Prepositional Phrase Attachment in Shallow Parsing

Vincent Van Asch
CLiPS - Computational Linguistics
University of Antwerp
Prinsstraat 13
B-2000 Antwerpen, Belgium
Vincent.VanAsch@ua.ac.be

Walter Daelemans
CLiPS - Computational Linguistics
University of Antwerp
Prinsstraat 13
B-2000 Antwerpen, Belgium
Walter.Daelemans@ua.ac.be

Abstract

In this paper we extend a shallow parser [6] with prepositional phrase attachment. Although the PP attachment task is a well-studied task in a discriminative learning context, it is mostly addressed in the context of artificial situations like the quadruple classification task [18] in which only two possible attachment sites, each time a noun or a verb, are possible. In this paper we provide a method to evaluate the task in a more natural situation, making it possible to compare the approach to full statistical parsing approaches. First, we show how to extract anchor-pp pairs from parse trees in the GENIA and WSJ treebanks. Next, we discuss the extension of the shallow parser with a PP-attacher. We compare the PP attachment module with a statistical full parsing approach [4] and analyze the results. More specifically, we investigate the domain adaptation properties of both approaches (in this case domain shifts between journalistic and medical language).

Keywords

prepositional phrase attachment, shallow parsing, machine learning of language

1 Introduction

Shallow parsing (also called partial parsing) is an approach to language processing that computes a basic analysis of sentence structure rather than attempting full syntactic analysis.

Originally defined by Abney [1] as a task to be solved with handcrafted regular expressions (finite state methods) and limited to finding basic (non-recursive) phrases in text, the label shallow parsing has meanwhile broadened its scope to machine learning methods and to a set of related tasks including part of speech tagging, finding phrases (chunking), clause identification, grammatical role labeling, etc. Especially the machine learning approach to shallow parsing, pioneered by Ramshaw and Marcus [17] has been investigated intensively, in part because of the availability of benchmark datasets and competitions (CoNLL shared tasks 1999 to 2001)[1].

[1] See `http://ifarm.nl/signll/conll/`

It has been argued in [10] and by others that full parsing often provides too much (or not enough) information for some frequent natural language processing tasks. For example, for information retrieval, finding basic NPs and VPs is arguably sufficient, and for information extraction and other text mining tasks, finding syntactic-semantic relations between verbs and base NPs (who did what when and where) is more important than having an elaborate configurational syntactic analysis, provided this shallow analysis can be computed in a deterministic, efficient, robust, and accurate way. Another advantage is that the modules in a machine learning based shallow parser can be trained independently, and allow the inclusion of more information sources (input features) than is possible in statistical parsing (because of sparse data problems). This flexibility in feature engineering, inherent in discriminative, supervised learning approaches to shallow parsing should make the approach more flexible, e.g. when engineering features robust for domain shifts.

However, a shallow approach also has its shortcomings, an important one being that prepositional phrases, which contain important semantic information for interpreting events, are left unattached. Furthermore, while statistical full parsing used to be more noise-sensitive and less efficient than shallow parsing, that is no longer necessarily the case with recent developments in parse ranking.

In this paper, we extend an existing memory based shallow parser, MBSP [5, 6], with a machine learning based prepositional phrase attachment module, and compare it to PP attachment in a state of the art statistical parser. The machine learning method chosen is memory-based learning. We also investigate the ability of this Memory-based PP attachment (MBPA) to cope with the problem of domain adaptation, i.e. the often dramatic decrease in accuracy when testing a trained system on data from a domain different from the domain of the data on which it was trained.

The remainder of this paper starts with an explanation of how the corpus is prepared in order to use it for PP attachment, Section 2. In Section 3 we explain the architecture of the memory-based PP-attacher. Section 4 discusses the experiments and shows the results. In this section we also compare our system to a statistical parser, the Collins parser [3, 4]. An overview of related work can be found in Section 5. Finally, Section 6 concludes this paper and discusses options for further research.

