Shallow parsing on the basis of words only: A case study

Antal van den Bosch and Sabine Buchholz
ILK / Computational Linguistics and AI
Tilburg University
Tilburg, The Netherlands
{Antal.vdnBosch,S.Buchholz}@kub.nl

Abstract

We describe a case study in which a memory-based learning algorithm is trained to simultaneously chunk sentences and assign grammatical function tags to these chunks. We compare the algorithm's performance on this parsing task with varying training set sizes (yielding learning curves) and different input representations. In particular we compare input consisting of words only, a variant that includes word form information for low-frequency words, gold-standard POS only, and combinations of these. The word-based shallow parser displays an apparently log-linear increase in performance, and surpasses the flatter POS-based curve at about 50,000 sentences of training data. The low-frequency variant performs even better, and the combinations is best. Comparative experiments with a real POS tagger produce lower results. We argue that we might not need an explicit intermediate POS-tagging step for parsing when a sufficient amount of training material is available and word form information is used for low-frequency words.

1 Introduction

It is common in parsing to assign part-of-speech (POS) tags to words as a first analysis step providing information for further steps. In many early parsers, the POS sequences formed the only input to the parser, i.e. the actual words were not used except in POS tagging. Later, with feature-based grammars, information on POS had a more central place in the lexical entry of a word than the identity of the word itself, e.g. MAJOR and other HEAD features in (Pollard and Sag, 1987). In the early days of statistical parsers, POS were explicitly and often exclusively used as symbols to base probabilities on; these probabilities are generally more reliable than lexical probabilities, due to the inherent sparseness of words.

In modern lexicalized parsers, POS tagging is often interleaved with parsing proper instead of being a separate preprocessing module (Collins, 1996; Ratnaparkhi, 1997). Charniak (2000) notes that having his generative parser generate the POS of a constituent's head before the head itself increases performance by 2 points. He suggests that this is due to the usefulness of POS for estimating back-off probabilities.

Abney's (1991) chunking parser consists of two modules: a chunker and an attacher. The chunker divides the sentence into labeled, non-overlapping sequences (chunks) of words, with each chunk containing a head and (nearly) all of its premodifiers, exluding arguments and postmodifiers. His chunker works on the basis of POS information alone, whereas the second module, the attacher, also uses lexical information. Chunks as a separate level have also been used in Collins (1996) and Ratnaparkhi (1997).

This brief overview shows that the main reason for the use of POS tags in parsing is that they provide useful generalizations and (thereby) counteract the sparse data problem. However, there are two objections to this reasoning. First, as naturally occurring text does not come POS-tagged, we first need a module to assign POS. This tagger can base its decisions only on the information present in the sentence, i.e. on the words themselves. The question then arises whether we could use this information directly, and thus save the explicit tagging step. The second objection is that sparseness of data is tightly coupled to the amount of training material used. As training material is more abundant now than it was even a few years ago, and today's computers can handle these amounts, we might ask whether there is now enough data to overcome the sparseness problem for certain tasks.

To answer these two questions, we designed the following experiments. The task to be learned is a shallow parsing task (described below). In one experiment, it has to be performed on the basis of the "gold-standard", assumed-perfect POS taken directly from the training data, the Penn Treebank (Marcus et al., 1993), so as to abstract from a particular POS tagger and to provide an upper bound. In another experiment, parsing is done on the basis of the words alone. In a third, a special encoding of low-frequency words is used. Finally, words and POS are combined. In all experiments, we increase the amount of training data stepwise and record parse performance for each step. This yields four learning curves. The word-based shallow parser displays an apparently log-linear increase in performance, and surpasses the flatter POS-based curve at about 50,000 sentences of training data. The low-frequency variant performs even better, and the combinations is best. Comparative experiments with a real POS tagger produce lower results.

The paper is structured as follows. In Section 2 we describe the parsing task, its input representation, how this data was extracted from the Penn Treebank, and how we set up the learning curve experiments using a memory-based learner. Section 3 provides the experimental learning curve results and analyses them. Section 4 contains a comparison of the effects with gold-standard and automatically assigned POS. We review related research in Section 5, and formulate our conclusions in Section 6.

