A Stacked, Voted, Stacked Model for Named Entity Recognition

Dekai Wu†∗ and Grace Ngai‡ and Marine Carpuat†

† Human Language Technology Center
HKUST
Department of Computer Science
University of Science and Technology
Clear Water Bay, Hong Kong
{dekai,marine}@cs.ust.hk

‡ Hong Kong Polytechnic University
Department of Computing
Kowloon
Hong Kong
csgngai@polyu.edu.hk

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
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 categorized 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.

<table>
<thead>
<tr>
<th>English devel.</th>
<th>Precision</th>
<th>Recall</th>
<th>F_{β=1}</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Boost) Base</td>
<td>88.64%</td>
<td>87.68%</td>
<td>88.16%</td>
</tr>
<tr>
<td>(Boost) Base + Stacked</td>
<td>89.26%</td>
<td>88.29%</td>
<td>88.77%</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>English devel.</th>
<th>Precision</th>
<th>Recall</th>
<th>F_{β=1}</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Boost) Base</td>
<td>88.64%</td>
<td>87.68%</td>
<td>88.16%</td>
</tr>
<tr>
<td>(Boost + STBL) Base</td>
<td>87.83%</td>
<td>88.79%</td>
<td>88.31%</td>
</tr>
</tbody>
</table>

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 mod-
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.

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

<table>
<thead>
<tr>
<th>Language</th>
<th>Precision</th>
<th>Recall</th>
<th>F_β=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>English devel.</td>
<td>91.88%</td>
<td>92.98%</td>
<td>92.42</td>
</tr>
<tr>
<td>LOC</td>
<td>86.27%</td>
<td>85.91%</td>
<td>86.09</td>
</tr>
<tr>
<td>MISC</td>
<td>75.88%</td>
<td>77.07%</td>
<td>76.47</td>
</tr>
<tr>
<td>ORG</td>
<td>78.19%</td>
<td>72.97%</td>
<td>75.49</td>
</tr>
<tr>
<td>PER</td>
<td>83.94%</td>
<td>87.26%</td>
<td>85.57</td>
</tr>
<tr>
<td>Overall</td>
<td>90.18%</td>
<td>88.86%</td>
<td>89.51</td>
</tr>
<tr>
<td>German devel.</td>
<td>75.13%</td>
<td>61.39%</td>
<td>67.57</td>
</tr>
<tr>
<td>LOC</td>
<td>75.46%</td>
<td>45.05%</td>
<td>56.42</td>
</tr>
<tr>
<td>MISC</td>
<td>80.05%</td>
<td>47.86%</td>
<td>59.91</td>
</tr>
<tr>
<td>ORG</td>
<td>74.19%</td>
<td>61.96%</td>
<td>67.52</td>
</tr>
<tr>
<td>PER</td>
<td>75.92%</td>
<td>54.67%</td>
<td>63.56</td>
</tr>
<tr>
<td>Overall</td>
<td>75.20%</td>
<td>59.35%</td>
<td>66.34</td>
</tr>
</tbody>
</table>

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

References