Ambiguous Part-of-Speech Tagging for Improving Accuracy and Domain Portability of Syntactic Parsers

Kazuhiro Yoshida[*] **Yoshimasa Tsuruoka**[†] **Yusuke Miyao**[*] **Jun'ichi Tsujii**[*†‡]
[*]Department of Computer Science, University of Tokyo
[†]School of Informatics, University of Manchester
[‡]National Center for Text Mining
{kyoshida,tsuruoka,yusuke,tsujii}@is.s.u-tokyo.ac.jp

Abstract

We aim to improve the performance of a syntactic parser that uses a part-of-speech (POS) tagger as a preprocessor. Pipelined parsers consisting of POS taggers and syntactic parsers have several advantages, such as the capability of domain adaptation. However the performance of such systems on raw texts tends to be disappointing as they are affected by the errors of automatic POS tagging. We attempt to compensate for the decrease in accuracy caused by automatic taggers by allowing the taggers to output multiple answers when the tags cannot be determined reliably enough. We empirically verify the effectiveness of the method using an HPSG parser trained on the Penn Treebank. Our results show that ambiguous POS tagging improves parsing if outputs of taggers are weighted by probability values, and the results support previous studies with similar intentions. We also examine the effectiveness of our method for adapting the parser to the GENIA corpus and show that the use of ambiguous POS taggers can help development of portable parsers while keeping accuracy high.

1 Introduction

Some parsers use POS taggers as their preprocessors, and some use integrated models that achieve tagging and parsing simultaneously. Because the latter approach is more general, it is successfully used by some of the state-of-the-art parsers, such as Charniak's [Charniak and Johnson, 2005], as a natural consequence of the pursuit of accuracy. However, integrated models of tagging and parsing tend to be complex and computationally expensive, both in terms of training and run-time costs.

On the other hand, pipelined systems of POS taggers and parsers can be built with independently developed taggers and parsers. In such models, we can easily make use of taggers that use state-of-the-art sequence labeling techniques, most of which are difficult to be incorporated into syntactic disambiguation models. Advantages of pipelined parsers also include their ability to adapt to domains. POS taggers for a new domain are much easier to develop than full parsers, because training corpora for POS taggers are easier to construct compared to those for full parsers, which require the annotation of nested phrase structures.

However, independence assumption of taggers and parsers may degrade the overall accuracy of the parsers. Watson [2006] reported that using an automatic POS tagger caused the F1 score of grammatical relations output by a parser to drop by 2.02 points. She attempted to weaken the independence assumption by letting the taggers output multiple tags for each word, weighted by probability values. Her approach improved the F1 score by 0.66 points.

In this paper, we verify Watson's results on ambiguous POS tagging using an HPSG [Pollard and Sag, 1994] parser developed and trained on the Penn Treebank [Marcus *et al.*, 1994]. At the same time, we investigate the effectiveness of ambiguous POS tagging for domain adaptation of parsers using the GENIA corpus [Ohta *et al.*, 2002] as the test set. Experimental results show that the multiple output without probability values cannot improve the parser much and suggest the importance of probability distribution of multiple tags obtained by POS taggers. Additionally, we show that the positive effect of ambiguous POS tagging is maintained for domains unfamiliar to the parser.

2 Background

In this section, we describe the POS tagger and syntactic parser used in our experiments. These taggers and parsers were combined to make a pipelined syntactic parser for raw texts using the strategy described in the next section.

As both of our tagging and parsing models are based on log-linear classifiers, we first briefly introduce log-linear models and then describe our tagger and parser.

