In: *Proceedings of CoNLL-2000 and LLL-2000*, pages 154–156, Lisbon, Portugal, 2000.

Chunking with WPDV Models

Hans van Halteren
Dept. of Language and Speech, Univ. of Nijmegen
P.O. Box 9103, 6500 HD Nijmegen
The Netherlands
hvh@let.kun.nl

1 Introduction

In this paper I describe the application of the WPDV algorithm to the CoNLL-2000 shared task, the identification of base chunks in English text (Tjong Kim Sang and Buchholz, 2000). For this task, I use a three-stage architecture: I first run five different base chunkers, then combine them and finally try to correct some recurring errors. Except for one base chunker, which uses the memory-based machine learning system TiMBL,[1] all modules are based on WPDV models (van Halteren, 2000a).

2 Architecture components

The first stage of the chunking architecture consists of five different **base chunkers**:

1) As a baseline, I use a stacked TiMBL model. For the first level, following Daelemans et al. (1999), I use as features all words and tags in a window ranging from five tokens to the left to three tokens to the right. For the second level (cf. Tjong Kim Sang (2000)), I use a smaller window, four left and two right, but add the IOB suggestions made by the first level for one token left and right (but not the focus).

2) The basic WPDV model uses as features the words in a window ranging from one left to one right, the tags in a window ranging from three left to three right, and the IOB suggestions for the previous two tokens.[2]

3) In the reverse WPDV model, the direction of chunking is reversed, i.e. it chunks from the end of each utterance towards the beginning.

4) In the R&M WPDV model, Ramshaw and Marcus's type of IOB-tags are used, i.e. starts of chunks are tagged with a B-tag only if the preceding chunk is of the same type, and with an I-tag otherwise.

5) In the LOB WPDV model, the Penn wordclass tags (as produced by the Brill tagger) are replaced by the output of a WPDV tagger trained on 90% of the LOB corpus (van Halteren, 2000b).

For all WPDV models, the number of features is too high to be handled comfortably by the current WPDV implementation. For this reason, I use a maximum feature subset size of four and a threshold frequency of two.[3]

The second stage consists of a **combination** of the outputs of the five base chunkers, using another WPDV model. Each chunker contributes a feature containing the IOB suggestions for the previous, current and next token. In addition, there is a feature for the word and a feature combining the (Penn-style) wordclass tags of the previous, current and next token. For the combination model, I use no feature restrictions, and the default hill-climbing procedure.

In the final stage, I apply **corrective measures** to systematic errors which are observed in the output of leave-one-out experiments on the training data. For now, I focus on the most frequent phrase type, the NP, and especially on one weak point: determination of the start position of NPs. I use separate WPDV models for each of the following cases:

1) Should a token now marked I-NP start a

[1] Cf. http://ilk.kub.nl/.

[2] For unseen data, i.e. while being applied, the IOB suggestions used are of course those suggested by the model itself, not the true ones.

[3] Cf. van Halteren (2000a). Also, the difference between training and running (correct IOB-tags vs model suggestions) leads to a low expected generalization quality of hill-climbing. I therefore stop climbing after a single effective step, but using an alternative climbing procedure, in which not only the single best multiplications/division is applied per step, but which during every step applies all multiplications/divisions that yielded improvements while the opposite operation did not.

Phrase type	Number in test set	TiMBL	WPDV basic	WPDV reverse	WPDV R&M	WPDV LOB	Combination	Corrective measures
ADJP	438	64.99	71.14	76.18	70.52	69.83	74.55	74.52
ADVP	866	75.03	78.96	79.83	78.16	78.50	80.09	79.86
CONJP	9	36.36	45.45	18.18	20.69	58.82	42.11	42.11
INTJ	2	66.67	66.67	66.67	66.67	0.00	66.67	66.67
LST	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NP	12422	91.85	92.65	92.56	92.00	92.35	93.72	93.84
PP	4811	95.66	96.53	96.85	96.06	96.65	97.09	97.10
PRT	106	63.10	73.63	68.60	74.07	73.45	74.31	74.31
SBAR	535	76.50	82.27	85.54	84.18	84.77	85.41	85.41
VP	4658	92.11	92.80	92.84	92.37	91.45	93.61	93.65
Overall	23852	91.15	92.29	92.47	91.72	91.90	93.26	93.32

Table 1: $F_{\beta=1}$ measurements for all systems (as described in the text). In addition we list the number of occurrences of each phrase type in the test set.

new NP?[4] Features used: the wordclass tag sequence within the NP up to the current token, the wordclass sequence within the NP from the current token, and the current, previous and next word within the NP.

2) Should a token now marked B-NP continue a preceding NP? Features used: type and structure (in terms of wordclass tags) of the current and the preceding two chunks, and the final word of the current and the preceding chunk.

3) Should (part of) a chunk now preceding an NP be part of the NP? Features used: type and structure (in wordclass tags) of the current, preceding and next chunk (the latter being the NP), and the final word of the current and next chunk.

For all three models, the number of different features is large. Normally, this would force the use of feature restrictions. The training sets are very small, however, so that the need for feature restrictions disappears and the full model can be used. On the other hand, the limited size of the training sets has as a disadvantage that hill-climbing becomes practically useless. For this reason, I do not use hill-climbing but simply take the initial first order weight factors.

