# Fast NP Chunking Using Memory-Based Learning Techniques

Jorn Veenstra & Sabine Buchholz
Computational Linguistics at Tilburg University
PO-box 90153, 5000 LE Tilburg
the Netherlands
veenstra@kub.nl

20th April 1998

**Abstract**

In this paper we discuss the application of Memory-Based Learning (MBL) to fast NP chunking. We first discuss the application of a fast decision tree variant of MBL (IGTree) on the dataset described in (Ramshaw and Marcus, 1995), which consists of roughly 50,000 test and 200,000 train items. In a second series of experiments we used an architecture of two cascaded IGTrees. In the second level of this cascaded classifier we added context predictions as extra features so that incorrect predictions from the first level can be corrected, yielding a 97.2% generalisation accuracy with training and testing times in the order of seconds to minutes.

**Submission Type:** regular paper

**Topic Areas:** robust parsing, NP chunking, memory-based learning

**Author of Record:** Jorn Veenstra

**Under consideration for other conferences (specify)?** no

# Fast NP Chunking Using Memory-Based Learning Techniques

**Abstract**

In this paper we discuss the application of Memory-Based Learning (MBL) to fast NP chunking. We first discuss the application of a fast decision tree variant of MBL (IGTree) on the dataset described in (Ramshaw and Marcus, 1995), which consists of roughly 50,000 test and 200,000 train items. In a second series of experiments we used an architecture of two cascaded IGTrees. In the second level of this cascaded classifier we added context predictions as extra features so that incorrect predictions from the first level can be corrected, yielding a 97.2% generalisation accuracy with training and testing times in the order of seconds to minutes.

## 1 Introduction

Language technology applications often require fast and accurate modules that can be developed and adapted rapidly (Daelemans et al., 1998a). In this paper we concentrate on a fast implementation of an NP chunking module. In chunking, a sentence is divided into nonoverlapping segments based on a relatively crude analysis; for an early paper on chunking, see (Abney, 1991). NP chunking is the segmentation of a sentence in so-called base-NPs. A base-NP is a non-recursive NP. The output of the NP chunker looks as in the following sentence:

**1** *[The emprisoned freedom fighter] will not see [any roses] [this year] .*

Information about the NP chunks in a sentence can be used for Information Retrieval or other text mining applications. Moreover, NP chunking is sometimes used as a preprocessing phase for parsing, c.f. (Collins, 1996; Ratnaparkhi, 1997). In parsing, NP chunking can be seen as one step in a cascade of classification tasks.

One way to see NP chunking as a classification task is to describe it as a tagging problem, as first defined by (Ramshaw and Marcus, 1995). They assign to each word in a sentence a tag that indicates whether this word is i) inside a base-NP (I), ii) outside a base-NP (O), or iii) inside a base-NP and the preceding word is in a different base-NP (B) (in the rest of the paper we will refer to these tags as IOB tags). Sentence 1 with the IOB tags looks like:

**2** *The/I emprisoned/I freedom/I fighter/I will/O not/O see/O any/I roses/I this/B year/I ./O*

These three IOB tags suffice to chunk every sentence unambiguously since there are no embedded chunks. We used these IOB tags for our experiments with MBL.

This paper is organised as follows: In Section 2 we give an introduction to Memory-Based Learning, focusing on the fast decision tree variant IGTree. In Section 3.1 we discuss a series of experiments in which we apply IGTree, implemented in TiMBL[1], to NP chunking. In Section 3.2 we discuss a cascaded NP chunker using IGTree. The first level of this chunker is discussed in Section 3.1, the second level uses IOB classifications of context words made in the first level as extra features. This cascaded NP chunker reaches a generalisation accuracy of 97.2% with a processing speed in the order of hundreds or thousands of words per second both in training and testing. In Section 3.3 we discuss the application of the usually more accurate but slower classifier IB1-IG to NP Chunking. In Section 4 we discuss the work

[1] TiMBL is an MBL software package, available for research purposes via the URL: http://ilk.kub.nl/

by (Ramshaw and Marcus, 1995), who applied transformation based learning to NP chunking.

## 2 Memory-Based Learning

Memory-Based Learning (MBL) is a form of *supervised, inductive learning from examples.* Examples or cases are represented as vectors of feature values with an associated class label. During training, a set of cases (the training set) is presented in an incremental fashion to the classifier and then added to memory (the case base). During testing, a set of previously unseen feature-value patterns (the test set) is presented to the system. For each test pattern its distance to all cases in memory is computed and the category of the nearest case(s) is used to predict the class of the test pattern. The approach is based on the assumption that reasoning is based on direct reuse of stored experiences rather than on the application of knowledge (such as rules or decision trees) abstracted from experience.