International Conference RANLP 2009 - Borovets, Bulgaria, pages 12–17

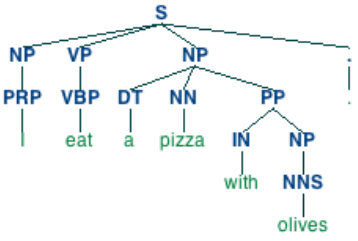
2 Data preparation

In this section, we explain the extraction of the training and test data and the algorithm used to create instances from treebanks.

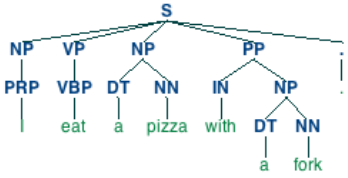
2.1 Training and test data

The memory-based PP-attacher is trained on sections 2 through 21 of the Penn Treebank 2 Wall Street Journal corpus (WSJ) [14]. The development of the system is done using the first 2000 PPs of sections 0-1 of WSJ. Evaluation of the system is done on the next set of 2000 PPs and additional evaluation is done using the first 2000 PPs of the GENIA corpus [20].

The corpora used for training and testing consist of tree structures representing the syntactic structure of sentences, as shown in Figures 1a and 1b. We transform the trees into a flat representation in order to be able to define one unique attachment site (anchor) for every prepositional phrase (PP). A flat representation of an anchor-PP pair consists of a pair of indices. The first element of the pair is the index in the sentence of the anchor; the second element is the index of the preposition. Word count starts at zero. For the sentence in Figure 1a the representation is (3, 4). For the sentence in Figure 1b the representation is (1, 4).



(a) I eat a pizza with olives.



(b) I eat a pizza with a fork.

Fig. 1: Example tree structures

The example tree structures in Figure 1 are relatively straightforward to rewrite into a flat representation.

The basic setup had to be extended by rules for specific cases. A good example is conjunction. In the sentence:

I see cats on the roof and behind the windows.

the parent node of the PP-nodes is a node that also holds the conjunction. Therefore, in this case the algorithm does not take the sibling of the PP-node but it takes a sibling of the parent node of the PP-node.

The extraction algorithm yields 8933 prepositions from sections 0-1 of the WSJ corpus. For 1.95% of the PP-nodes in those two sections no anchor is found. For sections 2-21 1.99% of the 95,955 PP-nodes remain without an anchor. Some anchor-PP pairs are removed in a post-processing step because we limit the task to preposition-NP PPs and disregard preposition-ADJP sequences.

Table 1 shows the chunk type distribution of the anchors. A fairly equal amount of the anchors are nouns and verbs. A minor part has an adjective, comparative adjective or something else as the anchor point.

NP	50.5%
VP	45.8%
Other	3.7%

Table 1: The distribution of the anchors among the chunk types

2.2 Extracting chunks and prepositional phrases

The memory-based PP-attacher (MBPA) is defined as a module within a shallow parser [6]. The MBPA is trained on the WSJ, and it needs chunk and pos tag information from other modules in that shallow parser. In order to prevent indirect contamination of the training data with test data, we retrained the modules of the shallow parser delivering input to the PP attachment module on Wall Street Journal sections 2-21, using the script of the 2000 CoNLL shared task to extract IOB-style chunks from WSJ trees.

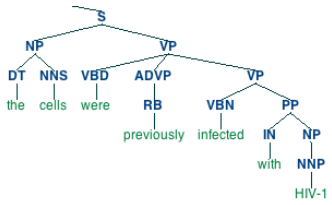


Fig. 2: A tree structure with undetermined cut-off

Converting syntactic trees into a flat representation introduces approximation errors. Figure 2 is an illustration of the problems encountered when looking for syntactic phrases. It is unclear which node should be used as the break point. Since the evaluation is based on chunks, the decisions made in the flattening step may have an influence on the final results. To minimize the bias we use the algorithm that was used to prepare the training data for the memory-based chunker to extract chunks from the syntactic trees output by Collins.

