Representing Text Chunks

Erik F. Tjong Kim Sang
Center for Dutch Language and Speech
University of Antwerp
Universiteitsplein 1
B-2610 Wilrijk, Belgium
erikt@uia.ac.be

Jorn Veenstra
Computational Linguistics
Tilburg University
P.O. Box 90153
5000 LE Tilburg, The Netherlands
veenstra@kub.nl

Abstract

Dividing sentences in chunks of words is a useful preprocessing step for parsing, information extraction and information retrieval. (Ramshaw and Marcus, 1995) have introduced a "convenient" data representation for chunking by converting it to a tagging task. In this paper we will examine seven different data representations for the problem of recognizing noun phrase chunks. We will show that the the data representation choice has a minor influence on chunking performance. However, equipped with the most suitable data representation, our memory-based learning chunker was able to improve the best published chunking results for a standard data set.

1 Introduction

The text corpus tasks parsing, information extraction and information retrieval can benefit from dividing sentences in chunks of words. (Ramshaw and Marcus, 1995) describe an error-driven transformation-based learning (TBL) method for finding NP chunks in texts. NP chunks (or baseNPs) are non-overlapping, non-recursive noun phrases. In their experiments they have modeled chunk recognition as a tagging task: words that are inside a baseNP were marked `I`, words outside a baseNP received an `O` tag and a special tag `B` was used for the first word inside a baseNP immediately following another baseNP. A text example:

original:
In $[_N$ early trading $_N]$ in $[_N$ Hong Kong $_N]$ $[_N$ Monday $_N]$, $[_N$ gold $_N]$ was quoted at $[_N$ $ 366.50 $_N]$ $[_N$ an ounce $_N]$.
tagged:
In/O early/I trading/I in/O Hong/I Kong/I Monday/B ,/O gold/I was/O quoted/O at/O $/I 366.50/I an/B ounce/I ./O

Other representations for NP chunking can be used as well. An example is the representation used in (Ratnaparkhi, 1998) where all the chunk-initial words receive the same start tag (analogous to the `B` tag) while the remainder of the words in the chunk are paired with a different tag. This removes tagging ambiguities. In the Ratnaparkhi representation equal noun phrases receive the same tag sequence regardless of the context in which they appear.

The data representation choice might influence the performance of chunking systems. In this paper we discuss how large this influence is. Therefore we will compare seven different data representation formats for the baseNP recognition task. We are particularly interested in finding out whether with one of the representation formats the best reported results for this task can be improved. The second section of this paper presents the general setup of the experiments. The results can be found in the third section. In the fourth section we will describe some related work.

2 Methods and experiments

In this section we present and explain the data representation formats and the machine learning algorithm that we have used. In the final part we describe the feature representation used in our experiments.

2.1 Data representation

We have compared four complete and three partial data representation formats for the baseNP recognition task presented in (Ramshaw and Marcus, 1995). The four complete formats all use an `I` tag for words that are inside a baseNP and an `O` tag for words that are outside a baseNP. They differ

IOB1	O	I	I	O	I	I	B	O	I	O	O	O	I	I	B	I	O
IOB2	O	B	I	O	B	I	B	O	B	O	O	O	B	I	B	I	O
IOE1	O	I	I	O	I	E	I	O	I	O	O	O	I	E	I	I	O
IOE2	O	I	E	O	I	E	E	O	E	O	O	O	I	E	I	E	O
IO	O	I	I	O	I	I	I	O	I	O	O	O	I	I	I	I	O
[	.	[	.	.	[	.	[	.	[	.	.	.	[	.	[	.	.
]	.	.	]	.	.	]	]	.	]	.	.	.	.	]	.	]	.

Table 1: The chunk tag sequences for the example sentence *In early trading in Hong Kong Monday , gold was quoted at $ 366.50 an ounce .* for seven different tagging formats. The `I` tag has been used for words inside a baseNP, `O` for words outside a baseNP, `B` and `[` for baseNP-initial words and `E` and `]` for baseNP-final words.

in their treatment of chunk-initial and chunk-final words:

- IOB1 The first word inside a baseNP immediately following another baseNP receives a B tag (Ramshaw and Marcus, 1995).
- IOB2 All baseNP-initial words receive a B tag (Ratnaparkhi, 1998).
- IOE1 The final word inside a baseNP immediately preceding another baseNP receives an E tag.
- IOE2 All baseNP-final words receive an E tag.