In AI, the concept of MBL has appeared in several disciplines (from computer vision to robotics) using names such as: example-based, case-based, similarity-based, exemplar-based, instance-based, lazy, analogical or nearest-neigbour (Stanfill and Waltz, 1986; Aha, 1991; Kolodner, 1993; Cost and Salzberg, 1993). Similar accounts about this type of analogical reasoning can be found in non-mainstream linguistics and psycholinguistics as well (Skousen, 1989; Derwing and Skousen, 1989; Chandler, 1992; Scha, 1992). In computational linguistics the idea of MBL has recently gained some popularity (Daelemans and Van den Bosch, 1996; Ng and Lee, 1996; Federici and Pirelli, 1996; Cardie, 1996).

**Similarity Metric** The performance of MBL (accuracy on the test set) crucially depends on the distance metric (or similarity metric) used. For non-numerical values, as mostly used in linguistic applications, the most straightforward distance metric is given in Equation 1:

$$\Delta(X,Y) = \sum_{i=1}^{n} \delta(x_i, y_i) \tag{1}$$

Where $X$ and $Y$ are the patterns to be compared; the distance between two patterns in the $i$th value is given by:

$$\delta(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

This algorithm is often referred to as IB1 (Aha, 1991). In the TiMBL (Daelemans et al., 1998b) implementation of MBL, IB1 is extended in the following way: When a case is associated with more than one class in the training set (i.e. the case is ambiguous), the distribution of cases over the different classes is kept and the most frequently occurring class is selected when the ambiguous case is used for extrapolation.

In this distance metric, all features of a case are considered equally important for the classification. In the case of NP chunking, as described below, the POS tag of the focus word obviously contributes more to the IOB classification of the word than the POS tag of the word two positions to the left of the focus word. Hence we weight each feature with its *Information Gain (IG)*; expressing the average amount of reduction of training set information entropy when knowing the value of the feature (Daelemans and van den Bosch, 1992; Quinlan, 1993; Hunt et al., 1966). Information gain is computed as in Equation 3:

$$w_f = \frac{H(C) - \sum_{v \in V_f} P(v) \times H(C|v)}{si(f)} \tag{3}$$

where the split info $si(f)$ is given by:

$$si(f) = -\sum_{v \in V_f} P(v) log_2 P(v) \tag{4}$$

and the entropy $H(C)$:

$$H(C) = -\sum_{c \in C} P(c) log_2 P(c) \quad (5)$$

**Speed** Classification in MBL using IG-weighted IB1 (IB1-IG) is computationally relatively costly: for each test item all feature values must be compared to the corresponding feature values of all training items. Without optimisation, it has an asymptotic retrieval complexity of $O(NF)$ (where $N$ is the number of items in memory, and $F$ the number of features). The same asymptotic complexity is of course found for memory storage in this approach. IGTree is a heuristic approximation of the IB1-IG algorithm.

**The IGTree Algorithm** IGTree (Daelemans et al., 1997) combines two algorithms: one for compressing case bases into trees, and one for retrieving classification information from these trees. During the construction of IGTree decision trees, cases are stored as paths of connected nodes. All nodes contain a test (based on one of the features) and a class label (representing the default class at that node). Nodes are connected via arcs denoting the outcomes for the test (feature values). A feature relevance ordering technique (in this case: Information Gain (IG), see Equation 3) is used to determine the order in which features are used as tests in the tree. This order is fixed in advance, so the maximal depth of the tree is always equal to the number of features and at the same level of the tree all nodes have the same test (this makes IGTrees an instance of *oblivious decision trees*; c.f. Langley and Sage (1994)).

Besides restricting search to those memory cases that match only on this feature, the case memory can be optimised by further restricting search to the second most important feature, followed by the third important feature, etcetera. A considerable compression is obtained as similar cases partially share paths.

After converting the case base to a tree in which all cases are fully represented as paths, storing all feature values, we compress the tree even more by resticting the paths to those input feature values that disambiguate the classification from all other cases in the training material. The idea is that it is not necessary to fully store a case as a path when only a few feature values of the case make its classification unique. For this chunking application this means that only context feature values that actually contribute to disambiguation are used in the construction of the tree.

Leaf nodes contain the unique class label corresponding to a path in the tree. Non-terminal nodes contain information about the *most probable* or *default* classification given the path thus far, according to the bookkeeping information on class occurrences maintained by the tree construction algorithm. Finding the classification of a new case involves traversing the tree (i.e. matching all the feature values of the test case with arcs in the order of the overall feature IG) and either retrieving a classification when a leaf is reached or using the default classification on the last matching non-terminal node if a feature value match fails.