2 Task representation, data preparation, and experimental setup

We chose a shallow parsing task as our benchmark task. If, to support an application such as information extraction, summarization, or question answering, we are only interested in parts of the parse tree, then a shallow parser forms a viable alternative to a full parser. Li and Roth (2001) show that for the chunking task it is specialized in, their shallow parser is more accurate and more robust than a general-purpose, i.e. full, parser.

Our shallow parsing task is a combination of *chunking* (finding and labelling non-overlapping syntactically functional sequences) and what we will call *function tagging*. Our chunks and functions are based on the annotations in the third release of the Penn Treebank (Marcus et al., 1993). Below is an example of a tree and the corresponding chunk (subscripts on brackets) and function (superscripts on headwords) annotation:

```
((S (ADVP-TMP Once)
    (NP-SBJ-1 he)
    (VP was
        (VP held
            (NP *-1)
            (PP-TMP for
                    (NP three months))
            (PP without
                (S-NOM (NP-SBJ *-1)
                       (VP being
                           (VP charged)
))))) .))
```

[$_{ADVP}$ Once$^{ADVP-TMP}$] [$_{NP}$ he^{NP-SBJ}]
[$_{VP}$ was held$^{VP/S}$] [$_{PP}$ for^{PP-TMP}]
[$_{NP}$ three months^{NP}] [$_{PP}$ without^{PP}]
[$_{VP}$ being charged$^{VP/S-NOM}$] .

Nodes in the tree are labeled with a syntactic category and up to four function tags that specify grammatical relations (e.g. SBJ for subject), subtypes of adverbials (e.g. TMP for temporal), discrepancies between syntactic form and syntactic function (e.g. NOM for non-nominal constituents functioning nominally) and notions like topicalization. Our chunks are based on the syntactic part of the constituent label. The conversion program is the same as used for the CoNLL-2000 shared task (Tjong Kim Sang and Buchholz, 2000). Head words of chunks are assigned a function code that is based on the full constituent label of the parent and of ancestors with a different category, as in the case of VP/S-NOM in the example.

2.1 Task representation and evaluation method

To formulate the task as a machine-learnable classification task, we use a representation that encodes the joint task of chunking and function-tagging a sentence in per-word classification instances. As illustrated in Table 2.1, an instance (which corresponds to a row in the table) consists of the values for all features (the columns) and the function-chunk code for the focus word. The features describe the focus word and its local context. For the chunk part of the code, we adopt the "Inside", "Outside", and "Between" (IOB) encoding originating from (Ramshaw and Marcus, 1995). For the function part of the code, the value is either the function for the head of a chunk, or the dummy value NOFUNC for all non-heads. For creating the POS-based task, all words are replaced by the gold-standard POS tags associated with them in the Penn Treebank. For the combined task, both types of features are used simultaneously.

When the learner is presented with new instances from heldout material, its task is thus to assign the combined function-chunk codes to either words or POS in context. From the sequence of predicted function-chunk codes, the complete chunking and function assignment can be reconstructed. However, predictions can be inconsistent, blocking a straightforward reconstruction of the complete shallow parse. We employed the following four rules to resolve such problems: (1) When an O chunk code is followed by a B chunk code, or when an I chunk code is followed by a B chunk code with a different chunk type, the B is converted to an I. (2) When more than one word in a chunk is given a function code, the function code of the rightmost word is taken as the chunk's function code. (3) If all words of the chunk receive NOFUNC tags, a *prior* function code is assigned to the rightmost word of the chunk. This prior, estimated on the training set, represents the most frequent function code for that type of chunk.

To measure the success of our learner, we compute the precision, recall and their harmonic mean, the F-score[1] with β=1 (Van Rijsbergen, 1979). In the combined function-chunking evaluation, a chunk is only counted as correct when its boundaries, its type and its function are identified correctly.

