A Stacked, Voted, Stacked Model for Named Entity Recognition

Dekai WU[†*] dekai@cs.ust.hk
Grace NGAI[‡] csgngai@polyu.edu.hk
Marine CARPUAT[†] marine@cs.ust.hk

[†] Human Language Technology Center
HKUST
Department of Computer Science
University of Science and Technology
Clear Water Bay, Hong Kong

[‡] Hong Kong Polytechnic University
Department of Computing
Kowloon
Hong Kong

Abstract

This paper investigates stacking and voting methods for combining strong classifiers like boosting, SVM, and TBL, on the named-entity recognition task. We demonstrate several effective approaches, culminating in a model that achieves error rate reductions on the development and test sets of 63.6% and 55.0% (English) and 47.0% and 51.7% (German) over the CoNLL-2003 standard baseline respectively, and 19.7% over a strong AdaBoost baseline model from CoNLL-2002.

1 Introduction

We describe multiple stacking and voting methods that effectively combine strong classifiers such as boosting, SVM, and TBL, for the named-entity recognition (NER) task. NER has emerged as an important step for many natural language applications, including machine translation, information retrieval and information extraction. Much of the research in this field was pioneered in the Message Understanding Conference (MUC) (Sundheim, 1995), which performed detailed entity extraction and identification on English documents. As a result, most current NER systems which have impressive performances have been specially constructed and tuned for English MUC-style documents. It is unclear how well they would perform when applied to another language.

Our system was designed for the CoNLL-2003 shared task, the goal of which is to identify and classify four types of named entities: PERSON, LOCATION, ORGANIZATION and MISCELLANEOUS. The task specifications were that two languages would be involved. We were given about a month to develop our system on the first language, which was English, but only two weeks to adapt it to the surprise language, which was German.

Given the goal of the shared task, we designed our system to achieve a high performance without relying too heavily on knowledge that is very specific for a particular language or domain. In the spirit of language-independence, we avoided using features and information which would not be easily obtainable for almost any major language.

2 Classification Methods

To carry out the stacking and voting experiments, we constructed a number of relatively strong individual component models of the following kinds.

2.1 Boosting

The main idea behind boosting algorithms is that a set of many weak classifiers can be effectively combined to yield a single strong classifier. Each weak classifier is trained sequentially, increasingly focusing more heavily on the instances that the previous classifiers found difficult to classify.

For the boosting framework, our system uses AdaBoost.MH (Freund and Schapire, 1997), an n-ary classification variant of the original binary AdaBoost algorithm. It performs well on a number of natural language processing problems, including text categorization (Schapire and Singer, 2000) and word sense disambiguation (Escudero et al., 2000). In particular, it has also been demonstrated that boosting can be used to build language-independent NER models that perform exceptionally well (Wu et al., 2002; Carreras et al., 2002).

2.2 Support Vector Machines

Support Vector Machines (SVMs) have gained a considerable following in recent years (Boser et al., 1992), particularly in dealing with high-dimensional spaces

*The author would like to thank the Hong Kong Research Grants Council (RGC) for supporting this research in part through two research grants (RGC 6083/99E and RGC 6256/00E).

such as commonly found in natural language problems like text categorization (Joachims, 1998). SVMs have shown promise when applied to chunking (Kudo and Matsumoto, 2001) and named entity recognition (Sassano and Utsuro, 2000; McNamee and Mayfield, 2002), though performance is quite sensitive to parameter choices.

2.3 Transformation-based Learning

Transformation-based learning (Brill, 1995) (TBL) is a rule-based machine learning algorithm that was first introduced by Brill and used for part-of-speech tagging. The central idea of transformation-based learning is to learn an ordered list of rules which progressively improve upon the current state of the training set. An initial assignment is made based on simple statistics, and then rules are greedily learned to correct the mistakes, until no net improvement can be made.

The experiments presented in this paper were performed using the fnTBL toolkit (Ngai and Florian, 2001), which implements several optimizations in rule learning to drastically speed up the time needed for training.