A final compression is obtained by pruning the derived tree. All leaf node daughters of a mother node that have the same class as that node are removed from the tree, as their class information does not contradict the default class information already present at the mother node. Again, this compression does not affect the generalisation performance of IGTree.

For a more detailed discussion on IGTrees, see (Daelemans et al., 1997).

## 3 Experiments

We carried out two series of experiments. In the first we used IGTree for the NP chunking task. In the second series we used what we will call cascaded processing to improve the

| feature(s) | accuracy |
|---|---|
| most frequent | 54.5% |
| focus word | 93.1% |
| focus tag | 94.3% |

Table 1: Baseline results on the R&M NP chunking dataset

performance of IGTree.

**Data** For our experiments on NP chunking we made use of the NP chunking dataset as described in Ramshaw and Marcus (1995). This dataset is taken from sections 15–18 from the parsed data in the Penn Treebank corpus of Wall Street Journal text (Marcus et al., 1993) for training and section 20 for testing, amounting to 47,377 test cases and 203,751 train cases. The tree annotation was used to algorithmically derive the NP chunks; the words in the sentence were tagged with an I for inside an NP chunk, O for outside and B for inside an NP chunk following another NP chunk (see Section 1).

**Baseline** Some preliminary experiments were conducted to compute the baseline score for the IOB tag prediction on the dataset used. Three baseline score were determined: the accuracy reached by i) taking the most frequent class (i.e. I), and by only taking ii) the focus word, or iii) the focus tag into account, see Table 1. In the last two cases the most frequent class over the training set is taken in case of an unknown word or tag, in case of ambiguity the most frequent class for that case is taken, in case of a tie the most frequent class (from among the tied classes) over the whole training set is taken. It is shown that the baseline is as high as 94.3% for the focus tag only.

### 3.1 Experiments with IGTree1

In a first series of experiments we tested the performance of IGTree (see section 2) on the R&M-dataset.

| Method | Tagger | accuracy |
|---|---|---|
| IGTree1 | Brill | 96.6% |
| IGTree1 | MBT | 96.8% |

Table 2: First results on the R&M NP chunking dataset, with Brill tags and MBT tags, using IGTree

**Data** For this series of experiments we took the word form and the POS tag of the words one and two positions to the left of the focus word, the word at focus position and the word one position to the right as features, hence the input vectors were of the form: (word-2, word-1, word0, word+1, POS-2, POS-1, POS0, POS+1, target IOBtag).

R&M used a transformation based tagger (Brill, 1993) to assign POS tags to their train and test data. We first used the same Brill tags for one experiment with IGTree. Since one of our aims is to construct an NP chunker which can take untagged sentences as input and "NP chunked" sentences as output we also used the Memory-Based Tagger (MBT) (Daelemans et al., 1996) to tag the datasets.

**Results** The results in Table 2 show the accuracy in predicting the IOB tags, using the one-level IGTree1 classifier, with tags assigned by a transformation based (Brill) tagger and by a Memory-Based tagger (MBT). The usage of the MBT tagger yields a slighty higher accuracy, but this difference is not significant.

The precision and recall, shown in Table 4, is measured at the level of NP chunk prediction, i.e. if a predicted chunk exactly matches a target chunk it adds to the precision and if a target chunk exactly matches a predicted chunk it adds to the recall. The 97.0% recall for chunk retrieval is very high, but the precision of only 83.1% is low. In the next section we will discuss an attempt to improve the precision with a cascaded IGTree classifier.

### 3.2 Experiments with IGTree2

In the research described in this paper we want to develop an NP chunker which is, besides fast and flexible, also accurate. Training and testing of the NP chunker described in Section 3.1 is fast, but the generalisation accuracy, especially the precision of chunk retrieval, can be improved. Therefore, we used the fast classifier IGTree in a second series of experiments, in which we defined NP chunking as a cascaded classification task: in the first level we used the same input data as in section 3.1, tagged with the MBT. In the second level we added the classifications of the first level of the four feature words as extra features. On the R&M dataset the time usage of IGTree is in the order minutes for training and testing together.

**Data** The IOB predictions of the focus word and the surrounding words are included as extra features in the dataset for the second pass. To compute these features for the train set we ran a tenfold cross-validation (10cv) using IGTree on the train set. We then used the predictions for the 10% left out cases to add the IOB tags as extra features. Finally the train set was constructed by putting together the ten partitions. The advantage of using the predicted classes instead of the original target classes is that this way we can add information about the classifier to the training set; typical errors for the classifier can be taken into account in the second level of processing, see (Van den Bosch, 1997) for a discussion of this issue in the context of MBL.