Table 2 shows the results of the comparison between the shallow parser and the Collins parser. The table shows the scores at chunk level. A chunk is correctly identified if it has the same label and it spans the same words as the gold standard chunk. When either the label or the span is not correct, the chunk is a false

	Our system			Collins			
	prec	recall	f-score	prec	recall	f-score	share
-	0.97	0.97	0.97	0.96	0.88	0.92	20.76%
ADJP	0.79	0.74	0.76	0.76	0.77	0.76	1.41%
ADVP	0.84	0.83	0.83	0.86	0.81	0.83	2.75%
CONJP	0.56	0.80	0.66	0.60	0.60	0.60	0.04%
INTJ	0.67	0.20	0.31	0.40	0.20	0.27	0.02%
LST	1.00	0.53	0.70	0.88	0.47	0.61	0.03%
NP	0.94	0.95	0.94	0.93	0.92	0.93	41.21%
PreP	0.97	0.98	0.97	0.97	0.97	0.97	15.57%
PRT	0.77	0.79	0.78	0.79	0.87	0.83	0.37%
SBAR	0.88	0.88	0.88	0.89	0.91	0.90	1.83%
UCP	0.00	0.00	0.00	0.00	0.00	0.00	0.00%
VP	0.94	0.93	0.93	0.83	0.88	0.85	16.01%
weighted mean	0.94	0.95	0.94	0.92	0.91	0.91	100%

Table 2: Results of chunking with our system and Collins

positive, when a chunk in the gold standard is not present in the system's output it is considered a false negative. The shallow parser chunking module and Collins perform comparably well although the former has slightly better results. This implies that there is no reason not to use the memory-based chunker as the basis for the PP-attacher. The PreP chunk in Table 2 is not equal to a PP. In this paper, the label PP is used for the combination of a preposition and a noun phrase, the PreP chunk is a chunk that consists of one or more prepositions only.

The logical first step in finding anchors for prepositional noun phrases is finding the PPs. When extracting the anchor-PP pairs (see Section 2.1) PPs are recognized by the label of the nodes. Since there are no nodes in the input of the MBPA a different strategy is used. PPs are retrieved by a regular expression-like algorithm. All *preposition NP* sequences are considered to be PPs. There are two exceptions to this regular expression rule. Sequences like *preposition " NP* ('in "very modest amounts"') and *preposition VBG NP* ('in making paper') are also considered PPs.

2.3 Creating the instances

The core of the PP-attacher is a memory-based machine learner (supervised, classification-based learning). Every PP found by the algorithm discussed in the previous subsection is a trigger for creating several instances. One instance is created for every combination of the PP in focus and a candidate-anchor. Candidate-anchors are the NPs and VPs of the sentence that are not part of the PP itself. For example,

I eat a pizza with olives.

will induce the creation of 3 instances. One instance for the combination *I–with*, one for the combination *eat–with* and one for the combination *pizza–with*. In the classification task, the machine learner will have to decide whether an instance suggests a true anchor or not. The advantage of this approach is that the machine learner can investigate every possible anchor for its validity and not only the VP and NP in front of the PP. The drawback of this approach is that we have skewed data. There will be many more negative instances in the instance base as can be seen in Table 3.

The features of the instances were chosen on the basis of previous work in machine learning based PP attachment and related tasks: the number of commas between the PP and the candidate anchor, the number of other punctuation marks between the PP and the candidate anchor, the token-distance between the PP and the candidate anchor, the preposition if the candidate anchor is an NP that is part of a PP, the lemma and POS-tag of the last token of the candidate anchor, the lemma and POS tag of the token just in front of the preposition of the PP, the lemma of the preposition, the lemma and POS-tag of the last token of the NP of the PP, the number of NPs between the candidate anchor and the PP, the number of PPs between the candidate anchor and the PP, and NP anchor tendency. If a preposition is for 10% of the cases in the training corpus attached to an NP, the NP anchor tendency will be 10.