We wanted to compare these data representation formats with a standard bracket representation. We have chosen to divide bracketing experiments in two parts: one for recognizing opening brackets and one for recognizing closing brackets. Additionally we have worked with another partial representation which seemed promising: a tagging representation which disregards boundaries between adjacent chunks. These boundaries can be recovered by combining this format with one of the bracketing formats. Our three partial representations are:

- [All baseNP-initial words receive an [tag, other words receive a . tag.
-] All baseNP-final words receive a] tag, other words receive a . tag.
- IO Words inside a baseNP receive an `I` tag, others receive an `O` tag.

These partial representations can be combined in three pairs which encode the complete baseNP structure of the data:

- [+] A word sequence is regarded as a baseNP if the first word has received an [tag, the final word has received a] tag and these are the only brackets that have been assigned to words in the sequence.
- [+ IO In the IO format, tags of words that have received an `I` tag and an [tag are changed into `B` tags. The result is interpreted as the IOB2 format.
- IO +] In the IO format, tags of words that have received an `I` tag and a] tag are changed into `E` tags. The result is interpreted as the IOE2 format.

Examples of the four complete formats and the three partial formats can be found in table 1.

2.2 Memory-Based Learning

We have build a baseNP recognizer by training a machine learning algorithm with correct tagged data and testing it with unseen data. The machine learning algorithm we used was a Memory-Based Learning algorithm (MBL). During training it stores a symbolic feature representation of a word in the training data together with its classification (chunk tag). In the testing phase the algorithm compares a feature representation of a test word with every training data item and chooses the classification of the training item which is closest to the test item.

In the version of the algorithm that we have used, IB1-IG, the distances between feature representations are computed as the weighted sum of distances between individual features (Daelemans et al., 1998). Equal features are defined to have distance 0, while the distance between other pairs is some feature-dependent value. This value is equal to the information gain of the feature, an information theoretic measure which contains the

	word/POS context	$F_{\beta=1}$
IOB1	L=2/R=1	89.17
IOB2	L=2/R=1	88.76
IOE1	L=1/R=2	88.67
IOE2	L=2/R=2	89.01
[+]	L=2/R=1 + L=0/R=2	89.32
[+ IO	L=2/R=0 + L=1/R=1	89.43
IO +]	L=1/R=1 + L=0/R=2	89.42

Table 2: Results first experiment series: the best $F_{\beta=1}$ scores for different left (L) and right (R) word/POS tag pair context sizes for the seven representation formats using 5-fold cross-validation on section 15 of the WSJ corpus.

normalized entropy decrease of the classification set caused by the presence of the feature. Details of the algorithm can be found in (Daelemans et al., 1998)[1].

2.3 Representing words with features

An important decision in an MBL experiment is the choice of the features that will be used for representing the data. IB1-IG is thought to be less sensitive to redundant features because of the data-dependent feature weighting that is included in the algorithm. We have found that the presence of redundant features has a negative influence on the performance of the baseNP recognizer.

In (Ramshaw and Marcus, 1995) a set of transformational rules is used for modifying the classification of words. The rules use context information of the words, the part-of-speech tags that have been assigned to them and the chunk tags that are associated with them. We will use the same information as in our feature representation for words.

In TBL, rules with different context information are used successively for solving different problems. We will use the same context information for all data. The optimal context size will be determined by comparing the results of different context sizes on the training data. Here we will perform four steps. We will start with testing different context sizes of words with their part-of-speech tag. After this, we will use the classification results of the best context size for determining the optimal context size for the classification tags. As a third step, we will evaluate combinations of classification results and find the best combination. Finally we will examine the influence of an MBL algorithm parameter: the number of examined nearest neighbors.

[1] IB1-IG is a part of the TiMBL software package which is available from http://ilk.kub.nl

3 Results

We have used the baseNP data presented in (Ramshaw and Marcus, 1995)[2]. This data was divided in two parts. The first part was training data and consisted of 211727 words taken from sections 15, 16, 17 and 18 from the Wall Street Journal corpus (WSJ). The second part was test data and consisted of 47377 words taken from section 20 of the same corpus. The words were part-of-speech (POS) tagged with the Brill tagger and each word was classified as being inside or outside a baseNP with the IOB1 representation scheme. The chunking classification was made by (Ramshaw and Marcus, 1995) based on the parsing information in the WSJ corpus.