2.1 Log-linear models

Log-linear models are among the most widely used machine learning techniques in NLP literature, and we use the models both for POS taggers and syntactic parsers. A conditional log-linear model calculates the probability of an event E given the

```
Input: Sentence $s$
Output: Tag sequence $t_1, \ldots t_n$
Algorithm:
  1. foreach $t_i$ do $t_i :=$ NULL
  2. foreach $t_i =$ NULL
       Compute probability distribution
       $P(t_i|t_{i-2}, t_{i-1}, t_{i+1}, t_{i+2}, s)$ (1)
       by log-linear models
  3. Let $i$ be easiest place to tag in
       $t_i$ := (Most probable tag for $t_i$)
  4. repeat 2 and 3 until
       $t_i \neq$ NULL for each $i$
```

Figure 1: Algorithm for POS tagging

condition C as follows:

$$P(E|C) = \frac{\exp(\sum_i \lambda_i f_i(E, C))}{Z_C},$$

where the real-valued functions f_i are used to indicate useful features for predicting E, parameters λ_i are optimized to fit the model to the training data, and Z_C is a normalization constant. Several criteria for estimating the parameters of log-linear models exist, and we used MAP estimation with Gaussian prior distribution [Chen and Rosenfeld, 1999], which is most commonly and successfully applied in various NLP tasks.

2.2 POS tagger

We employ the bidirectional inference model proposed by Tsuruoka et al. [2005] for our POS taggers. The model consists of 16 log-linear models, each of which provides the probability

$$P(t_i|t_{i-2}, t_{i-1}, t_{i+1}, t_{i+2}, s), \tag{1}$$

where s is a given sentence, and t_i is the POS tag for the ith word. An algorithm used by the POS taggers is shown in Figure 1. The key idea of the algorithm is that it does not work in the usual left-to-right manner, instead it iteratively tries to tag words that can be tagged most easily. "Easiness" of tagging is measured by the highest probability value among the probability distribution of the tags. When we calculated Eq. 1 in step 2 of the algorithm, each from $t_{i-2}, t_{i-1}, t_{i+1}, t_{i+2}$ could be NULL, so we prepared $2^4 = 16$ classifiers to cover every pattern of the appearances of NULL. Each token in a training corpus was used as training data for all 16 classifiers. The features used by the classifiers were the surface strings of words, base forms of words obtained by a dictionary, prefixes and suffixes, and POSs of already tagged words.

Though the algorithm described above is totally deterministic, a slight modification with beam search strategy could make it output an approximation of k-best tag sequences with probability values.

2.3 Grammar and parser

The parser we used is based on Head-Driven Phrase Structure Grammar (HPSG), which was developed using the Penn Treebank [Miyao *et al.*, 2004]. The disambiguation model for the HPSG grammar is a combination of two log-linear models.

The first log-linear model is for selecting lexical entries for words of a POS-tagged sentence, which estimates the probability

$$P(l_i|w_i, t_i), \tag{2}$$

where w_i, t_i, and l_i represent an ith word, POS tag, and lexical entry in a sentence, respectively. The information of w_i and t_i are used in combination as features of the log-linear model.

The second model is for selecting the most probable parse from an HPSG parse forest F which is constructed from lexical entries that are weighted by the first model:

$$P(T|l_{1,\ldots,n}, w_{1,\ldots,n}, t_{1,\ldots,n}),$$

where $T \in F$. The overall disambiguation model is

$$\operatorname*{argmax}_{T \in F} P(T|w_{1,\ldots,n}, t_{1,\ldots,n}) =$$
$$\operatorname*{argmax}_{T \in F} P(T|l_{1,\ldots,n}, w_{1,\ldots,n}, t_{1,\ldots,n}) \prod_i P(l_i|w_i, t_i).$$

The parsing algorithm for the disambiguation model is a CYK-like chart parsing with iterative beam search [Ninomiya *et al.*, 2005].

3 Combining POS taggers and parsers

In pipelined parsers, POS taggers and syntactic parsers are developed separately, and we can freely change taggers for parsers, without any special care for the algorithms. To make pipelined systems of ambiguous POS taggers and parsers work the same way, we restrict the interface of ambiguous taggers and formalize the condition of the syntactic disambiguation models that can be applied to the output of ambiguous taggers without modification.

We concentrate on the following situation: for each word in a sentence, POS taggers output a set of candidate POS tags, and the obtained tags are used to restrict parses in the parsing model.