	count	percentage
NP	43,049	6.0%
VP	42,285	5.9%
NONE	630,720	88.1%
TOTAL	716,054	100%

Table 3: Distribution of classes in sections 2-21 of WSJ

3 The memory-based PP-attacher

The input of the MBPA module consists of sentences tagged with Part-of-Speech tags, IOB-chunk tags and the lemmata for every word by other modules of the shallow parser. The output of the system is a set of pairs, where each pair represents a PP with its corresponding attachment point.

3.1 Machine Learning Approaches

The machine learning approach we chose is memory-based learning, as implemented in the open source software package TiMBL[2]. We used version 6.1 [7]. Memory-based learning (MBL) is a supervised inductive algorithm for learning classification tasks based on the k-nearest neighbor classification rule.

[2] Available from `http://ilk.uvt.nl`.

However, the machine learner used to train the PP attachment module can be any algorithm that assigns classes to instances. For comparison, we also implemented a system using maxent, an eager learning method, as the machine learner. Maxent[3] is an implementation of maximum entropy modeling. It is a general purpose machine learning framework that constructs a model that captures the distribution of outcomes for a given context in the training data [13].

3.2 Heuristic decision making

If the classifier would be able to predict the anchors with 100% accuracy, no post-processing would be necessary. Only one instance, the one with the correct anchor, would carry a positive class label and all other instances would have a negative classlabel. But due to misclassifications, multiple or no anchors may be identified by the machine learner. An extra step ensures that the system presents one unique anchor for every PP. In case the PP in focus is classified positively with exactly one anchor, that anchor-PP pair is returned as the solution. The other possible outcomes of the classification step are:

1. No instance for the PP in focus got a positive class ⇒ There is no anchor identified yet.
2. More than one instance for the PP in focus got a positive class ⇒ We have a decrease of possible anchors but still no unique anchor.

To resolve these cases, we need an extra step. A baseline algorithm is used if no anchor has been found (case 1). If there are still several candidate anchors to choose from, the entropy is used to reduce the set of candidates to just one unique anchor (case 2).

Baseline

The baseline is computed using a simple rule-based PP-attacher. If a rule fails, the next rule in the hierarchy is checked. The hierarchy of the rules of the baseline algorithm is:

1. Take the nearest NP or VP in front of the PP. We take an NP if in the training corpus the preposition of the PP is associated more frequently with NP anchors. Otherwise we take the VP anchor.
2. Take the nearest anchor in front of the PP.
3. Take the nearest VP anchor behind the PP.
4. Take the nearest anchor behind the PP.

Entropy

When the classification step results in a draw, the candidate with the lowest entropy will be the anchor. The entropy is calculated using the distribution of the classes of the nearest neighbors. When processing an instance with TiMBL we can obtain the (weighted) distribution of the classes of instances in memory that are in the neighborhood of the test instance. The entropy of an instance is computed using this distribution. The formula is:

$$-\sum_{i=1}^{n} P(c_i) log_2(P(c_i)) \quad (1)$$

with
- n: the total number of different classes in the distribution
- $P(c_i)$: $\frac{\text{the number of instances in the neighborhood with class i}}{\text{the total number of instances in the neighborhood}}$

The memory-based learner has an optional weighing parameter. If weighing is applied, $P(c_i)$ is calculated using the weighted counts instead of the plain counts.

The candidate anchor with the lowest entropy is regarded as the correct and unique anchor for a given PP. The rationale behind this decision is that choosing the candidate anchor with the lowest entropy means choosing the anchor for which the classifier was the most certain of its class.

Post-processing rules

For completeness, we also mention two post-processing rules that are applied because of some idiosyncrasies in the treebank data and common errors of the attacher-system. These rules are:

- If there are 2 consecutive prepositions the second preposition will always be attached to the first.
- If a PP is attached to a noun phrase anchor between parentheses, and the PP is not inside the parentheses, then the noun phrase before the parentheses becomes the anchor. This is done because the NP between the parentheses is most of the time an elaboration/abbreviation of the noun phrase in front.

