In: *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132, Lisbon, Portugal, 2000.

Introduction to the CoNLL-2000 Shared Task: Chunking

Erik F. Tjong Kim Sang
CNTS – Language Technology Group
University of Antwerp
erikt@uia.ua.ac.be

Sabine Buchholz
ILK, Computational Linguistics
Tilburg University
s.buchholz@kub.nl

Abstract

We describe the CoNLL-2000 shared task: dividing text into syntactically related non-overlapping groups of words, so-called text chunking. We give background information on the data sets, present a general overview of the systems that have taken part in the shared task and briefly discuss their performance.

1 Introduction

Text chunking is a useful preprocessing step for parsing. There has been a large interest in recognizing non-overlapping noun phrases (Ramshaw and Marcus (1995) and follow-up papers) but relatively little has been written about identifying phrases of other syntactic categories. The CoNLL-2000 shared task attempts to fill this gap.

2 Task description

Text chunking consists of dividing a text into phrases in such a way that syntactically related words become member of the same phrase. These phrases are non-overlapping which means that one word can only be a member of one chunk. Here is an example sentence:

> [$_{\text{NP}}$ He] [$_{\text{VP}}$ reckons] [$_{\text{NP}}$ the current account deficit] [$_{\text{VP}}$ will narrow] [$_{\text{PP}}$ to] [$_{\text{NP}}$ only £ 1.8 billion] [$_{\text{PP}}$ in] [$_{\text{NP}}$ September] .

Chunks have been represented as groups of words between square brackets. A tag next to the open bracket denotes the type of the chunk. As far as we know, there are no annotated corpora available which contain specific information about dividing sentences into chunks of words of arbitrary types. We have chosen to work with a corpus with parse information, the Wall Street Journal (WSJ) part of the Penn Treebank II corpus (Marcus et al., 1993), and to extract chunk information from the parse trees in this corpus. We will give a global description of the various chunk types in the next section.

3 Chunk Types

The chunk types are based on the syntactic category part (i.e. without function tag) of the bracket label in the Treebank (cf. Bies (1995) p.35). Roughly, a chunk contains everything to the left of and including the syntactic head of the constituent of the same name. Some Treebank constituents do not have related chunks. The head of S (simple declarative clause) for example is normally thought to be the verb, but as the verb is already part of the VP chunk, no S chunk exists in our example sentence.

Besides the head, a chunk also contains premodifiers (like determiners and adjectives in NPs), but no postmodifiers or arguments. This is why the PP chunk only contains the preposition, and not the argument NP, and the SBAR chunk consists of only the complementizer.

There are several difficulties when converting trees into chunks. In the most simple case, a chunk is just a syntactic constituent without any further embedded constituents, like the NPs in our examples. In some cases, the chunk contains only what is left after other chunks have been removed from the constituent, cf. "(VP loves (NP Mary))" above, or ADJPs and PPs below. We will discuss some special cases during the following description of the individual chunk types.

3.1 NP

Our NP chunks are very similar to the ones of Ramshaw and Marcus (1995). Specifically, possessive NP constructions are split in front of the possessive marker (e.g. [$_{\text{NP}}$ Eastern Airlines] [$_{\text{NP}}$ ' creditors]) and the handling of coordinated NPs follows the Treebank annotators. However, as Ramshaw and Marcus do not describe the details of their conversion algorithm, results may differ in difficult cases, e.g. involving NAC and NX.[1]

An ADJP constituent inside an NP constituent becomes part of the NP chunk:

> (NP The (ADJP most volatile) form)
> → [$_{\text{NP}}$ the most volatile form]

3.2 VP

In the Treebank, verb phrases are highly embedded; see e.g. the following sentence which contains four VP constituents. Following Ramshaw and Marcus' V-type chunks, this sentence will only contain one VP chunk:

> ((S (NP-SBJ-3 Mr. Icahn) (VP may not (VP want (S (NP-SBJ *-3) (VP to (VP sell ...))))) .))
> → [$_{\text{NP}}$ Mr. Icahn] [$_{\text{VP}}$ may not want to sell] ...

It is still possible however to have one VP chunk directly follow another: [$_{\text{NP}}$ The impression] [$_{\text{NP}}$ I] [$_{\text{VP}}$ **have got]** [$_{\text{VP}}$ **is]** [$_{\text{NP}}$ they] [$_{\text{VP}}$ 'd love to do] [$_{\text{PRT}}$ away] [$_{\text{PP}}$ with] [$_{\text{NP}}$ it]. In this case the two VP constituents did not overlap in the Treebank.

