbf{w}$. First, we argue why an online learner is more suitable than a batch learner like a Support Vector Machine (SVM) for this task. We then review some standard online learners (e.g. perceptron) before presenting the Bayes Point Machine (BPM) (Herbrich et al., 2001; Harrington et al., 2003).

4.1 Online Learning

An online learner differs from a batch learner in that it adjusts $\mathbf{w}$ incrementally as each input sample is revealed. Although the training data for our parsing problem exists as a batch (i.e. all input samples are available during training), we can apply online learning by presenting the input samples in some sequential order. For large training set sizes, a batch learner may face computational difficulties since there already exists an exponential number of parses per input sentence. Online learning is more tractable since it works with one input at a time.

A popular online learner is the perceptron. It adjusts $\mathbf{w}$ by updating it with the feature vector whenever a misclassification on the current input sample occurs. It has been shown that such updates converge in a finite number of iterations if the data is linearly separable. The averaged perceptron (Collins, 2002) is a variant which averages the $\mathbf{w}$ across all iterations; it has demonstrated good generalization especially with data that is not linearly separable, as in many natural language processing problems.

[2]The Chu-Liu-Edmonds' decoder, which is based on a maximal spanning tree algorithm, can run in $O(N^2)$, but our simpler implementation of $O(N^3)$ was sufficient.

Recently, the good generalization properties of Support Vector Machines have prompted researchers to develop large margin methods for the online setting. Examples include the margin perceptron (Duda et al., 2001), ALMA (Gentile, 2001), and MIRA (which is used to train the parser in (McDonald et al., 2005a)). Conceptually, all these methods attempt to achieve a large margin and approximate the maximum margin solution of SVMs.

4.2 Bayes Point Machines

The Bayes Point Machine (BPM) achieves good generalization similar to that of large margin methods, but is motivated by a very different philosophy of Bayesian learning or model averaging. In the Bayesian learning framework, we assume a prior distribution over $\mathbf{w}$. Observations of the training data revise our belief of $\mathbf{w}$ and produce a posterior distribution. The posterior distribution is used to create the final $\mathbf{w}_{\text{BPM}}$ for classification:

$$\mathbf{w}_{\text{BPM}} = E_{p(\mathbf{w}|D)}[\mathbf{w}] = \sum_{i=1}^{|V(D)|} p(\mathbf{w}_i|D)\,\mathbf{w}_i \quad (3)$$

where $p(\mathbf{w}|D)$ is the posterior distribution of the weights given the data D and $E_{p(\mathbf{w}|D)}$ is the expectation taken with respect to this distribution. The term $|V(D)|$ is the size of the *version space* $V(D)$, which is the set of weights $\mathbf{w}_i$ that is consistent with the training data (i.e. the set of $\mathbf{w}_i$ that classifies the training data with zero error). This solution achieves the so-called *Bayes Point*, which is the best approximation to the Bayes optimal solution given finite training data.

In practice, the version space may be large, so we approximate it with a finite sample of size I. Further, assuming a uniform prior over weights, we get the following equation:

$$\mathbf{w}_{\text{BPM}} = E_{p(\mathbf{w}|D)}[\mathbf{w}] \approx \sum_{i=1}^{I} \frac{1}{I}\mathbf{w}_i \quad (4)$$

Equation 4 can be computed by a very simple algorithm: (1) Train separate perceptrons on different random shuffles of the entire training data, obtaining a set of $\mathbf{w}_i$. (2) Take the average (arithmetic mean) of the weights $\mathbf{w}_i$. It is well-known that perceptron training results in different weight vector solutions if the data samples are presented sequentially in different orders. Therefore, random shuffles of the data and training a perceptron on each shuffle is effectively equivalent to sampling different models ($\mathbf{w}_i$) in the version space. Note that this averaging operation should not be confused with ensemble techniques such as Bagging or Boosting–ensemble techniques average the output hypotheses, whereas BPM averages the weights (models).

```
Input: Training set $D = ((x_1, y_1), (x_2, y_2), \ldots, (x_T, y_T))$
Output: $\mathbf{w}_{\text{BPM}}$
Initialize: $\mathbf{w}_{\text{BPM}} = \mathbf{0}$
for $i$ = 1 to I; do
    Randomly shuffle the sequential order of samples in $D$
    Initialize: $\mathbf{w}_i = \mathbf{0}$
    for $t$ = 1 to T; do
        $\hat{y}_t = \mathbf{w}_i \cdot x_t$
        if ($\hat{y}_t \neq y_t$) then $\mathbf{w}_i = \mathbf{w}_i + y_t x_t$
    done
    $\mathbf{w}_{\text{BPM}} = \mathbf{w}_{\text{BPM}} + \frac{1}{I}\mathbf{w}_i$
done
```

Figure 1: Bayes Point Machine pseudo-code.