In our experiments, we compared the following three types of taggers.

single The *single* tagger outputs the most probable single tag for each word.

multi The *multi* tagger outputs a set of candidate POS tags for each word.

prob The *prob* tagger is similar to *multi*, but each POS tag is weighted by its probability. That is, it provides the probability distribution $P(t_{ij}|S)$, where S is an input sentence and t_{ij} represents the jth candidate POS tag of the ith word.

We assume that the parsing model consists of two independent models: *terminal* and *non-terminal*. A parse T of a given sentence S consists of the terminal structure T^t and non-terminal structure T^{nt}, and terminal and non-terminal models are used to estimate the probability distributions $P(T^t|S)$ and $P(T^{nt}|T^t,S)$. The best parse is given by

$$\operatorname*{argmax}_{T^{nt},T^t} P(T^{nt},T^t|S) =$$
$$\operatorname*{argmax}_{T^{nt},T^t} P(T^{nt}|T^t,S)P(T^t|S).$$

Then, we assume that the terminal structure T^t can be further decomposed into a sequence of terminal labels $l_i, \dots, l_n$, where the label l_i corresponds to the ith word of the sentence, and terminal labels are independent of one another in the terminal model:

$$P(T^t|S) = \prod_i P(l_i|S).$$

The overall disambiguation model becomes

$$\operatorname*{argmax}_{T^{nt},T^t} P(T^{nt}|l_0,\dots,l_n,S) \prod_i P(l_i|S). \qquad (3)$$

We assume that a tractable estimation method and disambiguation algorithm applicable to Eq. 3 exist.

Let us then rewrite the distribution $P(l_i|S)$ to make it dependent on outputs of POS taggers, so that we can incorporate the information of POS tags into the parsing model.

$$P(l_i|S) = \sum_j P(t_{ij},S)P(l_i|t_{ij},S), \qquad (4)$$

where t_{ij} is the jth element of a set of POS tags assigned to the ith word of the sentence, and $P(t_{ij},S)$ is the probability of that tag calculated by the *prob* tagger (*multi* and *single* taggers can be integrated into the model similarly by assuming the tags assigned to the same word by the taggers are equally probable). By replacing $P(l_i|S)$ that appears in Eq. 3 with Eq. 4, we can apply the same disambiguation algorithm as Eq. 3 to the combined system of an ambiguous POS tagger and the parsing model.

This strategy is applicable to a wide range of grammars and disambiguation models including PCFG, where the terminal model is used to assign POS tags to words and lexicalized grammars such as HPSG, where the terminal model assigns lexical entries to words. The HPSG parser described in Section 2.3 is straightforwardly adapted to this model, by taking Eq. 2 as a terminal model.

One problem with this strategy is the increased ambiguity introduced by multiple tags. As reported by Watson [2006], the increase in computational costs can be suppressed by applying appropriate parsing algorithms. The parsing algorithm we used [Ninomiya *et al.*, 2005] is suitable for such purposes, and we will show experimentally that the disambiguation of the above model can be performed with efficiency similar to models with single taggers.

3.1 Implementation of *prob* tagger

There can be various methods for implementing the *prob* tagger described above. Our implementation outputs approximation of tag probabilities by marginalizing the probability of k-best tag sequences obtained by the algorithm described in Section 2.2.

LP/LR	F1
87.12/86.71	86.91

Table 1: Accuracy on section 22 with correct POS tags

To control the ambiguity of the tagger, we introduce a parameter θ which is used to threshold the candidate tags. If the marginalized probability of a candidate tag is smaller than the probability of the best tag of the same word multiplied by θ, the candidate is discarded.

Figure 2 illustrates example outputs of each tagger from the 3 best analysis of the sentence, “Mr. Meador had been executive vice president of Balcor,” without thresholding.