2.2 Data preparation

Our total data set consists of all 74,024 sentences in the Wall Street Journal, Brown and ATIS Corpus subparts of the Penn Treebank III. We randomized the order of the sentences in this dataset, and then split it into ten 90%/10% partitionings with disjoint 10% portions, in order to run 10-fold cross-validation experiments (Weiss and Kulikowski, 1991). To provide differently-sized training sets for learning curve experiments, each training set (of 66,627 sentences) was also clipped at the following sizes: 100 sentences, 500, 1000, 2000, 5000, 10,000, 20,000 and 50,000. All data was converted to instances as illustrated in Table 2.1. For the total data set, this yields 1,637,268 instances, one for each word or punctuation mark. 62,472 word types occur in the total data set, and 874 different function-chunk codes.

2.3 Classifier: Memory-based learning

Arguably, the choice of algorithm is not crucial in learning curve experiments. First, we aim at measuring relative differences arising from the selection of types of input. Second, there are indications that increasing the training set of language processing tasks produces much larger performance gains than varying among algorithms at fixed training set sizes; moreover, these differences also tend to get smaller with larger data sets (Banko and Brill, 2001).

Memory-based learning (Stanfill and Waltz, 1986; Aha et al., 1991; Daelemans et al., 1999b) is a supervised inductive learning algorithm for learning classification tasks. Memory-based learning treats a set of labeled (pre-classified) training instances as points in a multi-dimensional feature space, and stores them as such in an *instance base* in memory (rather than performing some abstraction over them). Classification in memory-based learning is performed by the k-NN algorithm (Cover and Hart, 1967) that searches for the k 'nearest neighbors' according to the distance function between two in-

[1] $F_\beta = \frac{(\beta^2+1)\cdot \text{precision}\cdot\text{recall}}{\beta^2\cdot\text{precision}+\text{recall}}$

Left context			Focus	Right context			Function-chunk code	
_	_	_	Once	he	was	held	I-ADVP	ADVP-TMP
_	_	Once	he	was	held	for	I-NP	NP-SBJ
_	Once	he	was	held	for	three	I-VP	NOFUNC
Once	he	was	held	for	three	months	I-VP	VP/S
he	was	held	for	three	months	without	I-PP	PP-TMP
was	held	for	three	months	without	being	I-NP	NOFUNC
held	for	three	months	without	being	charged	I-NP	NP
for	three	months	without	being	charged	.	I-PP	PP
three	months	without	being	charged	.	_	I-VP	NOFUNC
months	without	being	charged	.	_	_	I-VP	VP/S-NOM
without	being	charged	.	_	_	_	O	NOFUNC

Table 1: Encoding into instances, with words as input, of the example sentence "Once he was held for three months without being charged ."

stances X and Y, $\Delta(X,Y) = \sum_{i=1}^{n} w_i \; \delta(x_i, y_i)$, where n is the number of features, w_i is a weight for feature i, and δ estimates the difference between the two instances' values at the ith feature. The classes of the k nearest neighbors then determine the class of the new case.

In our experiments, we used a variant of the IB1 memory-based learner and classifier as implemented in TiMBL (Daelemans et al., 2001). On top of the k-NN kernel of IB1 we used the following metrics that fine-tune the distance function and the class voting automatically: (1) The weight (importance) of a feature i, w_i, is estimated in our experiments by computing its *gain ratio* GR_i (Quinlan, 1993). This is the algorithm's default choice. (2) Differences between feature values (i.e. words or POS tags) are estimated by the real-valued outcome of the modified value difference metric (Stanfill and Waltz, 1986; Cost and Salzberg, 1993). (3) k was set to seven. This and the previous parameter setting turned out best for a chunking task using the same algorithm as reported by Veenstra and van den Bosch (2000). (4) Class voting among the k nearest neighbours is done by weighting each neighbour's vote by the inverse of its distance to the test example (Dudani, 1976). In Zavrel (1997), this distance was shown to improve over standard k-NN on a PP-attachment task. (5) For efficiency, search for the k-nearest neighbours is approximated by employing TRIBL (Daelemans et al., 1997), a hybrid between pure k-NN search and decision-tree traversal. The switch point of TRIBL was set to 1 for the words only and POS only experiments, i.e. a decision-tree split was made on the most important feature, the focus word, respectively focus POS. For the experiments with both words and POS, the switch point was set to 2 and the algorithm was forced to split on the focus word *and* focus POS. The metrics under 1) to 4) then apply to the remaining features.