Adverbs/adverbial phrases become part of the VP chunk (as long as they are in front of the main verb):

> (VP could (ADVP very well) (VP show ...))
> → [$_{\text{VP}}$ could very well show] ...

In contrast to Ramshaw and Marcus (1995), predicative adjectives of the verb are not part of the VP chunk, e.g. in "[$_{\text{NP}}$ they] [$_{\text{VP}}$ are] [$_{\text{ADJP}}$ unhappy]".

In inverted sentences, the auxiliary verb is not part of any verb phrase in the Treebank. Consequently it does not belong to any VP chunk:

> ((S (SINV (CONJP Not only) does (NP-SBJ-1 your product) (VP have (S (NP-SBJ *-1) (VP to (VP be (ADJP-PRD excellent)))))) , but ...
> → [$_{\text{CONJP}}$ Not only] does [$_{\text{NP}}$ your product] [$_{\text{VP}}$ have to be] [$_{\text{ADJP}}$ excellent] , but ...

3.3 ADVP and ADJP

ADVP chunks mostly correspond to ADVP constituents in the Treebank. However, ADVPs inside ADJPs or inside VPs if in front of the main verb are assimilated into the ADJP respectively VP chunk. On the other hand, ADVPs that contain an NP make two chunks:

> (ADVP-TMP (NP a year) earlier)
> → [$_{\text{NP}}$ a year] [$_{\text{ADVP}}$ earlier]

ADJPs inside NPs are assimilated into the NP. And parallel to ADVPs, ADJPs that contain an NP make two chunks:

> (ADJP-PRD (NP 68 years) old)
> → [$_{\text{NP}}$ 68 years] [$_{\text{ADJP}}$ old]

It would be interesting to see how changing these decisions (as can be done in the Treebank-to-chunk conversion script[2]) influences the chunking task.

3.4 PP and SBAR

Most PP chunks just consist of one word (the preposition) with the part-of-speech tag IN. This does not mean, though, that finding PP chunks is completely trivial. INs can also constitute an SBAR chunk (see below) and some PP chunks contain more than one word. This is the case with fixed multi-word prepositions such as *such as, because of, due to*, with prepositions preceded by a modifier: *well above, just after, even in, particularly among* or with coordinated prepositions: *inside and outside*. We think that PPs behave sufficiently differently from NPs in a sentence for not wanting to group them into one class (as Ramshaw and Marcus did in their N-type chunks), and that on the other hand tagging all NP chunks inside a PP as I-PP would only confuse the chunker. We therefore chose not to handle the recognition of true PPs (prep.+NP) during this first chunking step.

---

[1] E.g. (NP-SBJ (NP Robin Leigh-Pemberton) , (NP (NAC Bank (PP of (NP England))) governor) ,) which we convert to [$_{\text{NP}}$ Robin Leigh-Pemberton] , Bank [$_{\text{PP}}$ of] [$_{\text{NP}}$ England] [$_{\text{NP}}$ governor] whereas Ramshaw and Marcus state that ' "governor" is not included in any baseNP chunk'.

[2] The Treebank-to-chunk conversion script is available from http://ilk.kub.nl/~sabine/chunklink/

SBAR chunks mostly consist of one word (the complementizer) with the part-of-speech tag IN, but like multi-word prepositions, there are also multi-word complementizers: *even though, so that, just as, even if, as if, only if.*

3.5 CONJP, PRT, INTJ, LST, UCP

Conjunctions can consist of more than one word as well: *as well as, instead of, rather than, not only, but also.* One-word conjunctions (like *and, or*) are not annotated as CONJP in the Treebank, and are consequently no CONJP chunks in our data.

The Treebank uses the PRT constituent to annotate verb particles, and our PRT chunk does the same. The only multi-word particle is *on and off.* This chunk type should be easy to recognize as it should coincide with the part-of-speech tag RP, but through tagging errors it is sometimes also assigned IN (preposition) or RB (adverb).

INTJ is an interjection phrase/chunk like *no, oh, hello, alas, good grief!*. It is quite rare.

The list marker LST is even rarer. Examples are *1., 2., 3., first, second, a, b, c.* It might consist of two words: the number and the period.