The performance of the baseNP recognizer can be measured in different ways: by computing the percentage of correct classification tags (accuracy), the percentage of recognized baseNPs that are correct (precision) and the percentage of baseNPs in the corpus that are found (recall). We will follow (Argamon et al., 1998) and use a combination of the precision and recall rates: $F_{\beta=1}$ = (2*precision*recall)/(precision+recall).

In our first experiment series we have tried to discover the best word/part-of-speech tag context for each representation format. For computational reasons we have limited ourselves to working with section 15 of the WSJ corpus. This section contains 50442 words. We have run 5-fold cross-validation experiments with all combinations of left and right contexts of word/POS tag pairs in the size range 0 to 4. A summary of the results can be found in table 2.

The baseNP recognizer performed best with relatively small word/POS tag pair contexts. Different representation formats required different context sizes for optimal performance. All formats

[2] The data described in (Ramshaw and Marcus, 1995) is available from ftp://ftp.cis.upenn.edu/pub/chunker/

	word/POS context	chunk tag context	$F_{\beta=1}$
IOB1	L=2/R=1	1/2	90.12
IOB2	L=2/R=1	1/0	89.30
IOE1	L=1/R=2	1/2	89.55
IOE2	L=1/R=2	0/1	89.73
[+]	L=2/R=1 + L=0/R=2	0/0 + 0/0	89.32
[+ IO	L=2/R=0 + L=1/R=1	0/0 + 1/1	89.78
IO +]	L=1/R=1 + L=0/R=2	1/1 + 0/0	89.86

Table 3: Results second experiment series: the best $F_{\beta=1}$ scores for different left (L) and right (R) chunk tag context sizes for the seven representation formats using 5-fold cross-validation on section 15 of the WSJ corpus.

	word/POS	chunk tag	combinations	$F_{\beta=1}$
IOB1	2/1	1/1	0/0 1/1 2/2 3/3	90.53
IOB2	2/1	1/0	2/1	89.30
IOE1	1/2	1/2	0/0 1/1 2/2 3/3	90.03
IOE2	1/2	0/1	1/2	89.73
[+]	2/1 + 0/2	0/0 + 0/0	- + -	89.32
[+ IO	2/0 + 1/1	0/0 + 1/1	- + 0/1 1/2 2/3 3/4	89.91
IO +]	1/1 + 0/2	1/1 + 0/0	0/1 1/2 2/3 3/4 + -	90.03

Table 4: Results third experiment series: the best $F_{\beta=1}$ scores for different combinations of chunk tag context sizes for the seven representation formats using 5-fold cross-validation on section 15 of the WSJ corpus.

with explicit open bracket information preferred larger left context and most formats with explicit closing bracket information preferred larger right context size. The three combinations of partial representations systematically outperformed the four complete representations. This is probably caused by the fact that they are able to use two different context sizes for solving two different parts of the recognition problem.

In a second series of experiments we used a "cascaded" classifier. This classifier has two stages (cascades). The first cascade is similar to the classifier described in the first experiment. For the second cascade we added the classifications of the first cascade as extra features. The extra features consisted of the left and the right context of the classification tags. The focus chunk tag (the classification of the current word) accounts for the correct classification in about 95% of the cases. The MBL algorithm assigns a large weight to this input feature and this makes it harder for the other features to contribute to a good result. To avoid this we have refrained from using this tag. Our goal was to find out the optimal number of extra classification tags in the input. We performed 5-fold cross-validation experiments with all combinations of left and right classification tag contexts in the range 0 tags to 3 tags. A summary of the results can be found in table 3[3]. We achieved higher $F_{\beta=1}$ for all representations except for the bracket pair representation.

The third experiment series was similar to the second but instead of adding output of one experiment we added classification results of three, four or five experiments of the first series. By doing this we supplied the learning algorithm with information about different context sizes. This information is available to TBL in the rules which use different contexts. We have limited ourselves to examining all successive combinations of three, four and five experiments of the lists (L=0/R=0, 1/1, 2/2, 3/3, 4/4), (0/1, 1/2, 2/3, 3/4) and (1/0, 2/1, 3/2, 4/3). A summary of the results can be found in table 4. The results for four representation formats improved.