4 Experiments and results

We train 4 systems (baseline, MBL, maxent and a statistical parser) on sections 2-21 of WSJ. In the first set of experiments, Section 4.1, we used the trained systems to attach the second set of 2000 PPs of WSJ sections 0-1 to their anchors. In the second set of experiments, Section 4.2, we reuse the trained systems to attach 2000 PPs extracted from the GENIA corpus to their anchors.

For comparison, we parse every sentence fed to the MBPA with a state-of-the-art statistical parser, viz. Bikel's implementation of the Collins parser. Applying the PP extraction algorithm from Section 2.1 on the syntactic trees output by Collins will yield all anchor-PP pairs needed for evaluation.

4.1 Training and testing on WSJ corpus

Table 4 shows the accuracies of systems trained and tested on the WSJ corpus. We performed a χ^2 statistical test and found that maxent, MBL and Collins all significantly ($p < 5\%$) differ from the baseline system. The variation between the accuracies of the machine learning systems is not found to be significant. The 'not retrieved' column is due to POS, chunking and/or syntactic tree errors in the pre-processing step. Looking at the first 200 errors MBPA and Collins made,

[3] Available from `http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`.

shows that MBPA tends to misattach PPs at the start of the sentence. E.g. 'At the meeting, etc.' Collins has a higher number of 'not retrieved' PPs because it often inserts adverbs and quotes into the PP. E.g. 'by commenting [PP publicly on the case]'.

Table 5 shows the accuracies of the steps involved in the PP-attacher system. The 91.21% accuracy for TiMBL means that for 79.05% of the PPs TiMBL identified a unique anchor and that 91.21% of these anchors were correct. For 11.7% of the PPs, TiMBL could not find an anchor so baseline had to take over. 44.4% of the PPs baseline handled, were correctly assigned to their head. For 6.35% of the PPs, TiMBL found multiple anchors. Entropy handled these cases and found the correct anchor for 67% of them. After finding a unique anchor for every PP, the system made sure that for a sequence of PPs the last PP got attached to the previous. This happened for 1.2% of the PPs.

	Correct	Incorrect	Not retrieved
Baseline	69.85%	25.45%	1.70%
Collins	83.85%	13.30%	2.85%
MBL	82.65%	15.65%	1.70%
Maxent	81.40%	16.90%	1.70%

Table 4: The accuracies on 2000 anchor-PP pairs from WSJ

	Accuracy (%)	proportion (%)
TiMBL	91.21	79.05
Baseline	44.44	11.70
Entropy	66.93	6.35
consecutive preps	91.67	1.20

Table 5: The accuracies split into the different steps of MBPA

These results show that it is possible to develop a PP attachment module using supervised machine learning techniques, integrated as a module within a shallow parser, and reach state of the art accuracy when comparing to one of the best statistical parsers available today. This way the semantically important information carried by relations between PPs and their anchors becomes available to shallow analysis approaches with their advantages in terms of efficiency and flexibility. We were not able to find significant differences between lazy (TiMBL) and eager (Maxent) learning approaches for this problem.

4.2 Domain Adaptation

Adaptation of NLP systems to domains different from the one on which they were developed, is a crucial functionality to make the technology useful. Accuracy of systems deteriorates enormously when moving between different domains. Accuracy drops of 20 to 40 percent are not uncommon for tasks such as parsing, named-entity recognition, word sense disambiguation, and machine translation when moving from the source domain to the new target domain. Usually, no or limited labeled data exists for the target domain. We evaluated the PP attachment systems trained on the WSJ using 2000 anchor-PP pairs from the GENIA corpus. The WSJ corpus consists of news articles on mainly financial issues in contrast to the medical abstracts of the GENIA corpus. Although one cannot always clearly say where the boundaries between domains are, we assume that medical and financial texts are sufficiently different. Table 6 shows the accuracies for the different systems. We performed these experiments to gain more insight into the relative robustness of different approaches to PP attachment to domain shifts. The χ^2 test gave the same results as in the previous section: all systems perform significantly better than baseline but do not differ significantly from each other. As expected, the accuracy significantly decreases compared to the same-domain experiments.