3 Learning Curve Experiments

We report the learning curve results in three paragraphs. In the first, we compare the performance of a plain words input representation with that of a gold-standard POS one. In the second we introduce a variant of the word-based task that deals with low-frequency words. The last paragraph describes results with input consisting of words *and* POS tags.

Words only versus POS tags only As illustrated in Figure 1, the learning curves of both the word-based and the POS-based representation are upward with more training data. The word-based curve starts much lower but flattens less; in the tested range it has an approximately log-linear growth. Given the measured results, the word-based curve surpasses the POS-based curve at a training set size between 20,000 and 50,000 sentences. This proves two points: First, experiments with a fixed training set size might present a misleading snapshot. Second, the amount of training material available today is already enough to make words more valuable input than (gold-standard!) POS.

Low-frequency word encoding variant If TRIBL encounters an unknown word in the test material, it stops already at the decision tree stage and returns the default class without even using the information provided by the context. This is clearly disadvantageous and specific to this choice of al-

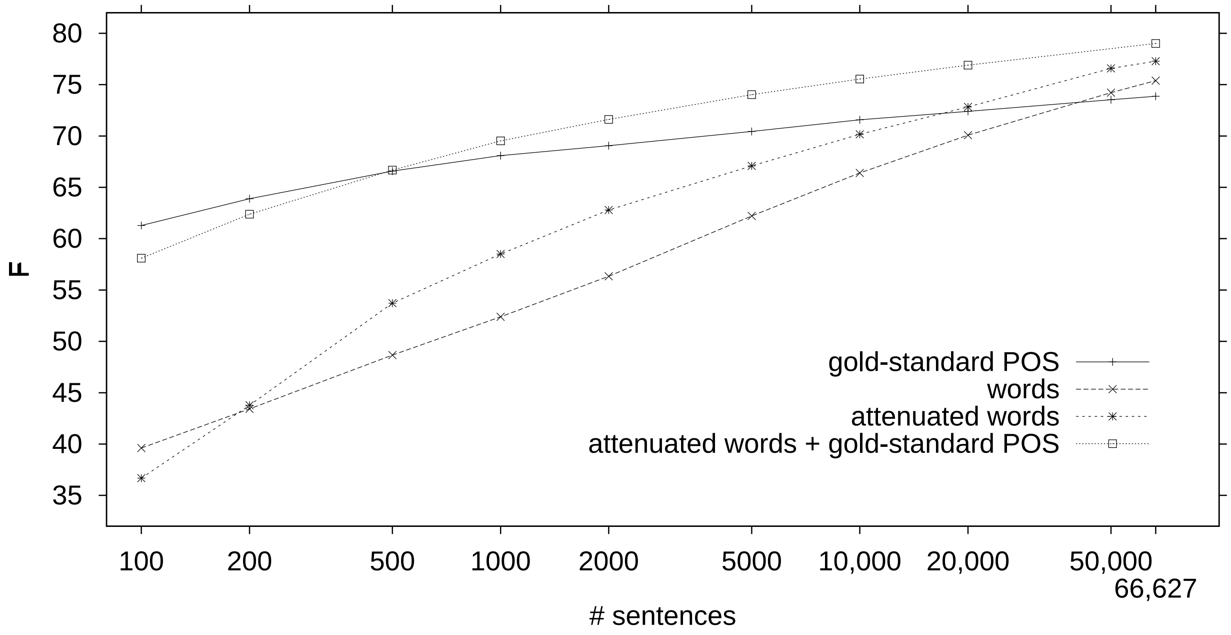

Figure 1: Learning curves of the main experiments on POS tags, words, attenuated words, and the combination of words and POS. The y-axis represents $F_{\beta=1}$ on combined chunking and function assignment. The x-axis represents the number of training sentences; its scale is logarithmic.