The first level classifier is used to add the IOB features to the test set. For reasons of speed we did not use the word forms as features for the second level classifier, we used only the POS tags and the IOB predictions, hence the input vectors were of the form: ([POS-2, POS-1], POS0, [POS+1], IOB-2, IOB-1, IOB0, IOB+1, target IOBtag).

| feature(s) | accuracy |
|---|---|
| focus POS tag & IOB4 | 97.2% |
| POS tag4 & IOB4 | 97.1% |

Table 3: Test results after the second pass, using the context predictions as extra features. IOB4: context prediction of word-2, word-1, focus-word and word+1; tag4: the same for POS tags

| method | precision | recall |
|---|---|---|
| IGTree1 | 83.1% | 97.0% |
| IGTree2 | 89.0% | 94.3% |

Table 4: Precision and recall of NP chunk prediction using IGTree with one level (IGTree1, see Section 3.1) and with two levels (IGTree2, see Section 3.2)

**Results** In Table 3 the prediction accuracy for IOB tags is shown, using the cascaded IGTree2. The generalisation accuracy is measured after the second level. We can see that we get a slighty higher accuracy (though not significant) if we use only the focus POS tag as feature, in combination with the four IOB tags.

In table 4 we show the precision and recall of the experiment using the one (IGTree1) and the two (IGTree2) level version of IGTree. It shows that IGTree2 has a higher precision and a lower recall than IGTree1, indicating that the second level of cascaded processing corrects a lot of mispredicted NP chunks in the first level, but has a tendency to "overcorrect".

**Time usage** In Table 5 we give an overview of the time usage of cascaded NP chunking on a Pentium pro 200 MHz Solaris machine. The features in the second level are only the POS tags and the IOB predictions of word-2, word-1, word0 and word+1.

In the current implementation of the IGTree2 NP chunker it is possible to process about $10^3$ words to process the train set and

| step | words p.s. |
|---|---|
| training | |
| 10cv on trainset | $2 * 10^3$ |
| constr. trainset | $2 * 10^3$ |
| total train | $10^3$ |
| testing only | |
| two levels | $3 * 10^3$ |

Table 5: Time usage for cascaded NP chunking using IGTree2

$3 * 10^3$ words per second to classify test cases.

### 3.3 IB1-IG

IGTree is a decision tree variant of IB1-IG (see Section 2). IB1-IG tends to yield a higher generalisation accuracy than IGTree. Therefore, we also applied IB1-IG on the R&M dataset.

The training time for IB1-IG is a bit shorter than for IGTree, but the testing time is significantly longer since each test case is compared to all train cases. This makes IB1-IG less suited for building a fast NP chunker.

We tested IB1-IG for the one-level classification, without the IOB prediction as extra features (IB1-IG1), and with the IOB predictions as extra features (IB1-IG2). The results are shown in Table 6.

## 4 Transformation based learning

Ramshaw and Marcus (1995) used transformation-based learning (Brill, 1993) as a classifier on their R&M dataset. Transformation based learning (for a more detailed description see (Brill, 1993; Ramshaw and Marcus, 1995)) can be roughly described as follows:

Start with a training corpus annotated with the correct classes and a large set of rule templates. Rules are instantiations of these rule templates. To train the transformation based system:

1. start with a baseline classifier to predict the classes in the train set,
2. search the space of possible rules to find the rule that maximally reduces the error (the difference between the correct and predicted classes),
3. apply this rule and start again at 2, or quit if some threshold error or number of steps is reached.

The rules thus found can now be applied (in the order they were found) to compute the generalisation error on the test set. The rule templates each define a small feature set that might be relevant for classification, e.g. [the POS tag of word0 and the POS tag of word-1]. Brill's tagger made use of 26 such templates, the NP chunker used 100 templates consisting of combinations of words, POS tags and IOB tags of the focus and its surrounding words. This large number of templates makes the transformation based NP chunker difficult to train. As (Ramshaw and Marcus, 1995, p.89) put it: "The large increase in the number of rule templates [...] pushed the training process against the available limits in terms of both time and space". It took days rather than hours to train the transformation based system. However, R&M took no particular effort to make their transformation based system fast since speed was not an issue (Ramshaw p.c.).

The generalisation accuracy for transformation based learning (97.4%) as shown in Table 6 is not significantly different from IGTree2 (97.2%), according to a $X^2$-test with $\alpha = 0.05$, $X^2_o = .1$ and $X^2_{crit} = 3.84$.