Each token is subjected to the appropriate model, or, if not in any of the listed situations, left untouched. To remove (some) resulting inconsistencies, I let an AWK script then change the IOB-tag of all comma's and coordinators that now end an NP into O.

[4]This cannot already be the first token of an NP, as I-tags following a different type of chunk are always immediately transformed to B-tags.

3 Results

The $F_{\beta=1}$ scores for all systems are listed in Table 1. They vary greatly per phrase type, partly because of the relative difficulty of the tasks but also because of the variation in the number of relevant training and test cases: the most frequent phrase types (NP, PP and VP) also show the best results. Note that three of the phrase types (CONJP, INTJ and LST) are too infrequent to yield statistically sensible information.

The TiMBL results are worse than the ones reported by Buchholz et al. (1999),[5] but the latter were based on training on WSJ sections 00-19 and testing on 20–24. When comparing with the NP scores of Daelemans et al. (1999), we see a comparable accuracy (actually slightly higher because of the second level classification).

The WPDV accuracies are almost all much higher. For NP, the basic and reverse model produce accuracies which can compete with the highest published non-combination accuracies so far. Interestingly, the reverse model yields the best overall score. This can be explained by the observation that many choices, e.g. PP/PRT and especially ADJP/part of NP, are based mostly on the right context, about which more information becomes available when the text is handled from right to left. The R&M-type IOB-tags are generally less useful than the standard ones, but still show exceptional quality for some phrase types, e.g. PRT. The results for the LOB model are disappointing, given the overall quality of the tagger used

[5]F_{ADJP}=66.7, F_{ADVP}=77.9 F_{NP}=92.3, F_{PP}=96.8, F_{VP}=91.8

(97.82% on the held-out 10% of LOB). I hypothesize this to be due to: a) differences in text type between LOB and WSJ, b) partial incompatibility between the LOB tags and the WSJ chunks and c) insufficiency of chunker training set size for the more varied LOB tags.

Combination, as in other tasks (e.g. van Halteren et al. (To appear)), leads to an impressive accuracy increase, especially for the three most frequent phrase types, where there is a sufficient number of cases to train the combination model on. There are only two phrase types, ADVP and SBAR, where a base chunker (reverse WPDV) manages to outperform the combination. In both cases the four normal direction base chunkers outvote the better-informed reverse chunker, probably because the combination system has insufficient training material to recognize the higher information value of the reverse model (for these two phrase types). Even though the results are already quite good, I expect that even more effective combination is possible, with an increase in training set size and the inclusion of more base chunkers, especially ones which differ substantially from the current, still rather homogeneous, set.

The corrective measures yield further improvement, although less impressive. Unsurprisingly, the increase is found mostly for the NP. The next most affected phrase type is the ADJP, which can often be joined with or removed from the NP. There is an increase in recall for ADJP (71.23% to 71.46%), but a decrease in precision (78.20% to 77.86%), leaving the $F_{\beta=1}$ value practically unchanged. For ADVP, there is a loss of accuracy, most likely caused by the one-shot correction procedure. This loss will probably disappear when a procedure is used which is iterative and also targets other phrase types than the NP. For VP, on the other hand, there is an accuracy increase, probably due to a corrected inclusion/exclusion of participles into/from NPs. The overall scores show an increase, especially due to the per-type increases for the very frequent NP and VP.

All scores for the chunking system as a whole, including precision and recall percentages, are listed in Table 2. For all phrase types, the system yields substantially better results than any previously published. I attribute the improvements primarily to the combination architecture, with a smaller but yet valuable contribution by the corrective measures. The choice for WPDV proves a good one, as the WPDV algorithm is able to cope well with all the modeling tasks in the system. Whether it is the best choice can only be determined by future experiments, using other machine learning techniques in the same architecture.

test data	precision	recall	$F_{\beta=1}$
ADJP	77.86%	71.46%	74.52
ADVP	80.52%	79.21%	79.86
CONJP	40.00%	44.44%	42.11
INTJ	100.00%	50.00%	66.67
LST	0.00%	0.00%	0.00
NP	93.55%	94.13%	93.84
PP	96.43%	97.78%	97.10
PRT	72.32%	76.42%	74.31
SBAR	87.77%	83.18%	85.41
VP	93.36%	93.95%	93.65
all	93.13%	93.51%	93.32

Table 2: Final results per chunk type, i.e. after applying corrective measures to base chunker combination.

References

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

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

H. van Halteren. 2000a. A default first order family weight determination procedure for WPDV models. In *Proceedings of the CoNLL-2000*. Association for Computational Linguistics.

H. van Halteren. 2000b. The detection of inconsistency in manually tagged text. In *Proceedings of LINC2000*.

H. van Halteren, J. Zavrel, and W. Daelemans. To appear. Improving accuracy in wordclass tagging through combination of machine learning systems. *Computational Linguistics*.

E. F. Tjong Kim Sang. 2000. Noun phrase recognition by system combination. In *Proceedings of the ANLP-NAACL 2000*. Seattle, Washington, USA. Morgan Kaufman Publishers.

E. F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the CoNLL-2000*. Association for Computational Linguistics.