Marginalization of probability distribution output by POS taggers loses information about the sequential dependency of tags, which can harm the performance of the parser. For example, let us consider the sentence, “Time flies like an arrow.” When there are two candidate POS tag sequences “NN VBZ IN DT NN” and “VB NNS IN DT NN,” the output of *multi* tagger will be “{NN, VB} {VBZ, NNS} IN DT NN,” which can induce a tag sequence “VB VBZ IN DT NN” that is not included in the original candidates.

4 Experiments

The first experiment verified the effect of ambiguous POS tagging using the Penn Treebank for both training and testing. The second experiment examined the parser’s capability of domain adaptation, using the Penn Treebank for training the parser, biomedical texts for training the tagger, and the GENIA corpus for testing.

Both experiments used the same HPSG grammar and parser. The grammar was developed using the Penn Treebank sections 02-21, and the disambiguation model was trained on the same data. The accuracy of the parser evaluated on the Penn Treebank section 22[1] using the correct POS tags is shown in Table 1. *Labeled precision* (LP) and *labeled recall* (LR) are the precision and recall of dependencies with predicate-argument labels, and F1 is the harmonic mean of LP and LR. Each dependency between a pair of words is counted as one event. These measures were also used in the following experiments. The figures in Table 1 can be considered the practical upper bounds, because they are not affected by incorrect predictions of POS taggers. (We will refer to the results using the correct POS tags as *gold*.) In the following, comparisons of the systems were performed using the F1 scores.

4.1 Parsing Penn Treebank

The POS taggers described in this section were trained on the Penn Treebank sections 00-18, and the performance of the tagger on the development set was 96.79%. As described in Section 3.1, our taggers *multi* and *prob* have a hyperparameter θ for controlling the number of alternative answers

[1] Following the convention of literature on Penn Treebank parsing, we used section 22 for development, and section 23 for the final test.

Input	Mr	Meador	had	been	executive	vice	president	of	Balcor	.	
3 best sequences	NNP	NNP	VBD	VBN	JJ	NN	NN	IN	NNP	.	(0.879)
(probability)	NN	NN	VBD	VBN	JJ	NN	NN	IN	NNP	.	(0.075)
	NNP	NNP	VBD	VBN	NN	NN	NN	IN	NNP	.	(0.046)
single	NNP	NNP	VBD	VBN	JJ	NN	NN	IN	NNP	.	
multi	{NNP, NN}	{NNP, NN}	VBD	VBN	{JJ, NN}	NN	NN	IN	NNP	.	
prob	NNP(0.925)	NNP(0.925)	VBD	VBN	JJ(0.955)	NN	NN	IN	NNP	.	
(probability)	NN(0.075)	NN(0.075)			NN(0.045)						

Figure 2: Example of tagging results

	LP/LR	F1
multi $\theta = 0.1$	85.24/84.68	84.96
multi $\theta = 0.01$	84.13/83.64	83.88
multi $\theta = 0.001$	79.29/78.68	78.98
multi $\theta = 0.0001$	71.01/70.36	70.68
prob $\theta = 0.1$	85.26/84.73	84.99
prob $\theta = 0.01$	85.28/84.88	**85.08**
prob $\theta = 0.001$	85.19/84.80	84.99
prob $\theta = 0.0001$	85.08/84.60	84.84

Table 2: Accuracy with different θ

included in the output, where a smaller θ means a larger number of alternative answers. Before comparing ambiguous taggers (i.e., *multi* and *prob*) with *single*, we first determined which of *multi* and *prob* and which value of θ perform best.

The performance of the parser for various θ is shown in Table 2. The parsers with the *multi* tagger was outperformed by *prob* in all cases[2], so we only used the *prob* tagger in the following experiments. The performance of the parser with the *prob* tagger has a peak around $\theta = 0.01$. The accuracy slowly decreases for smaller θs, the reason for which might be the problem noted in the last paragraph of Section 3.1. We used the best performing θ $(= 0.01)$ in the following experiments.