In the fourth experiment series we have experimented with a different value for the number of nearest neighbors examined by the IB1-IG algorithm (parameter k). This algorithm standardly uses the single training item closest to the test

[3] In a number of cases a different base configuration in one experiment series outperformed the best base configuration found in the previous series. In the second series L/R=1/2 outperformed 2/2 for IOE2 when chunk tags were added and in the third series chunk tag context 1/1 outperformed 1/2 for IOB1 when different combinations were tested.

	word/POS	chunk tag	combinations	$F_{\beta=1}$
IOB1	3/3(k=3)	1/1	0/0(1) 1/1(1) 2/2(3) 3/3(3)	90.89 ± 0.63
IOB2	3/3(k=3)	1/0	3/3(3)	89.72 ± 0.79
IOE1	2/3(k=3)	1/2	0/0(1) 1/1(1) 2/2(3) 3/3(3)	90.12 ± 0.27
IOE2	2/3(k=3)	0/1	2/3(3)	90.02 ± 0.48
[+]	4/3(3) + 4/4(3)	0/0 + 0/0	- + -	90.08 ± 0.57
[+ IO	4/3(3) + 3/3(3)	0/0 + 1/1	- + 0/1(1) 1/2(3) 2/3(3) 3/4(3)	90.35 ± 0.75
IO +]	3/3(3) + 2/3(3)	1/1 + 0/0	0/1(1) 1/2(3) 2/3(3) 3/4(3) + -	90.23 ± 0.73

Table 5: Results fourth experiment series: the best $F_{\beta=1}$ scores for different combinations of left and right classification tag context sizes for the seven representation formats using 5-fold cross-validation on section 15 of the WSJ corpus obtained with IB1-IG parameter k=3. IOB1 is the best representation format but the differences with the results of the other formats are not significant.

item. However (Daelemans et al., 1999) report that for baseNP recognition better results can be obtained by making the algorithm consider the classification values of the three closest training items. We have tested this by repeating the first experiment series and part of the third experiment series for k=3. In this revised version we have repeated the best experiment of the third series with the results for k=1 replaced by the k=3 results whenever the latter outperformed the first in the revised first experiment series. The results can be found in table 5. All formats benefited from this step. In this final experiment series the best results were obtained with IOB1 but the differences with the results of the other formats are not significant.

We have used the optimal experiment configurations that we had obtained from the fourth experiment series for processing the complete (Ramshaw and Marcus, 1995) data set. The results can be found in table 6. They are better than the results for section 15 because more training data was used in these experiments. Again the best result was obtained with IOB1 ($F_{\beta=1}$=92.37) which is an improvement of the best reported $F_{\beta=1}$ rate for this data set ((Ramshaw and Marcus, 1995): 92.03).

We would like to apply our learning approach to the large data set mentioned in (Ramshaw and Marcus, 1995): Wall Street Journal corpus sections 2-21 as training material and section 0 as test material. With our present hardware applying our optimal experiment configuration to this data would require several months of computer time. Therefore we have only used the best stage 1 approach with IOB1 tags: a left and right context of three words and three POS tags combined with k=3. This time the chunker achieved a $F_{\beta=1}$ score of 93.81 which is half a point better than the results obtained by (Ramshaw and Marcus, 1995): 93.3 (other chunker rates for this data: accuracy: 98.04%; precision: 93.71%; recall: 93.90%).

4 Related work

The concept of chunking was introduced by Abney in (Abney, 1991). He suggested to develop a chunking parser which uses a two-part syntactic analysis: creating word chunks (partial trees) and attaching the chunks to create complete syntactic trees. Abney obtained support for such a chunking stage from psycholinguistic literature.

Ramshaw and Marcus used transformation-based learning (TBL) for developing two chunkers (Ramshaw and Marcus, 1995). One was trained to recognize baseNPs and the other was trained to recognize both NP chunks and VP chunks. Ramshaw and Marcus approached the chunking task as a tagging problem. Their baseNP training and test data from the Wall Street Journal corpus are still being used as benchmark data for current chunking experiments. (Ramshaw and Marcus, 1995) shows that baseNP recognition ($F_{\beta=1}$=92.0) is easier than finding both NP and VP chunks ($F_{\beta=1}$=88.1) and that increasing the size of the training data increases the performance on the test set.