3 Data Resources

3.1 Preprocessing the Data

The data that was provided by the CoNLL organizers was sentence-delimited and tokenized, and hand-annotated with named entity chunks. The English data was automatically labeled with part-of-speech and chunk tags from the memory-based tagger and chunker (Daelemans et al., 1996), and the German data was labelled with the decision-tree-based TreeTagger (Schmidt, 1994). We replaced the English part-of-speech tags with those generated by a transformation-based learner (Ngai and Florian, 2001). The chunk tags did not appear to help in either case and were discarded.

As we did not want to overly rely on characteristics which were specific to the Indo-European language family, we did not perform detailed morphological analysis; but instead, an approximation was made by simply extracting the prefixes and suffixes of up to 4 characters from all the words.

In order to let the system generalize over word types, we normalized the case information of all the words in the corpus by converting them to uniform lower case. To recapture the lost information, each word was annotated with a tag that specified if it was in all lower case, all upper case, or was of mixed case.

3.2 Gazetteers

Apart from the training and test data, the CoNLL organizers also provided two lists of named entities, one in English and one in Dutch. Part of the challenge for this year's shared task was to find ways of using this resource in the system.

To supplement the provided gazetteers, a large collection of names and words was downloaded from various web sources. This collection was used to compile a gazetteer of 120k uncategorized English proper names and a lexicon of 500k common English words. As there were no supplied gazetteers for German, we also compiled a gazetteer of 8000 German names, which were mostly personal first and last names and geographical locations, and a lexicon of 32k common German words.

Named entities in the corpus which appeared in the gazetteers were identified lexically or using a maximum forward match algorithm similar to that used in Chinese word segmentation. Once named entities have been identified in this preprocessing step, each word can then be annotated with an NE chunk tag corresponding to the output from the system. The learner can view the NE chunk tag as an additional feature.

The variations in this approach come from resolving conflicts between different possible type information for the same NE. The different ways that we dealt with the problem were: (1) Rank all the NE types by frequency in the training corpus. In the case of a conflict, default to the more common NE. (2) Give all the possible NEs to the boosting learner as a set of possible NE chunk tags. (3) Discard the NE type information and annotate each word with a tag indicating whether it is inside an NE.

4 Classifier Combination

It is a well-known fact that if several classifiers are available, they can be combined in various ways to create a system that outperforms the best individual classifier. Since we had several classifiers available to us, it was reasonable to investigate combining them in different ways.

4.1 Stacking

Like voting, stacking is a learning paradigm that constructs a combined model from several classifiers. The basic concept behind stacking is to train two or more classifiers sequentially, with each successive classifier incorporating the results of the previous ones in some fashion.

4.1.1 Integration of External Resources

As mentioned above, at the most basic level, lexicon and gazetteer information was integrated into our classifiers by including them as additional features. However we also experimented with several different ways of incorporating this information via stacking—one possible approach was to view the gazetteers as a separate system that would produce an output and then implement stacking to combine their outputs.

4.1.2 Division into Subtasks

One of the most straightforward approaches to stacking can be applied to tasks that are naturally divisible into hierarchically ordered subtasks. An example approach, which was taken by several of the participating teams in the CoNLL-2002 shared task, is to split the NER task into the identification phase, where named entities are identified in the text; and the classification phase, where the identified named entities are categoried into the various subtypes. Provided that the performance of the individual classifier is fairly high (otherwise errors made in the earlier stages could propagate down the chain), this has the advantage of reducing the complexity of the task for each individual classifier.

To construct such a system, we trained a stacked AdaBoost.MH classifier to perform NE reclassification on boundaries identified in the base model. The output of the initial models are postprocessed to remove all NE type information and then passed to this stacked classifier. As Table 1 shows, stacking the boosting models yields a significant gain in performance.

English devel.	precision	recall	$F_{\beta=1}$
(Boost) Base	88.64%	87.68%	88.16
(Boost) Base + Stacked	**89.26%**	**88.29%**	**88.77**

Table 1: Improving classification of NE types via stacked AdaBoost.MH.