It was interesting to see whether the contribution of ambiguous tagging changed according to the performance of the syntactic disambiguation model, because if ambiguous POS tagging did not work well with a poorer disambiguation model, ambiguous tagging could be considered not to help the performance in domains unfamiliar for the parser, where the performance of the disambiguation model was likely to be lower. We observed a change in performance of the parser when the number of sentences used for developing the grammar and for training the syntactic disambiguation model was changed. Table 3 shows the performance of the parser trained with various numbers of sentences selected from the Penn Treebank sections 02-21. Figure 3 shows F1 scores of the systems trained on 3000, 10000, and 39832 sentences.

Ambiguous POS tagging with the *prob* tagger could be contended to contribute to the performance of the parser, because the performance of the parser with the *prob* tagger was consistently better than that of the *single* tagger in all of the

# of training data	Tagging strategy	LP/LR	F1
300	*gold*	63.63/36.57	46.45
	single	62.51/35.43	45.23
	prob	61.46/39.62	48.18
1000	*gold*	75.21/68.41	71.65
	single	73.84/66.33	69.88
	prob	73.56/68.10	70.72
3000	*gold*	81.72/78.94	80.31
	single	79.97/77.42	78.67
	prob	79.90/78.34	79.11
10000	*gold*	85.35/84.32	84.83
	single	83.50/82.09	82.79
	prob	83.47/82.59	83.03
39832 (all)	*gold*	87.12/86.71	86.91
	single	85.26/84.72	84.99
	prob	85.28/84.88	85.08

Table 3: Accuracy with different sizes of training data

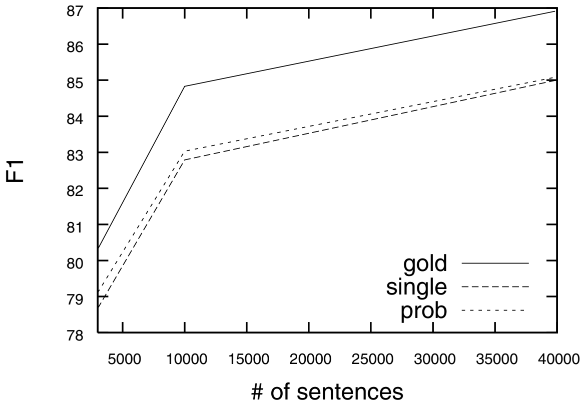

Figure 3: F1 of different sizes of training data

	LP/LR	F1	mean parsing time
gold	86.91/86.28	86.59	739 msec
single	84.41/83.80	84.10	785 msec
prob	85.24/84.89	85.06	936 msec

Table 4: Accuracy on test set

[2] Actually, *multi* has a negative effect, compared with the results of *single* shown in the 'all' row of Table 3, which has an F1 score of 84.99.

POS tagger	LP/LR	F1	mean parsing time
gold	83.21/81.57	82.38	845 msec
single-Penn	76.84/76.48	76.66	799 msec
prob-Penn	78.48/78.11	78.29	950 msec
single-BIO	81.75/80.65	81.20	803 msec
prob-BIO	82.60/81.44	82.02	873 msec

Table 5: Accuracy on GENIA

above experiments if evaluated in the F1 measure. With the entire training data used for the development of the parser, the contribution of ambiguous POS tagging was about 0.09[3]. Table 4 summarizes the performance of some of our systems evaluated using section 23 of the Penn Treebank. The overall tendency observed on the development set remained, but the contribution of ambiguous tagging was considerably bigger than in the development set.[4] As can be seen from the table, *prob* was the slowest of all, but the difference is not so big. This may be because Ninomiya's method of HPSG parsing, which outputs the first answer found, worked tolerantly to increase of ambiguity.

4.2 Parsing GENIA corpus

One of the most crucial advantages of separating POS tagging from syntactic parsing is that the domain adaptation of such systems can be partly achieved by just changing POS taggers. In this section, we verify that the effect of ambiguous POS tagging is also remains for domains with which parsing models are not familiar.