The BPM pseudocode is given in Figure 1. The inner loop is simply a perceptron algorithm, so the BPM is very simple and fast to implement. The outer loop is easily parallelizable, allowing speed-ups in training the BPM. In our specific implementation for dependency parsing, the line of the pseudocode corresponding to [$\hat{y}_t = \mathbf{w}_i \cdot x_t$] is replaced by Eq. 1 and updates are performed for each incorrect dependency link. Also, we chose to average each individual perceptron (Collins, 2002) prior to Bayesian averaging.

Finally, it is important to note that the definition of the version space can be extended to include weights with non-zero training error, so the BPM can handle data that is not linearly separable. Also, although we only presented an algorithm for linear classifiers (parameterized by the weights), arbitrary kernels can be applied to BPM to allow non-linear decision boundaries. Refer to (Herbrich et al., 2001) for a comprehensive treatment of BPMs.

5 Features

Dependency parsers for all four languages were trained using the same set of feature types. The feature types are essentially those described in (McDonald et al., 2005a). For a given pair of tokens, where one is hypothesized to be the parent and the other to be the child, we extract the word of the parent token, the part of speech of the parent token, the word of the child token, the part of speech of the child token and the part of speech of certain adjacent and intervening tokens. Some of these atomic features are combined in feature conjunctions up to four long, with the result that the linear classifiers described below approximate polynomial kernels. For example, in addition to the atomic features extracted from the parent and child tokens, the feature [ParentWord, ParentPOS, ChildWord, ChildPOS] is also added to the feature vector representing the dependency between the two tokens. Additional features are created by conjoining each of these features with the direction of the dependency (i.e. is the parent to the left or right of the child) and a quantized measure of the distance between the two tokens. Every token has exactly one parent. The root of the sentence has a special synthetic token as its parent.

Like McDonald et al, we add features that consider the first five characters of words longer than five characters. This truncated word crudely approximates stemming. For Czech and English the addition of these features improves accuracy. For Chinese and Arabic, however, it is clear that we need a different backoff strategy.

For Chinese, we truncate words longer than a single character to the first character.[3] Experimental results on the development test set suggested that an alternative strategy, truncation of words longer than two characters to the first two characters, yielded slightly worse results.

The Arabic data is annotated with gold-standard morphological information, including information about stems. It is also annotated with the output of an automatic morphological analyzer, so that researchers can experiment with Arabic without first needing to build these components. For Arabic, we truncate words to the stem, using the value of the lemma attribute.

All tokens are converted to lowercase, and numbers are normalized. In the case of English, Czech and Arabic, all numbers are normalized to a single token. In Chinese, months are normalized to a MONTH token, dates to a DATE token, years to a YEAR token. All other numbers are normalized to a single NUMBER token.

[3]There is a near 1:1 correspondence between characters and morphemes in contemporary Mandarin Chinese. However, most content words consist of more than one morpheme, typically two.

The feature types were instantiated using all oracle combinations of child and parent tokens from the training data. It should be noted that when the feature types are instantiated, we have considerably more features than McDonald et al. For example, for English we have 8,684,328 whereas they report 6,998,447 features. We suspect that this is mostly due to differences in implementation of the features that backoff to stems.

The averaged perceptrons were trained on the one-best parse, updating the perceptron for every edge and averaging the accumulated perceptrons after every sentence. Experiments in which we updated the perceptron based on k-best parses tended to produce worse results. The Chu-Liu-Edmonds algorithm was used for Czech. Experiments with the development test set suggested that the Eisner decoder gave better results for Arabic than the Chu-Liu-Edmonds decoder. We therefore used the Eisner decoder for Arabic, Chinese and English.

6 Results

Table 2 presents the accuracy of the dependency parsers. Dependency accuracy indicates for how many tokens we identified the correct head. Root accuracy, i.e. for how many sentences did we identify the correct root or roots, is reported as F1 measure, since sentences in the Czech and Arabic corpora can have multiple roots and since the parsing algorithms can identify multiple roots. Complete match indicates how many sentences were a complete match with the oracle dependency parse.