4.1.3 Error Correction

Another approach to stacking that we investigated in this work involves a closer interaction between the models. The general overview of this approach is for a given model to use the output of another trained model as its initial state, and to improve beyond it. The idea is that the second model, with a different learning and representation bias, will be able to move out of the local maxima that the previous model has settled into.

To accomplish this we introduced *Stacked TBL* (STBL), a variant of TBL tuned for this purpose (Wu et al., 2003). We found TBL to be an appropriate point of departure since it starts from an initial state of classification and learns rules to iteratively correct the current labeling. We aimed to use STBL to improve the base model from the preceding section.

STBL proved quite effective; in fact it yielded the best base model performance among all our models. Table 2 shows the result of stacking STBL on the boosting base model.

English devel.	precision	recall	$F_{\beta=1}$
(Boost) Base	88.64%	87.68%	88.16
(Boost + STBL) Base	**87.83%**	**88.79%**	**88.31**

Table 2: Improving the above AdaBoost.MH base model, via Stacked TBL (STBL).

4.2 Voting

The simplest approach to combining classifiers is through voting, which examines the outputs of the various models and selects the classifications which have a weight exceeding some threshold, where the weight is dependent upon the models that proposed this particular classification. The variations in this approach stem from the method by which weights are attached to the models. It is possible to assign varying weights to the models, in effect giving one model more "say" than the others. In our system, however, we simply assigned each model equal weight, and selected classifications which were proposed by a majority of models.

Voting was thus used to further improve the base model. Four models chosen for heterogeneity participated in the voting: two variants of the AdaBoost.MH model, the SVM model, and the Stacked TBL model.

As before, the stacked AdaBoost.MH reclassifier was applied to the voted result, yielding a final *stacked voted stacked model*.

This model gave the best overall results on the task as a whole. Table 3 shows the results of our system.

English devel.	precision	recall	$F_{\beta=1}$
Boost_1 + Stacked	89.26%	88.29%	88.77
Boost_2 + Stacked	82.98%	85.62%	84.28
SVM + Stacked	84.41%	85.71%	85.05
Boost + STBL + Stacked	89.09%	88.07%	88.57
Voted + Stacked	**90.18%**	**88.86%**	**89.51**

Table 3: Improving classification of NE types, via stacked voted stacked AdaBoost.MH, STBL, and SVM models.

5 Overall Results

Complete results on the development and test sets, for both English and German, are shown in Table 4.

6 Conclusion

This paper has presented an overview of our entry to the CoNLL-2003 shared task. As individual component models, we constructed strong AdaBoost.MH models, SVM models, and Stacked TBL models, and provided them with detailed features on the data.

We then demonstrated several stacking and voting models that proved capable of improving performance further. This was non-trivial since the individual component models were all quite strong to begin with. Because

English devel.	precision	recall	$F_{\beta=1}$
LOC	91.88%	92.98%	92.42
MISC	91.53%	83.19%	87.16
ORG	86.43%	80.76%	83.50
PER	90.39%	93.49%	91.91
overall	90.18%	88.86%	89.51

English Test	precision	recall	$F_{\beta=1}$
LOC	88.24%	86.81%	87.52
MISC	79.08%	78.06%	78.57
ORG	79.31%	73.63%	76.37
PER	83.09%	88.44%	85.68
overall	83.06%	82.31%	82.69

German devel.	precision	recall	$F_{\beta=1}$
LOC	75.13%	61.39%	67.57
MISC	75.46%	45.05%	56.42
ORG	80.05%	47.86%	59.91
PER	74.19%	61.96%	67.52
overall	75.92%	54.67%	63.56

German Test	precision	recall	$F_{\beta=1}$
LOC	74.94%	60.97%	67.23
MISC	72.77%	51.04%	60.00
ORG	61.22%	42.69%	50.30
PER	83.68%	73.39%	78.20
overall	75.20%	59.35%	66.34

Table 4: Overall results of stacked voted stacked model on development and test sets.

of this the vast majority of classifier combination models we tested actually turned out to *degrade* performance, or showed zero improvement. The models presented here worked well because they were each motivated by detailed analyses.