gorithm. A more general shortcoming is that the word form of an unknown word often contains useful information that is not available in the present setup. To overcome these two problems, we applied what Eisner (1997) calls "attenuation" to all words occurring ten times or less in training material. If such a word ends in a digit, it is converted to the string "MORPH-NUM"; if the word is six characters or longer it becomes "MORPH-XX" where XX are the final two letters, else it becomes "MORPH-SHORT". If the first letter is capitalised, the attenuated form is "MORPH-CAP". This produces sequences such as A number of MORPH-ts were MORPH-ly MORPH-ed by traders . (A number of developments were negatively interpreted by traders). We applied this attenuation method to all training sets. All words in test material that did not occur as words in the attenuated training material were also attenuated following the same procedure.

The curve resulting from the attenuated word-based experiment is also displayed in Figure 1. The curve illustrates that the attenuated representation performs better than the pure word-based one at all reasonable training set sizes. However the effect clearly diminishes with more training data, so we cannot exclude that the two curves will meet with yet more training data.

Combining words with POS tags Although the word-based curve, and especially its attenuated variant, end higher than the POS-based curve, POS might still be useful *in addition* to words. We therefore also tested a representation with both types of features. As shown in Figure 1, the "attenuated word + gold-standard POS" curve starts close to the gold-standard POS curve, attains break-even with this curve at about 500 sentences, and ends close to but higher than all other curves, including the "attenuated word" curve.

4

Although the performance increase through the addition of POS becomes smaller with more training data, it is still highly significant with maximal training set size. As the tags are the gold-standard tags taken directly from the Penn Treebank, this result provides an upper bound for the contribution of POS tags to the shallow parsing task under investigation. Automatic POS tagging is a well-studied

Input features	Precision	±	Recall	±	F-score	±
gold-standard POS	73.8	0.2	73.9	0.2	73.9	0.2
MBT POS	72.2	0.2	72.4	0.2	72.3	0.2
words	75.4	0.1	75.4	0.1	75.4	0.1
words + gold-standard POS	76.5	0.2	77.1	0.2	76.8	0.2
words + MBT POS	75.8	0.2	76.1	0.1	75.9	0.1
attenuated words	77.3	0.1	77.2	0.2	77.3	0.2
attenuated words + gold-standard POS	78.9	0.2	79.1	0.2	79.0	0.2
attenuated words + MBT POS	77.6	0.2	77.7	0.2	77.6	0.2

Table 2: Average precision, recall, and F-scores on the chunking-function-tagging task, with standard deviation, using the input features words, attenuated words, gold-standard POS, and MBT POS, and combinations, on the maximal training set size.

task (Church, 1988; Brill, 1993; Ratnaparkhi, 1996; Daelemans et al., 1996), and reported errors in the range of 2–6% are common. To investigate the effect of using automatically assigned tags, we trained MBT, a memory-based tagger (Daelemans et al., 1996), on the training portions of our 10-fold cross-validation experiment for the maximal data and let it predict tags for the test material. The memory-based tagger attained an accuracy of 96.7% (± 0.1; 97.0% on known words, and 80.9% on unknown words). We then used these MBT POS instead of the gold-standard ones.

The results of these experiments, along with the equivalent results using gold-standard POS, are displayed in Table 2. As they show, the scores with automatically assigned tags are always lower than with the gold-standard ones. When taken individually, the difference in F-scores of the gold-standard versus the MBT POS tags is 1.6 points. Combined with words, the MBT POS contribute 0.5 points (compared against words taken individually); combined with attenuated words, they contribute 0.3 points. This is much less than the improvement by the gold-standard tags (1.7 points) but still significant. However, as the learning curve experiments showed, this is only a snapshot and the improvement may well diminish with more training data.

A breakdown of accuracy results shows that the highest improvement in accuracy is achieved for focus words in the MORPH-SHORT encoding. In these cases, the POS tagger has access to more information about the low-frequency word (e.g. its suffix) than the attenuated form provides. This suggests that this encoding is not optimal.