The UCP chunk is reminiscent of the UCP (unlike coordinated phrase) constituent in the Treebank. Arguably, the conjunction is the head of the UCP, so most UCP chunks consist of conjunctions like *and* and *or*. UCPs are the rarest chunks and are probably not very useful for other NLP tasks.

3.6 Tokens outside

Tokens outside any chunk are mostly punctuation signs and the conjunctions in ordinary coordinated phrases. The word *not* may also be outside of any chunk. This happens in two cases: Either *not* is not inside the VP constituent in the Treebank annotation e.g. in

> ... (VP have (VP told (NP-1 clients)
> (S (NP-SBJ *-1) not (VP to (VP ship
> (NP anything))))))

or *not* is not followed by another verb (because the main verb is a form of *to be*). As the right chunk boundary is defined by the chunk's head, i.e. the main verb in this case, *not* is then in fact a postmodifier and as such not included in the chunk: "... [$_{\text{SBAR}}$ that] [$_{\text{NP}}$ there] [$_{\text{VP}}$ were] n't [$_{\text{NP}}$ any major problems] ."

3.7 Problems

All chunks were automatically extracted from the parsed version of the Treebank, guided by the tree structure, the syntactic constituent labels, the part-of-speech tags and by knowledge about which tags can be heads of which constituents. However, some trees are very complex and some annotations are inconsistent. What to think about a VP in which the main verb is tagged as NN (common noun)? Either we allow NNs as heads of VPs (not very elegant but which is what we did) or we have a VP without a head. The first solution might also introduce errors elsewhere... As Ramshaw and Marcus (1995) already noted: "While this automatic derivation process introduced a small percentage of errors on its own, it was the only practical way both to provide the amount of training data required and to allow for fully-automatic testing."

4 Data and Evaluation

For the CoNLL shared task, we have chosen to work with the same sections of the Penn Treebank as the widely used data set for base noun phrase recognition (Ramshaw and Marcus, 1995): WSJ sections 15–18 of the Penn Treebank as training material and section 20 as test material[3]. The chunks in the data were selected to match the descriptions in the previous section. An overview of the chunk types in the training data can be found in table 1. The data sets contain tokens (words and punctuation marks), information about the location of sentence boundaries and information about chunk boundaries. Additionally, a part-of-speech (POS) tag was assigned to each token by a standard POS tagger (Brill (1994) trained on the Penn Treebank). We used these POS tags rather than the Treebank ones in order to make sure that the performance rates obtained for this data are realistic estimates for data for which no treebank POS tags are available.

In our example sentence in section 2, we have used brackets for encoding text chunks. In the data sets we have represented chunks with three types of tags:

[3]The text chunking data set is available at http://lcg-www.uia.ac.be/conll2000/chunking/

count	%	type
55081	51%	NP (noun phrase)
21467	20%	VP (verb phrase)
21281	20%	PP (prepositional phrase)
4227	4%	ADVP (adverb phrase)
2207	2%	SBAR (subordinated clause)
2060	2%	ADJP (adjective phrase)
556	1%	PRT (particles)
56	0%	CONJP (conjunction phrase)
31	0%	INTJ (interjection)
10	0%	LST (list marker)
2	0%	UCP (unlike coordinated phrase)

Table 1: Number of chunks per phrase type in the training data (211727 tokens, 106978 chunks).

B-X	first word of a chunk of type X
I-X	non-initial word in an X chunk
O	word outside of any chunk

This representation type is based on a representation proposed by Ramshaw and Marcus (1995) for noun phrase chunks. The three tag groups are sufficient for encoding the chunks in the data since these are non-overlapping. Using these chunk tags makes it possible to approach the chunking task as a word classification task. We can use chunk tags for representing our example sentence in the following way:

He/B-NP reckons/B-VP the/B-NP current/I-NP account/I-NP deficit/I-NP will/B-VP narrow/I-VP to/B-PP only/B-NP £/I-NP 1.8/I-NP billion/B-NP in/B-PP September/B-NP ./O

The output of a chunk recognizer may contain inconsistencies in the chunk tags in case a word tagged I-X follows a word tagged O or I-Y, with X and Y being different. These inconsistencies can be resolved by assuming that such I-X tags start a new chunk.