	Correct	Incorrect	Not retrieved
Baseline	69.20%	27.00%	3.80%
Collins	78.80%	19.35%	1.85%
MB-attacher	77.70%	19.10%	3.20%
Maxent-attacher	77.15%	19.65%	3.20%

Table 6: The accuracies on 2000 anchor-PP pairs from GENIA

Table 7 shows the robustness of the systems to a domain shift from mainly financial to medical language. The higher the ratio, the lower the drop of accuracy. As can be seen, if no learning is involved (baseline) the system is most robust. A shallow approach is at an advantage here compared to full parsing because it allows more flexible feature engineering to alleviate the domain adaptation problem (e.g. by adding or removing specific lexical, syntactic, and semantic features to the classifiers. This is in general more difficult in a statistical parsing approach because of data sparseness.

5 Related work

As Atterer and Schütze [2] state, the classic formulation of the task of PP attachment, as defined in [19] and [11], is a simplification. The classic formulation uses quadruples (v, n_1, p, n_2) that were manually selected from a corpus. This helps performance of PP attachment systems but for a natural language application these quadruples are not available. In their experiments, the PP attachment systems they evaluated did not significantly improve on a state-of-the-art parser, Collins parser [3, 4]. The PP-attacher system in this paper does not make use of this simplified representation and therefore can be regarded as more fit for the task of natural language PP attachment.

Foth and Menzel [9] implemented a PP attachment predictor for German and incorporated it in a rule-based dependency parser [8]. The PP attachment pre-

	Ratio
Baseline	0.991
Collins	0.940
MB-attacher	0.940
Maxent-attacher	0.948

Table 7: The ratio of the accuracies GENIA/WSJ

dictor was based on a collocation measure and significantly increased the accuracy on the PP attachment subtask. Basically, the collocation measure is a number indicating whether a word and a preposition co-occur more often than chance. In this paper, we did not compute a collocation measure but for the NP anchor tendency feature we draw upon the same underlying idea.

As noticed in [21], the algorithm used to extract the pairs from the corpus has an influence on the accuracies reported, and makes comparing of results among systems for different corpora and languages difficult. The noun attachment rate and the extraction procedure are two important features when comparing results obtained using different corpora. As we tested our system and Collins' using the same training and test data, the comparison is reliable.

Other memory-based approaches to the problem of PP attachment can be found in [12] and [22]. [12] uses a memory-based PP-attacher combined with the MALTParser [16]. They showed that the dependency parser could not fully benefit from the separate PP-attacher although the PP-attacher module assigns PPs to their heads with a reasonable accuracy. The features they use for their PP-attacher system are lemmata, POS-tags and distances between words.

In their paper, [15] mainly focus on how to disambiguate between argument and adjunct PPs, but they provide an alternative way of extracting PPs from the WSJ treebank. Their final data contains quadruples and sets of multiple PP sequences.

6 Conclusion

In this paper we compared a shallow parsing approach to PP-attachment with a state of the art full parser. We used a flat representation of prepositional phrases and their associated attachment sites to train a machine learner for the PP attachment task. We showed that a memory-based approach can obtain results for the PP attachment task comparable to a state-of-the-art full parser. The PP attachment system proposed in this article is not limited to the classical quadruple approximation of the PP attachment task and therefore the system can be combined with any (shallow) parser that assigns part-of-speech tags, lemmata and chunk tags to natural language sentences. Such a PP attachment module can also be easily added to a full parser as a reattacher.