5 Related Research

Ramshaw and Marcus (1995), Muñoz et al. (1999), Argamon et al. (1998), Daelemans et al. (1999a) find NP chunks, using Wall Street Journal training material of about 9000 sentences. F-scores range between 91.4 and 92.8. The first two articles mention that words and (automatically assigned) POS together perform better than POS alone. Chunking is one part of the task studied here, so we also computed performance on chunks alone, ignoring function codes. Indeed the learning curve of words combined with gold-standard POS crosses the POS-based curve before 10,000 sentences on the chunking subtask.

Tjong Kim Sang and Buchholz (2000) give an overview of the CoNLL shared task of chunking. The types and definitions of chunks are identical to the ones used here. Training material again consists of the 9000 Wall Street Journal sentences with automatically assigned POS tags. The best F-score (93.5) is higher than the 91.5 F-score attained on chunking in our study using attenuated words only, but using the maximally-sized training sets. With gold-standard POS and attenuated words we attain an F-score of 94.2; with MBT POS tags and attenuated words, 92.8. In the CoNLL competition, all three best systems used combinations of classifiers instead of one single classifier. In addition, the effect of our mix of sentences from different corpora on top of WSJ is not clear.

Ferro et al. (1999) describe a system for finding grammatical relations in automatically tagged and manually chunked text. They report an F-score of 69.8 for a training size of 3299 *words* of elementary school reading comprehension tests. Buchholz et al. (1999) achieve 71.2 F-score for grammatical relation assignment on automatically tagged and chunked text after training on about 40,000 Wall Street Journal sentences. In contrast to these studies, we do not chunk before finding grammatical relations; rather, chunking is performed simultaneously with headword function tagging. Measuring F-scores on the correct assignment of functions to headwords in our study, we attain 78.2 F-score using words, 80.1 using attenuated words, 80.9 using attenuated words combined with gold-standard POS, and 79.7 using attenuated words combined with MBT POS (which is slightly worse than with attenuated words only). Our function tagging task is easier than finding grammatical relations as we tag a headword of a chunk as e.g. a subject *in isolation* whereas grammatical relation assignment also includes deciding which verb this chunk is the subject of. Aït-Mokhtar and Chanod (1997) describe a sequence of finite-state transducers in which function tagging is a separate step, after POS tagging and chunking. The last transducer then uses the function tags to extract subject/verb and object/verb relations (from French text).

6 Conclusion

POS are normally considered useful information in shallow and full parsing. Our learning curve experiments show that:

- The relative merit of words versus POS as input for the combined chunking and function-tagging task depends on the amount of training data available.
- The absolute performance of words depends on the treatment of rare words. The additional use of word form information (attenuation) improves performance.
- The addition of POS also improves performance. In this and the previous case, the effect becomes smaller with more training data.

Experiments with the maximal training set size show that:

- Addition of POS maximally yields an improvement of 1.7 points on this data.
- With realistic POS the improvement is much smaller.

Preliminary analysis shows that the improvement by realistic POS seems to be caused mainly by a superior use of word form information by the POS tagger. We therefore plan to experiment with a POS tagger and an attenuated words variant that use exactly the same word form information. In addition we also want to pursue using the combined chunker and grammatical function tagger described here as a first step towards grammatical *relation* assignment.

References

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

D. W. Aha, D. Kibler, and M. Albert. 1991. Instance-based learning algorithms. *Machine Learning*, 6:37–66.

S. Aït-Mokhtar and J.-P. Chanod. 1997. Subject and object dependency extraction using finite-state transducers. In *Proceedings of ACL'97 Workshop on Information Extraction and the Building of Lexical Semantic Resources for NLP Applications, Madrid*.

S. Argamon, I. Dagan, and Y. Krymolowski. 1998. A memory-based approach to learning shallow natural language patterns. In *Proc. of 36th annual meeting of the ACL*, pages 67–73, Montreal.

M. Banko and E. Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting and 10th Conference of the European Chapter of the Association for Computational Linguistics*, Toulouse, France.

E. Brill. 1993. *A Corpus-Based Approach to Language Learning*. Ph.D. thesis, University of Pennsylvania, Department of Computer and Information Science.