A convention appears to have arisen when reporting dependency accuracy to give results for English excluding punctuation (i.e., ignoring punctuation tokens in the output of the parser) and to report results for Czech including punctuation. In order to facilitate comparison of the present results with previously published results, we present measures including and excluding punctuation for all four languages. We hope that by presenting both sets of measurements, we also simplify one dimension along which published results of parse accuracy differ. A direct

	Including punctuation			Excluding punctuation		
Language	Dependency Accuracy	Root Accuracy	Complete Match	Dependency Accuracy	Root Accuracy	Complete Match
Arabic	79.9	90.0	9.80	79.8	87.8	10.2
Chinese	71.2	66.2	17.5	73.3	66.2	18.2
Czech	84.0	88.8	30.9	84.3	76.2	32.2
English	90.0	93.7	35.1	90.8	93.7	37.6

Table 2: Bayes Point Machine accuracy measured on blind test set.

comparison of parse results across languages is still difficult for reasons to do with the different nature of the languages, the corpora and the differing standards of linguistic detail annotated, but a comparison of parsers for two different languages where both results include punctuation is at least preferable to a comparison of results including punctuation to results excluding punctuation.

The results reported here for English and Czech are comparable to the previous best published numbers in (McDonald et al., 2005a), as Table 3 shows. This table compares McDonald et al.'s results for an averaged perceptron trained for ten iterations with no check for convergence (Ryan McDonald, pers. comm.), MIRA, a large margin classifier, and the current Bayes Point Machine results. To determine statistical significance we used confidence intervals for p=0.95. For the comparison of English dependency accuracy excluding punctuation, MIRA and BPM are both statistically significantly better than the averaged perceptron result reported in (McDonald et al., 2005a). MIRA is significantly better than BPM when measuring dependency accuracy and root accuracy, but BPM is significantly better when measuring sentences that match completely. From the fact that neither MIRA nor BPM clearly outperforms the other, we conclude that we have successfully replicated the results reported in (McDonald et al., 2005a) for English.

For Czech we also determined significance using confidence intervals for p=0.95 and compared results including punctuation. For both dependency accuracy and root accuracy, MIRA is statisticallty significantly better than averaged perceptron, and BPM is statistically significantly better than MIRA. Measuring the number of sentences that match completely, BPM is statistically significantly better than averaged perceptron, but MIRA is significantly better than BPM. Again, since neither MIRA nor BPM outperforms the other on all measures, we conclude that the results constitute a valiation of the results reported in (McDonald et al., 2005a).

For every language, the dependency accuracy of the Bayes Point Machine was greater than the accuracy of the best individual perceptron that contributed to that Bayes Point Machine, as Table 4 shows. As previously noted, when measuring against the development test set, we used human-annotated part-of-speech labels for English and Chinese.

Although the Prague Czech Dependency Treebank is much larger than the English Penn Treebank, all measurements are lower than the corresponding measurements for English. This reflects the fact that Czech has considerably more inflectional morphology than English, leading to data sparsity for the lexical features.

The results reported here for Arabic are, to our knowledge, the first published numbers for dependency parsing of Arabic. Similarly, the results for Chinese are the first published results for the dependency parsing of the Chinese Treebank 5.0.[4] Since the Arabic and Chinese numbers are well short of the numbers for Czech and English, we attempted to determine what impact the smaller corpora used for training the Arabic and Chinese parsers might have. We performed data reduction experiments, training the parsers on five random samples at each size smaller than the entire training set. Figure 2 shows the dependency accuracy measured on the complete development test set when training with samples of the data. The graph shows the average

[4](Wang et al., 2005) report numbers for *undirected* dependencies on the Chinese Treebank 3.0. We cannot meaningfully compare those numbers to the numbers here.

Language	Algorithm	DA	RA	CM
English (exc punc)	Avg. Perceptron	90.6	94.0	36.5
	MIRA	90.9	94.2	37.5
	Bayes Point Machine	90.8	93.7	37.6
Czech (inc punc)	Avg. Perceptron	82.9	88.0	30.3
	MIRA	83.3	88.6	31.3
	Bayes Point Machine	84.0	88.8	30.9

Table 3: Comparison to previous best published results reported in (McDonald et al., 2005a).