The performance on this task is measured with three rates. First, the percentage of detected phrases that are correct (precision). Second, the percentage of phrases in the data that were found by the chunker (recall). And third, the $F_{\beta=1}$ rate which is equal to (β^2+1)*precision*recall / $(\beta^2$*precision+recall) with β=1 (van Rijsbergen, 1975). The latter rate has been used as the target for optimization[4].

5 Results

The eleven systems that have been applied to the CoNLL-2000 shared task can be divided in four groups:

1. Rule-based systems: Villain and Day; Johansson; Déjean.
2. Memory-based systems: Veenstra and Van den Bosch.
3. Statistical systems: Pla, Molina and Prieto; Osborne; Koeling; Zhou, Tey and Su.
4. Combined systems: Tjong Kim Sang; Van Halteren; Kudoh and Matsumoto.

Vilain and Day (2000) approached the shared task in three different ways. The most successful was an application of the Alembic parser which uses transformation-based rules. Johansson (2000) uses context-sensitive and context-free rules for transforming part-of-speech (POS) tag sequences to chunk tag sequences. Déjean (2000) has applied the theory refinement system ALLiS to the shared task. In order to obtain a system which could process XML formatted data while using context information, he has used three extra tools. Veenstra and Van den Bosch (2000) examined different parameter settings of a memory-based learning algorithm. They found that modified value difference metric applied to POS information only worked best.

A large number of the systems applied to the CoNLL-2000 shared task uses statistical methods. Pla, Molina and Prieto (2000) use a finite-state version of Markov Models. They started with using POS information only and obtained a better performance when lexical information was used. Zhou, Tey and Su (2000) implemented a chunk tagger based on HMMs. The initial performance of the tagger was improved by a post-process correction method based on error driven learning and by

[4]In the literature about related tasks sometimes the tagging accuracy is mentioned as well. However, since the relation between tag accuracy and chunk precision and recall is not very strict, tagging accuracy is not a good evaluation measure for this task.

test data	precision	recall	$F_{\beta=1}$
Kudoh and Matsumoto	93.45%	93.51%	93.48
Van Halteren	93.13%	93.51%	93.32
Tjong Kim Sang	94.04%	91.00%	92.50
Zhou, Tey and Su	91.99%	92.25%	92.12
Déjean	91.87%	92.31%	92.09
Koeling	92.08%	91.86%	91.97
Osborne	91.65%	92.23%	91.94
Veenstra and Van den Bosch	91.05%	92.03%	91.54
Pla, Molina and Prieto	90.63%	89.65%	90.14
Johansson	86.24%	88.25%	87.23
Vilain and Day	88.82%	82.91%	85.76
baseline	72.58%	82.14%	77.07

Table 2: Performance of the eleven systems on the test data. The baseline results have been obtained by selecting the most frequent chunk tag for each part-of-speech tag.

incorporating chunk probabilities generated by a memory-based learning process. The two other statistical systems use maximum-entropy based methods. Osborne (2000) trained Ratnaparkhi's maximum-entropy POS tagger to output chunk tags. Koeling (2000) used a standard maximum-entropy learner for generating chunk tags from words and POS tags. Both have tested different feature combinations before finding an optimal one and their final results are close to each other.

Three systems use system combination. Tjong Kim Sang (2000) trained and tested five memory-based learning systems to produce different representations of the chunk tags. A combination of the five by majority voting performed better than the individual parts. Van Halteren (2000) used Weighted Probability Distribution Voting (WPDV) for combining the results of four WPDV chunk taggers and a memory-based chunk tagger. Again the combination outperformed the individual systems. Kudoh and Matsumoto (2000) created 231 support vector machine classifiers to predict the unique pairs of chunk tags. The results of the classifiers were combined by a dynamic programming algorithm.

The performance of the systems can be found in Table 2. A baseline performance was obtained by selecting the chunk tag most frequently associated with a POS tag. All systems outperform the baseline. The majority of the systems reached an $F_{\beta=1}$ score between 91.50 and 92.50. Two approaches performed a lot better: the combination system WPDV used by Van Halteren and the Support Vector Machines used by Kudoh and Matsumoto.

