Named Entity Recognition through Classifier Combination

Radu Florian and **Abe Ittycheriah** and **Hongyan Jing** and **Tong Zhang**
{raduf,abei,hjing,tzhang}@us.ibm.com
IBM T.J. Watson Research Center
1101 Kitchawan Rd, Yorktown Heights, NY 10598, USA

Abstract

This paper presents a classifier-combination experimental framework for named entity recognition in which four diverse classifiers (robust linear classifier, maximum entropy, transformation-based learning, and hidden Markov model) are combined under different conditions. When no gazetteer or other additional training resources are used, the combined system attains a performance of 91.6F on the English development data; integrating name, location and person gazetteers, and named entity systems trained on additional, more general, data reduces the F-measure error by a factor of 15 to 21% on the English data.

1 Introduction

This paper investigates the combination of a set of diverse statistical named entity classifiers, including a rule-based classifier – the t*ransformation-based learning* classifier (Brill, 1995; Florian and Ngai, 2001, henceforth fnTBL) with the forward-backward extension described in Florian (2002a), a *hidden Markov model* classifier (henceforth HMM), similar to the one described in Bikel et al. (1999), a *robust risk minimization classifier,* based on a regularized winnow method (Zhang et al., 2002) (henceforth RRM) and a *maximum entropy* classifier (Darroch and Ratcliff, 1972; Berger et al., 1996; Borthwick, 1999) (henceforth MaxEnt). This particular set of classifiers is diverse across multiple dimensions, making it suitable for combination:

- fnTBL is a *discriminant* classifier – it bases its classification decision only on the few most discriminant features active on an example – while HMM, RRM and MaxEnt are *agglomerative* classifiers – their decision is based on the combination of all features active for the particular example.
- In dealing with the data sparseness problem, fnTBL, MaxEnt and RRM investigate and integrate in their decision arbitrary feature types, while HMM is dependent on a prespecified back-off path.
- The search methods employed by each classifier are different: the HMM, MaxEnt and RRM classifiers construct a model for each example and then rely on a sequence search such as the Viterbi algorithm (Viterbi, 1967) to identify the best overall sequence, while fnTBL starts with most frequent classification (usually per token), and then dynamically models the interaction between classifications, effectively performing the search at training time.
- The classifiers also differ in their output: fnTBL and RRM return a single classification per example[1], while the MaxEnt and HMM classifiers return a probability distribution.

The remainder of the paper is organized as follows: Section 2 describes the features used by the classifiers, Section 3 briefly describes the algorithms used by each classifier, and Section 4 analyzes in detail the results obtained by each classifier and their combination.

2 The Classification Method and Features Used

All algorithms described in this paper identify the named entities in the text by labeling each word with a tag corresponding to its position relative to a named entity: whether it starts/continues/ends a specific named entity, or does not belong to any entity. RRM, MaxEnt, and fnTBL treat the problem entirely as a tagging task, while the HMM algorithm used here is constraining the transitions between the various phases, similar to the method described in (Bikel et al., 1999).

Feature design and integration is of utmost importance in the overall classifier design – a rich feature space is the key to good performance. Often, high performing classifiers operating in an impoverished space are surpassed by a lower performing classifier when the latter has access to enhanced feature spaces (Zhang et al., 2002; Florian,

[1] However, both classifiers' algorithms can be modified such that a class probability distribution is returned instead.

2002a). In accordance with this observation, the classifiers used in this research can access a diverse set of features when examining a word in context, including:

- words and their lemmas in a 5-word-window surrounding the current word
- the part-of-speech tags of the current and surrounding words
- the text chunks in a -1..1 window
- the prefixes and suffixes of length up to 4 of the current and the surrounding words
- a word feature flag for each word, similar to the flag described in (Bikel et al., 1999); examples of such assigned flags are *firstCap*, *2digit* and *allCaps*.
- gazetteer information, in the form of a list of 50,000 cities, 80,000 proper names and 3500 organizations
- the output of other two named entity classifiers, trained on a richer tagset data (32 named categories), used in the IBM question answering system (Ittycheriah et al., 2001)

In addition, a ngram-based capitalization restoration algorithm has been applied on the sentences that appear in all caps[2], for the English task.

3 The Algorithms

This section describes only briefly the classifiers used in combination in Section 4; a full description of the algorithms and their properties is beyond the scope of this paper – the reader is instead referred to the original articles.

3.1 The Robust Risk Minimization Classifier

This classifier is described in detail in (Zhang and Johnson, 2003, this volume), along with a comprehensive evaluation of its performance, and therefore is not presented here.

3.2 The Maximum Entropy Classifier

The MaxEnt classifier computes the posterior class probability of an example by evaluating the normalized product of the weights active for the particular example. The model weights are trained using the improved iterative scaling algorithm (Berger et al., 1996). To avoid running in severe over-training problems, a feature cutoff of 4 is applied before the model weights are learned. At decoding time, the best sequence of classifications is identified with the Viterbi algorithm.

3.3 The Transformation-Based Learning Classifier

Transformation-based learning is an error-driven algorithm which has two major steps: it starts by assigning some classification to each example, and then automatically proposing, evaluating and selecting the classification changes that maximally decrease the number of errors.