	Arabic	Chinese	Czech	English
Bayes Point Machine	78.4	83.8	84.5	91.2
Best averaged perceptron	77.9	83.1	83.5	90.8
Worst averaged perceptron	77.4	82.6	83.3	90.5

Table 4: Bayes Point Machine accuracy vs. averaged perceptrons, measured on development test set, excluding punctuation.

dependency accuracy for five runs at each sample size up to 5,000 sentences. English and Chinese accuracies in this graph use oracle part-of-speech tags. At all sample sizes, the dependency accuracy for English exceeds the dependency accuracy of the other languages. This difference is perhaps partly attributable to the use of oracle part-of-speech tags. However, we suspect that the major contributor to this difference is the part-of-speech tag set. The tags used in the English Penn Treebank encode traditional lexical categories such as noun, preposition, and verb. They also encode morphological information such as person (the VBZ tag for example is used for verbs that are third person, present tense–typically with the suffix -s), tense, number and degree of comparison. The part-of-speech tag sets used for the other languages encode lexical categories, but do not encode morphological information.[5] With small amounts of data, the perceptrons do not encounter sufficient instances of each lexical item to calculate reliable weights. The perceptrons are therefore forced to rely on the part-of-speech information.

It is surprising that the results for Arabic and Chinese should be so close as we vary the size of the training data (Figure 2) given that Arabic has rich morphology and Chinese very little. One possible explanation for the similarity in accuracy is that the rather poor root accuracy in Chinese indicates parses that have gone awry. Anecdotal inspection of parses suggests that when the root is not correctly identified, there are usually cascading related errors.

Czech, a morphologically complex language in which root identification is far from straightforward, exhibits the worst performance at small sample sizes. But (not shown) as the sample size increases, the accuracy of Czech and Chinese converge.

7 Conclusions

We have successfully replicated the state-of-the-art results for dependency parsing (McDonald et al., 2005a) for both Czech and English, using Bayes Point Machines. Bayes Point Machines have the appealing property of simplicity, yet are competitive with online wide margin methods.

We have also presented first results for dependency parsing of Arabic and Chinese, together with some analysis of the performance on those languages.

In future work we intend to explore the discriminative reranking of n-best lists produced by these parsers and the incorporation of morphological features.

[5]For Czech and Arabic we followed the convention established in previous parsing work on the Prague Czech Dependency Treebank of using the major and minor part-of-speech tags but ignoring other morphological information annotated on each node.

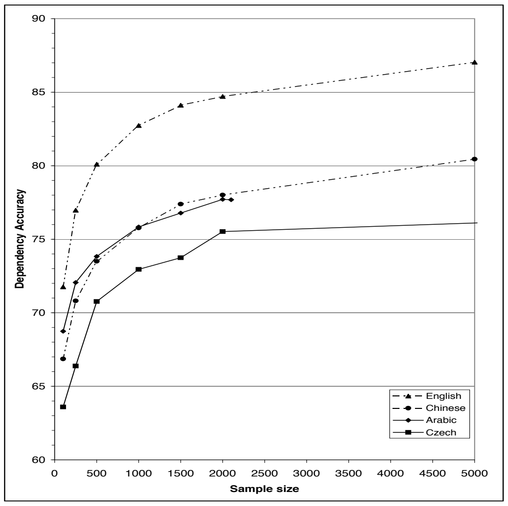

Figure 2: Dependency accuracy at various sample sizes. Graph shows average of five samples at each size and measures accuracy against the development test set.

Acknowledgements

We would like to thank Ryan McDonald, Otakar Smrž and Hiroyasu Yamada for help in various stages of the project.

References

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

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

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

R. O. Duda, P. E. Hart, and D. G. Stork. 2001. *Pattern Classification*. John Wiley & Sons, Inc.: New York.

J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING 1996*, pages 340–345.

Claudio Gentile. 2001. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242.

Edward Harrington, Ralf Herbrich, Jyrki Kivinen, John C. Platt, and Robert C. Williamson. 2003. Online bayes point machines. In *Proc. 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 241–252.

Ralf Herbrich, Thore Graepel, and Colin Campbell. 2001. Bayes point machines. *Journal of Machine Learning Research*, pages 245–278.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Assocation for Computational Linguistics*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005b. Online large-margin training of dependency parsers. Technical Report MS-CIS-05-11, Dept. of Computer and Information Science, Univ. of Pennsylvania.

Igor A. Melčuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

Arul Menezes and Chris Quirk. 2005. Microsoft research treelet translation system: IWSLT evaluation. In *Proceedings of the International Workshop on Spoken Language Translation*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd annual meeting of the Association for Computational Linguistics*.

Lucien Tesnière. 1959. *Éléments de syntaxe structurale*. Librairie C. Klincksieck.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pages 252–259.

Qin Iris Wang, Dale Schuurmans, and Dekang Lin. 2005. Strictly lexical dependency parsing. In *Proceedings of the Ninth International Workshop on Parsing Technologies*, pages 152–159.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2).

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, pages 195–206.