6 Related Work

In the early nineties, Abney (1991) proposed to approach parsing by starting with finding related chunks of words. By then, Church (1988) had already reported on recognition of base noun phrases with statistical methods. Ramshaw and Marcus (1995) approached chunking by using a machine learning method. Their work has inspired many others to study the application of learning methods to noun phrase chunking[5]. Other chunk types have not received the same attention as NP chunks. The most complete work is Buchholz et al. (1999), which presents results for NP, VP, PP, ADJP and ADVP chunks. Veenstra (1999) works with NP, VP and PP chunks. Both he and Buchholz et al. use data generated by the script that produced the CoNLL-2000 shared task data sets. Ratnaparkhi (1998) has recognized arbitrary chunks as part of a parsing task but did not report on the chunking performance. Part of the Sparkle project has concentrated on finding various sorts of chunks for the different languages

[5]An elaborate overview of the work done on noun phrase chunking can be found on http://lcg-www.uia.ac.be/~erikt/research/np-chunking.html

(Carroll et al., 1997).

7 Concluding Remarks

We have presented an introduction to the CoNLL-2000 shared task: dividing text into syntactically related non-overlapping groups of words, so-called text chunking. For this task we have generated training and test data from the Penn Treebank. This data has been processed by eleven systems. The best performing system was a combination of Support Vector Machines submitted by Taku Kudoh and Yuji Matsumoto. It obtained an $F_{\beta=1}$ score of 93.48 on this task.

Acknowledgements

We would like to thank the members of the CNTS - Language Technology Group in Antwerp, Belgium and the members of the ILK group in Tilburg, The Netherlands for valuable discussions and comments. Tjong Kim Sang is funded by the European TMR network Learning Computational Grammars. Buchholz is supported by the Netherlands Organization for Scientific Research (NWO).

References

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

Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. *Bracket Guidelines for Treebank II Style Penn Treebank Project*. Penn Treebank II cdrom.

Eric Brill. 1994. Some advances in rule-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*. Seattle, Washington.

Sabine Buchholz, Jorn Veenstra, and Walter Daelemans. 1999. Cascaded grammatical relation assignment. In *Proceedings of EMNLP/VLC-99*. Association for Computational Linguistics.

John Carroll, Ted Briscoe, Glenn Carroll, Marc Light, Dethleff Prescher, Mats Rooth, Stefano Federici, Simonetta Montemagni, Vito Pirrelli, Irina Prodanof, and Massimo Vanocchi. 1997. *Phrasal Parsing Software*. Sparkle Work Package 3, Deliverable D3.2.

Kenneth Ward Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Second Conference on Applied Natural Language Processing*. Austin, Texas.

Hérve Déjean. 2000. Learning syntactic structures with xml. In *Proceedings of CoNLL-2000 and LLL-2000*. Lisbon, Portugal.

Christer Johansson. 2000. A context sensitive maximum likelihood approach to chunking. In *Proceedings of CoNLL-2000 and LLL-2000*. Lisbon, Portugal.

Rob Koeling. 2000. Chunking with maximum entropy models. In *Proceedings of CoNLL-2000 and LLL-2000*. Lisbon, Portugal.

Taku Kudoh and Yuji Matsumoto. 2000. Use of support vector learning for chunk identification. In *Proceedings of CoNLL-2000 and LLL-2000*. Lisbon, Portugal.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2).

Miles Osborne. 2000. Shallow parsing as part-of-speech tagging. In *Proceedings of CoNLL-2000 and LLL-2000*. Lisbon, Portugal.

Ferran Pla, Antonio Molina, and Natividad Prieto. 2000. Improving chunking by means of lexical-contextual information in statistical language models. In *Proceedings of CoNLL-2000 and LLL-2000*. Lisbon, Portugal.

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*. Cambridge, MA, USA.

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

Erik F. Tjong Kim Sang. 2000. Text chunking by system combination. In *Proceedings of CoNLL-2000 and LLL-2000*. Lisbon, Portugal.

Hans van Halteren. 2000. Chunking with wpdv models. In *Proceedings of CoNLL-2000 and LLL-2000*. Lisbon, Portugal.

C.J. van Rijsbergen. 1975. *Information Retrieval*. Buttersworth.

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

Jorn Veenstra. 1999. Memory-based text chunking. In Nikos Fakotakis, editor, *Machine learning in human language technology*. workshop at ACAI 99.

Marc Vilain and David Day. 2000. Phrase parsing with rule sequence processors: an application to the shared conll task. In *Proceedings of CoNLL-2000 and LLL-2000*. Lisbon, Portugal.

GuoDong Zhou, Jian Su, and TongGuan Tey. 2000. Hybrid text chunking. In *Proceedings of CoNLL-2000 and LLL-2000*. Lisbon, Portugal.