S. Buchholz, J. Veenstra, and W. Daelemans. 1999. Cascaded grammatical relation assignment. In Pascale Fung and Joe Zhou, editors, *Proceedings of EMNLP/VLC-99*, pages 239–246. ACL.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL'00*, pages 132–139.

K. W. Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proc. of Second Applied NLP (ACL)*.

M.J. Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*.

S. Cost and S. Salzberg. 1993. A weighted nearest neighbour algorithm for learning with symbolic features. *Machine Learning*, 10:57–78.

T. M. Cover and P. E. Hart. 1967. Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, 13:21–27.

W. Daelemans, J. Zavrel, P. Berck, and S. Gillis. 1996. MBT: A memory-based part of speech tagger generator. In E. Ejerhed and I. Dagan, editors, *Proc. of Fourth Workshop on Very Large Corpora*, pages 14–27. ACL SIGDAT.

W. Daelemans, A. Van den Bosch, and J. Zavrel. 1997. A feature-relevance heuristic for indexing and compressing large case bases. In M. Van Someren and G. Widmer, editors, *Poster Papers of the Ninth European Conference on Machine Learing*, pages 29–38, Prague, Czech Republic. University of Economics.

W. Daelemans, S. Buchholz, and J. Veenstra. 1999a. Memory-based shallow parsing. In *Proceedings of CoNLL*, Bergen, Norway.

W. Daelemans, A. Van den Bosch, and J. Zavrel. 1999b. Forgetting exceptions is harmful in language learning. *Machine Learning, Special issue on Natural Language Learning*, 34:11–41.

W. Daelemans, J. Zavrel, K. Van der Sloot, and A. Van den Bosch. 2001. TiMBL: Tilburg memory based learner, version 4.0, reference guide. ILK Technical Report 01-04, Tilburg University. available from http://ilk.kub.nl.

S.A. Dudani. 1976. The distance-weighted k-nearest neighbor rule. In *IEEE Transactions on Systems, Man, and Cybernetics*, volume SMC-6, pages 325–327.

J. Eisner. 1997. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*.

L. Ferro, M. Vilain, and A. Yeh. 1999. Learning transformation rules to find grammatical relations. In *Proceedings of the Third Computational Natural Language Learning workshop (CoNLL)*, pages 43–52.

X. Li and D. Roth. 2001. Exploring evidence for shallow parsing. In *Proceedings of the Fifth Computational Natural Language Learning workshop (CoNLL)*.

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

M. Muñoz, V. Punyakanok, D. Roth, and D. Zimak. 1999. A learning approach to shallow parsing. In Pascale Fung and Joe Zhou, editors, *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 168–178.

C. Pollard and I. Sag. 1987. *Information-Based Syntax and Semantics, Volume 1: Fundamentals*, volume 13 of *CSLI Lecture Notes*. Center for the Study of Language and Information, Stanford.

J.R. Quinlan. 1993. C4.5*: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

L.A. Ramshaw and M.P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the 3rd ACL/SIGDAT Workshop on Very Large Corpora, Cambridge, Massachusetts, USA*, pages 82–94.

A. Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proc. of the Conference on Empirical Methods in Natural Language Processing, May 17-18, 1996, University of Pennsylvania*.

A. Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing, EMNLP-2, Providence, Rhode Island*, pages 1–10.

C. Stanfill and D. Waltz. 1986. Toward memory-based reasoning. *Communications of the* ACM, 29(12):1213–1228, December.

E. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132, Lisbon, Portugal.

C.J. Van Rijsbergen. 1979. *Information Retrieval*. Buttersworth, London.

J. Veenstra and Antal van den Bosch. 2000. Single-classifier memory-based phrase chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 157–159, Lisbon, Portugal.

S. Weiss and C. Kulikowski. 1991. *Computer systems that learn*. San Mateo, CA: Morgan Kaufmann.

J. Zavrel. 1997. An empirical re-examination of weighted voting for k-NN. In *Proceedings of the 7th Belgian-Dutch Conference on Machine Learning*, pages xx–xx.