The work by Ramshaw and Marcus has inspired three other groups to build chunking algorithms. (Argamon et al., 1998) introduce Memory-Based Sequence Learning and use it for different chunking experiments. Their algorithm stores sequences of POS tags with chunk brackets and uses this information for recognizing chunks in unseen data. It performed slightly worse on baseNP recognition than the (Ramshaw and Marcus, 1995) experiments ($F_{\beta=1}$=91.6). (Cardie and Pierce, 1998) uses a related method but they only store POS tag sequences forming complete baseNPs. These sequences were applied to unseen tagged data after which post-processing repair rules were used for fixing some frequent errors. This approach performs worse than other reported approaches ($F_{\beta=1}$=90.9).

	accuracy	precision	recall	$F_{\beta=1}$
IOB1	97.58%	92.50%	92.25%	92.37
IOB2	96.50%	91.24%	92.32%	91.78
IOE1	97.58%	92.41%	92.04%	92.23
IOE2	96.77%	91.93%	92.46%	92.20
[+]	-	93.66%	90.81%	92.22
[+ IO	-	91.47%	92.61%	92.04
IO +]	-	91.25%	92.54%	91.89
(Ramshaw and Marcus, 1995)	97.37%	91.80%	92.27%	92.03
(Veenstra, 1998)	97.2%	89.0%	94.3%	91.6
(Argamon et al., 1998)	-	91.6 %	91.6%	91.6
(Cardie and Pierce, 1998)	-	90.7%	91.1%	90.9

Table 6: The $F_{\beta=1}$ scores for the (Ramshaw and Marcus, 1995) test set after training with their training data set. The data was processed with the optimal input feature combinations found in the fourth experiment series. The accuracy rate contains the fraction of chunk tags that was correct. The other three rates regard baseNP recognition. The bottom part of the table shows some other reported results with this data set. With all but two formats IB1-IG achieves better $F_{\beta=1}$ rates than the best published result in (Ramshaw and Marcus, 1995).

(Veenstra, 1998) uses cascaded decision tree learning (IGTree) for baseNP recognition. This algorithm stores context information of words, POS tags and chunking tags in a decision tree and classifies new items by comparing them to the training items. The algorithm is very fast and it reaches the same performance as (Argamon et al., 1998) ($F_{\beta=1}$=91.6). (Daelemans et al., 1999) uses cascaded MBL (IB1-IG) in a similar way for several tasks among which baseNP recognition. They do not report $F_{\beta=1}$ rates but their tag accuracy rates are a lot better than accuracy rates reported by others. However, they use the (Ramshaw and Marcus, 1995) data set in a different training-test division (10-fold cross validation) which makes it difficult to compare their results with others.

5 Concluding remarks

We have compared seven different data formats for the recognition of baseNPs with memory-based learning (IB1-IG). The IOB1 format, introduced in (Ramshaw and Marcus, 1995), consistently came out as the best format. However, the differences with other formats were not significant. Some representation formats achieved better precision rates, others better recall rates. This information is useful for tasks that require chunking structures because some tasks might be more interested in high precision rates while others might be more interested in high recall rates.

The IB1-IG algorithm has been able to improve the best reported $F_{\beta=1}$ rates for a standard data set (92.37 versus (Ramshaw and Marcus, 1995)'s 92.03). This result was aided by using non-standard parameter values (k=3) and the algorithm was sensitive for redundant input features. This means that finding an optimal performance or this task requires searching a large parameter/feature configuration space. An interesting topic for future research would be to embed IB1-IG in a standard search algorithm, like hill-climbing, and explore this parameter space. Some more room for improved performance lies in computing the POS tags in the data with a better tagger than presently used.

References

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

Shlomo Argamon, Ido Dagan, and Yuval Krymolowski. 1998. A memory-based approach to learning shallow natural language patterns. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING-ACL '98)*.

Claire Cardie and David Pierce. 1998. Error-driven pruning of treebank grammars for base noun phrase identification. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING-ACL '98)*.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 1998. *TiMBL: Tilburg Memory Based Learner - version 1.0 - Reference Guide*. ILK, Tilburg University, The Netherlands. http://ilk.kub.nl/~ilk/papers/ilk9803.ps.gz.

Walter Daelemans, Antal van den Bosch, and Jakub Zavrel. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning*, 11.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora.*

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

Jorn Veenstra. 1998. Fast np chunking using memory-based learning techniques. In *BENELEARN-98: Proceedings of the Eigth Belgian-Dutch Conference on Machine Learning.* ATO-DLO, Wageningen, report 352.