The test data we used was the GENIA treebank [Tateisi *et al.*, 2005], which annotates Penn Treebank-style tree structures to sentences of the GENIA corpus. The GENIA corpus consists of the abstracts of biomedical papers adopted from MEDLINE [Campbell *et al.*, 1997]. We trained our POS tagger with the mixture of the WSJ part of the Penn Treebank, the Penn BioIE corpus [Kulick *et al.*, 2004], and the GENIA corpus, which consist of 38219, 29422, and 18508 sentences, respectively. Following Hara et. al. [2005], we adopted 50 abstracts (467 sentences) for the evaluation. Performance of the *single* tagger on these sentences was 98.27%.

The result is shown in Table 5. *single-Penn* and *prob-Penn* are *single* and *prob* taggers trained on the Penn Treebank, and *single-BIO* and *prob-BIO* are those trained on the biomedical domain (i.e., the Penn BioIE corpus and the GENIA corpus). Compared to the results with gold POS tags, the use of automatic taggers trained on a different domain degraded the performance by about 4 to 5 points in the F1 measure. This drop was recovered using well-tuned taggers, and remarkably, the results of *prob-BIO* were only 0.36 points lower than the upper bound.[5] The difference in processing time was, again, not so significant.

In summary, we showed that the adaptation of POS taggers can significantly improve the performance of parsers in new domains, and the use of ambiguous POS tagging can extend the improvement further.

5 Related Work

Charniak et al. [1996] investigated the use of POS taggers that output multiple tags for parsing and concluded that single taggers are preferable, because the accuracy of the tag sequences selected by parsers did not improve significantly, while the increase in computational cost is considerable. Watson [2006] revisited the same task and observed that in terms of the accuracy of parsing, multiple taggers improve the performance significantly. Our results show that the taggers should pass to the parser not only multiple answers, but also probability values for each candidate. Watson's results also imply that appropriate parsing strategies can make the increase of computational cost not as problematic as Charniak et al. suggested. Our results again agree with Watson's at this point.

Clark et al. [2004] introduced supertaggers that output multiple supertag candidates, and the taggers were used as a front end of their CCG parser. Because the training of supertaggers require corpora more deeply annotated than POS tagged ones, their method should have difficulties if we exploit taggers for domain adaptation.

Adaptation of general parsers to a biomedical domain has already attracted several researchers. Because the biggest difference between general English and biomedical literature is their vocabularies, use of lexical resources and tools is a natural approach. Lease et al. [2005] modified Charniak's parser [1999] to make it conform to GENIA's tagging convention and made use of biomedical dictionaries to restrict the parser output. They obtained an impressive error reduction of 14.2% and expected that further improvement is possible using named-entity recognizers.

We can see that the performance shown in Section 4.2 is significantly lower than those in 4.1. This suggests that changing POS taggers can only partially achieve domain adaptation, so we should consider training syntactic disambiguation models on the target domain if we desire further improvement. This is what Hara et al. [2005] explored, and their results suggested that a small number of in-domain data could bring considerable improvements of the performance if out-of-domain data was as large as the Penn Treebank. Because their evaluation was done with POS tagged sentences, whether this is still the case for the task of parsing raw texts is an open question

6 Conclusion

In this paper, we demonstrated that the accuracy of parsers could be improved by allowing POS taggers to output multiple answers for some words. When the performance of the parser was evaluated in terms of precision/recall of predicate argument relations, the performance of the parser with the

[3] Though this figure alone seems not to be significant, we will show that the results of the same experiment using the test set have significant improvements.

[4] The improvement of *prob* over *single* is significant with $p < 10^{-4}$ according to stratified shuffling tests [Cohen, 1995].

[5] The improvement, in terms of recall, of *prob* over *single* is significant with $p < 10^{-4}$ according to stratified shuffling tests. The improvement of precision is not significant.

ambiguous POS tagger consistently outperformed that with the conventional unambiguous tagger throughout the experiments, and this was also the case for a domain that was not used to train the parser. If the goal is to optimize the accuracy of predicate-argument structures, we can say ambiguous tagging is profitable, and introducing ambiguous tagging without harming the capability of domain adaptation is possible.