## 5 Conclusion

The results in this paper show that it is possible to construct a fast and flexible cascaded IGTree2 NP chunker with a not significant loss of accuracy compared to transformation based learning and IB1-IG.

In case of training and testing the IGTree2 NP chunker on completely new data comparable to the R&M data it will take in the order of 3 minutes to train the IGTree2 classifier. After this training phase the IGTree2

| Methods | Tagger | recall | precision | accuracy |
|---|---|---|---|---|
| Transformation | Brill | 92.3% | 91.8% | 97.4% |
| IGTree1 | MBT | 97.0% | 83.1% | 96.8% |
| IB1-IG1 | MBT | 92.7% | 89.4% | 97.3% |
| IGTree2 | MBT | 94.3% | 89.0% | 97.2% |
| IB1IG2 | MBT | 94.1% | 87.4% | 97.2% |

Table 6: Overview of the score on the R&M dataset of several classifiers

chunker will process about $3 * 10^3$ words per second.

## Acknowledgements

This research was done in the context of the "Induction of Linguistic Knowledge" (ILK) research programme, supported by the Netherlands Organization for Scientific Research (NWO).

## References

Steven Abney. 1991. Parsing by chunks. In *Principle-Based Parsing*. Kluwer Academic Publishers, Dordrecht.

D. W. Aha. 1991. Incremental constructive induction: an instance-based approach. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 117–121, Evanston, ILL. Morgan Kaufmann.

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

Claire Cardie. 1996. Automatic feature set selection for case-based learning of linguistic knowledge. In *Proc. of Conference on Empirical Methods in NLP*. University of Pennsylvania.

S. Chandler. 1992. Are rules and modules really necessary for explaining language? *Journal of Psycholinguistic research*, 22(6):593–606.

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.

Walter Daelemans and Antal van den Bosch. 1992. Generalisation performance of backpropagation learning on a syllabification task. In M. F. J. Drossaers and A. Nijholt, editors, *Proc. of TWLT3: Connectionism and Natural Language Processing*, pages 27–37, Enschede. Twente University.

W. Daelemans and A. Van den Bosch. 1996. Language-independent data-oriented grapheme-to-phoneme conversion. In J. P. H. Van Santen, R. W. Sproat, J. P. Olive, and J. Hirschberg, editors, *Progress in Speech Processing*, pages 77–89. Springer-Verlag, Berlin.

Walter Daelemans, Jakub Zavrel, Peter Berck, and Steven 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 A. Weijters. 1997. IGTree: using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11:407–423.

W. Daelemans, A. Van den Bosch, J. Zavrel, J. Veenstra, S. Buchholz, and G. J. Busser. 1998a. Rapid development of NLP modules with Memory-Based Learning. In *Proceedings of ELSNET in Wonderland, March, 1998*, pages 105–113. ELSNET.

W. Daelemans, J. Zavrel, K. Van der Sloot, and A. Van den Bosch. 1998b. TiMBL: Tilburg Memory Based Learner, version 1.0, reference manual. Technical Report ILK-9803, ILK, Tilburg University.

B. L. Derwing and R. Skousen. 1989. Real time morphology: Symbolic rules or analogical networks. *Berkeley Linguistic Society*, 15:48–62.

S. Federici and V. Pirelli. 1996. Analogy, computation and linguistic theory. In *New Methods in Language Processing*. UCL Press, London.

E. B. Hunt, J. Marin, and P. J. Stone. 1966. *Experiments in induction*. Academic Press, New York, NY.

J. Kolodner. 1993. *Case-based reasoning*. Morgan Kaufmann, San Mateo, CA.

P. Langley and S. Sage. 1994. Oblivious decision trees and abstract cases. In D. W. Aha, editor, *Case-Based Reasoning: Papers from the 1994 Workshop (Technical Report WS-94-01)*, Menlo Park, CA. AAAI Press.

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.

Hwee Tou Ng and Hian Beng Lee. 1996. Intergrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *Proc. of 34th meeting of the Assiociation for Computational Linguistics*.

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 *Proc. of third workshop on very large corpora*, pages 82–94, June.

A. Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. Technical Report cmp-lg/9706014, Computation and Language, http://xxx.lanl.gov/list/cmp-lg/, June.

R. Scha. 1992. Virtual Grammars and Creative Algorithms. *Gramma/TTT Tijdschrift voor Taalkunde*, 1:57–77.

R. Skousen. 1989. *Analogical modeling of language*. Kluwer Academic Publishers, Dordrecht.

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

A. Van den Bosch. 1997. *Learning to pronounce written words: A study in inductive language learning*. Ph.D. thesis, Universiteit Maastricht. forthcoming.