The shallow memory-based PP attachment module is fairly robust to a domain shift of the testing corpus but further research should focus on how to improve the robustness. Building a more robust PP attachment system would legitimate the use of the PP-attacher system as a reattachment module in any full parser.

Acknowledgements

This research was made possible through financial support from the University of Antwerp (BIOGRAPH GOA-project).

References

[1] S. Abney. Parsing by chunks. In *Principle-Based Parsing*, pages 257–278. Kluwer Academic Publishers, 1991.

[2] M. Atterer and H. Schütze. Prepositional phrase attachment without oracles. *Computational Linguistics*, 33(4):469–476, 2007.

[3] D. Bikel. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–512, 2004.

[4] M. Collins. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637, 2003.

[5] W. Daelemans, S. Buchholz, and J. Veenstra. Memory-based shallow parsing. In *Proceedings of CoNLL-99*, pages 53–60, Bergen, Norway, 1999.

[6] W. Daelemans and A. van den Bosch. *Memory-Based Language Processing*. Studies in Natural Language Processing. Cambridge University Press, Cambridge, 2005.

[7] W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. Timbl: Tilburg memory-based learner, version 6.1. Technical Report ILK 07-07, Tilburg University, 2007.

[8] K. Foth, M. Daum, and W. Menzel. Parsing unrestricted german text with defeasible constraints. In *Constraint Solving and Language Processing*, volume 3438 of *Lecture Notes in Computer Science*, pages 140–157. Springer, Berlin/Heidelberg, 2005.

[9] K. Foth and W. Menzel. The benefit of stochastic PP attachment to a rule-based parser. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 223–230, 2006.

[10] J. Hammerton, M. Osborne, S. Armstrong, and W. Daelemans. Introduction to special issue on machine learning approaches to shallow parsing. *Journal of Machine Learning Research*, 2(Mar):551–558, 2002.

[11] D. Hindle and M. Rooth. Structural ambiguity and lexical relations. *Computational Linguistics*, 19(1):103–120, 1993.

[12] S. Kübler, S. Ivanova, and E. Klett. Combining dependency parsing with PP attachment. In *Fourth Midwest Computational Linguistics Colloquium*, 2007.

[13] Z. Le. Maximum Entropy Modeling Toolkit for Python and C++, 2004. Version 20061005.

[14] M. Marcus, G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. The Penn treebank: annotating predicate argument structure. In *HLT '94: Proceedings of the workshop on Human Language Technology*, pages 114–119, Morristown, NJ, USA, 1994. Association for Computational Linguistics.

[15] P. Merlo and E. E. Ferrer. The notion of argument in prepositional phrase attachment. *Computational Linguistics*, 32(3):341–377, 2006.

[16] J. Nivre. *Inductive Dependency Parsing*, volume 34 of *Text, Speech and Language Technology*. Springer, 2006.

[17] L. Ramshaw and M. Marcus. Text chunking using transformation-based learning. In D. Yarovsky and K. Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94. Association for Computational Linguistics, 1995.

[18] A. Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. PhD thesis, Computer and Information Science, University of Pennsylvania, 1998.

[19] A. Ratnaparkhi, J. Reynar, and S. Roukos. A maximum entropy for prepositional phrase attachment. In *Workshop on Human Language Technology*, pages 250–255, 1994.

[20] Y. Tateisi, A. Yakushiji, T. Ohta, and J. Tsujii. Syntax annotation for the genia corpus. In *Proceedings of the IJCNLP 2005, Companion volume*, pages 222–227, October 2005.

[21] M. Volk. How bad is the problem of PP-attachment? A comparison of English, German and Swedish. In *Proceedings of the Third ACL-SIGSEM Workshop on Prepositions*, pages 81–88, 2006.

[22] J. Zavrel, W. Daelemans, and J. Veenstra. Resolving PP attachment ambiguities with memory-based learning. In *Proceedings CoNLL 1997*, pages 136–144, 1997.