One apparent deficiency of our system is that it cannot take into account the dependencies among outputs of the POS taggers. Perhaps we can achieve further improvement by running separate parsing processes for each candidate POS sequence that is output by the POS tagger, but this approach might not be acceptable considering computational cost.

References

[Campbell *et al.*, 1997] K. Campbell, D. Oliver, and E. Shortliffe. The Unified Medical Language System: Toward a Collaborative Approach For Solving Terminological Problems, 1997.

[Charniak and Johnson, 2005] E. Charniak and M. Johnson. Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In *Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, June 2005.

[Charniak *et al.*, 1996] Eugene Charniak, Glenn Carroll, John Adcock, Anthony R. Cassandra, Yoshihiko Gotoh, Jeremy Katz, Michael L. Littman, and John McCann. Taggers for Parsers. *Artificial Intelligence*, 85(1-2):45–57, 1996.

[Charniak, 1999] Eugene Charniak. A Maximum-Entropy-Inspired Parser. Technical Report CS-99-12, 1999.

[Chen and Rosenfeld, 1999] S. Chen and R. Rosenfeld. A Gaussian prior for smoothing maximum entropy models. In *Technical Report CMUCS*, 1999.

[Clark and Curran, 2004] Stephen Clark and James Curran. The importance of supertagging for wide-coverage CCG parsing. In *20th International Conference on Computational Linguistics*, 2004.

[Cohen, 1995] Paul R. Cohen. *Empirical methods for artificial intelligence*. MIT Press, Cambridge, MA, USA, 1995.

[Hara *et al.*, 2005] Tadayoshi Hara, Yusuke Miyao, and Jun'ichi Tsujii. Adapting a probabilistic disambiguation model of an HPSG parser to a new domain. In *Natural Language Processing IJCNLP 2005*, volume 3651 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag GmbH, 2005.

[Kulick *et al.*, 2004] S. Kulick, A. Bies, M. Liberman, M. Mandel, R. McDonald, M. Palmer, A. Schein, and L. Ungar. Integrated annotation for biomedical information extraction. In *Proc. of HLT/NAACL 2004*, 2004.

[Lease and Charniak, 2005] Matthew Lease and Eugene Charniak. Parsing Biomedical Literature. In *IJCNLP*, pages 58–69, 2005.

[Marcus *et al.*, 1994] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2), 1994.

[Miyao *et al.*, 2004] Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. Corpus-oriented Grammar Development for Acquiring a Head-driven Phrase Structure Grammar from the Penn Treebank. In *Proc. of IJCNLP-04*, 2004.

[Ninomiya *et al.*, 2005] Takashi Ninomiya, Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Tsujii. Efficacy of Beam Thresholding, Unification Filtering and Hybrid Parsing in Probabilistic HPSG Parsing. In *Proc. of IWPT 2005*, pages 103–114, 2005.

[Ohta *et al.*, 2002] Tomoko Ohta, Yuka Tateisi, Hideki Mima, and Jun'ichi Tsujii. GENIA Corpus: an Annotated Research Abstract Corpus in Molecular Biology Domain. In *Proceedings of the Human Language Technology Conference (HLT 2002)*, March 2002.

[Pollard and Sag, 1994] Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, 1994.

[Tateisi *et al.*, 2005] Yuka Tateisi, Akane Yakushiji, Tomoko Ohta, and Jun'ichi Tsujii. Syntax Annotation for the GENIA corpus. In *Proceedings of the IJCNLP 2005, Companion volume*, pages 222–227, October 2005.

[Tsuruoka and Tsujii, 2005] Yoshimasa Tsuruoka and Jun'ichi Tsujii. Bidirectional Inference with the Easiest-First Strategy for Tagging Sequence Data. In *Proceedings of HLT/EMNLP 2005*, pages 467–474, 2005.

[Watson, 2006] R. Watson. Part-of-speech Tagging Models for Parsing. In *Proc. of CLUK 2006*, 2006.