We did investigate a number of ways in which gazetteers could be incorporated. The gazetteer supplied for the shared task was found not to improve performance significantly, because our models were already adequately powerful to correctly identify most of the named entities supplied by the gazetteer. However, minimal effort to augment the gazetteers did result in a performance boost. Moreover, performance was further improved by the inclusion of a common word lexicon not containing any named entities.

Inspection revealed that some errors found in the output of the system stemmed from either erroneous sentence boundaries in the test data, or difficult-to-avoid inconsistencies in the the gold standard annotations. For example, in the following:

1. … [Panamanian $]_{MISC}$ boxing legend …
2. … [U.S. $]_{LOC}$ collegiate national champion …

both "Panamanian" and "U.S." are used as modifiers, but one is annotated as a MISC-type NE while the other is considered a LOC-type.

The stacked voted stacked model obtained an improvement of 4.83 F-Measure points on the English development set over our best model from the CoNLL-2002 shared task which we took as our baseline, resulting in a substantial 19.7% error rate reduction. The system achieves this respectable performance using very little in the way of outside resources—only a part-of-speech tagger and some common wordlists—which can be obtained easily for almost any major language. Most features we used can also be used for uninflected and non-Indo-European languages such as Chinese, where the prefixes and suffixes can be replaced by decomposing the words at the character level. This is in keeping with the the language-independent spirit of the shared task.

References

Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. 1992. A training algorithm for optimal margin classifiers. In David Haussler, editor, *Proceedings of the 4th Workshop on Computational Learning Theory*, pages 144–152, San Mateo, CA. ACM Press.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–565.

Xavier Carreras, Lluís Màrquez, and Lluís Padró. 2002. Named entity extraction using adaboost. In *Proceedings of CoNLL-2002*, pages 167–170. Taipei, Taiwan.

Walter Daelemans, Jakub Zavrel, and Peter Berck. 1996. MBT: A memory-based part of speech tagger-generator.

Gerard Escudero, Lluís Màrquez, and German Rigau. 2000. Boosting applied to word sense disambiguation. In *European Conference on Machine Learning*, pages 129–141.

Yoav Freund and Rob E. Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. In *Journal of Computer and System Sciences, 55(1)*, pages 119–139.

Thorsten Joachims. 1998. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE. Springer Verlag, Heidelberg, DE.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of NAACL-2001*.

Paul McNamee and James Mayfield. 2002. Entity extraction without language-specific resources. In *Proceedings of CoNLL-2002*, pages 183–186. Taipei, Taiwan.

Grace Ngai and Radu Florian. 2001. Transformation-based learning in the fast lane. In *Proceedings of NAACL'01*, pages 40–47, Pittsburgh, PA. ACL.

Manabu Sassano and Takehito Utsuro. 2000. Named entity chunking techniques in supervised learning for Japanese named entity recognition. In *Proceedings of COLING-2000*, pages 705–711.

Rob E. Schapire and Yoram Singer. 2000. BoosTexter: A boosting-based system for text categorization. In *Machine Learning, 39(2/3*, pages 135–168.

Helmut Schmidt. 1994. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Natural Language Processing*, pages 44–49, Manchester, U.K.

Beth Sundheim. 1995. MUC6 named entity task defnition, version 2.1. In *Proceedings of the Sixth Message Understanding Conference (MUC6)*.

Dekai Wu, Grace Ngai, Marine Carpuat, Jeppe Larsen, and Yongsheng Yang. 2002. Boosting for named entity recognition. In *Proceedings of CoNLL-2002*, pages 195–198. Taipei, Taiwan.

Dekai Wu, Grace Ngai, and Marine Carpuat. 2003. Forthcoming.