	English		German	
	(a)	(b)	(a)	(b)
HMM	82.0	74.6	-	-
TBL	88.1	81.2	69.5	68.6
MaxEnt	90.8	85.6	68.0	67.3
RRM	92.1	85.5	70.7	71.3

Tab. 1: Individual classifier results on the two test sets.

TBL has some attractive qualities that make it suitable for the language-related tasks: it can automatically integrate heterogeneous types of knowledge, without the need for explicit modeling, it is error–driven, and has an inherently dynamic behavior.

The particular setup in which fnTBL is used in this work is described in Florian (2002a): in a first phase, TBL is used to identify the entity boundaries, followed by a sequence classification stage, where the entities identified at the first step are classified using internal and external clues[3].

3.4 The Hidden Markov Model Classifier

The HMM classifier used in the experiments in Section 4 follows the system description in (Bikel et al., 1999), and it performs sequence classification by assigning each word either one of the named entity types or the label NOT-A-NAME to represent "not a named entity". The states in the HMM are organized into regions, one region for each type of named entity plus one for NOT-A-NAME. Within each of the regions, a statistical bigram language model is used to compute the likelihood of words occurring within that region (named entity type). The transition probabilities are computed by deleted interpolation (Jelinek, 1997), and the decoding is done through the Viterbi algorithm. The particular implementation we used underperformed consistently all the other classifiers on German, and is not included.

4 Combination Methodology and Experimental Results

The results obtained by each individual classifier, broken down by entity type, are presented in Table 1. Out of the four classifiers, the MaxEnt and RRM classifiers are the best performers, followed by the modified fnTBL classifier and the HMM classifier. The error-based classifiers (RRM and fnTBL) tend to obtain balanced precision/recall numbers, while the other two tend to be more precise at the expense of recall. To facilitate comparison with other classifiers for this task, most reported results

[2] Usually, document titles, but also table headers, etc.

[3] The method of retaining only the boundaries and reclassifying the entities was shown to improve the performance of 11 of the 12 systems participating in the CoNLL-2002 shared tasks, in both languages (Florian, 2002b).

are obtained by using features exclusively extracted from the training data.

In general, given n classifiers, one can interpret the classifier combination framework as combining probability distributions:

$$P\left(C|w,C_1^n\right) = f\left(\left(P_i\left(C|w,C_1^n\right)\right)_{i=1\ldots n}\right) \qquad (1)$$

where C_i is the classifier i's classification output, f is a combination function. A widely used combination scheme is through linear interpolation of the classifiers' class probability distribution

$$\begin{aligned} P\left(C|w,C_1^n\right) &= \sum_{i=1}^{n} P\left(C|w,i,C_i\right) \cdot P\left(i|w\right) \\ &= \sum_{i=1}^{n} P_i\left(C|w,C_i\right) \cdot \lambda_i\left(w\right) \qquad (2) \end{aligned}$$

The weights $\lambda_i\left(w\right)$ encode the importance given to classifier i in combination, for the context of word w, and $P_i\left(C|w,C_i\right)$ is an estimation of the probability that the correct classification is C, given that the output of the classifier i on word w is C_i.

To estimate the parameters in Equation (2), the provided training data was split into 5 equal parts, and each classifier was trained, in a round-robin fashion, on 4 fifths of the data and applied on the remaining fifth. This way, the entire training data can be used to estimate the weight parameters $\lambda_i\left(w\right)$ and $P_i\left(C|w,C_i\right)$ but, at decoding time, the individual classifier outputs C_i are computed by using the entire training data.

Table 2 presents the combination results, for different ways of estimating the interpolation parameters. A simple combination method is the *equal voting* method (van Halteren et al., 2001; Tjong Kim Sang et al., 2000), where the parameters are computed as $\lambda_i\left(w\right) = \frac{1}{n}$ and $P_i\left(C|w,C_i\right) = \delta\left(C,C_i\right)$, where δ is the Kronecker operator ($\delta\left(x,y\right) := \left(x = y?1:0\right)$) – each of the classifiers votes with equal weight for the class that is most likely under its model, and the class receiving the largest number of votes wins. However, this procedure may lead to ties, where some classes receive the same number of votes – one usually resorts to randomly selecting one of the tied candidates in this case – Table 2 presents the average results obtained by this method, together with the variance obtained over 30 trials. To make the decision deterministically, the weights associated with the classifiers can be chosen as $\lambda_i\left(w\right) = P_i\left(error\right)$. In this method, presented in Table 2 as *weighted voting,* better performing classifiers will have a higher impact in the final classification.

Method	*Precision*	*Recall*	*Fmeasure*
Best Classifier	91.37%	88.56%	89.94
Equal voting	91.5±0.13	91.0±0.06	91.23±0.08
Weighted voting	92.13%	91.00%	91.56
Model 1	90.99%	90.81%	90.9
Model 2	92.43%	90.86%	**91.64**
RRM	92.01%	91.25%	**91.63**

Tab. 2: Classifier combination results on English devset data

	Development		Test	
Language	Unique	Corpus	Unique	Corpus
English	33.4%	8.0%	40.3%	11.7%
German	52%	16.2%	48.6%	14.2%

Tab. 3: Word statistics (percent unknown words)

In the voting methods, each classifier gave its entire vote to one class – its own output. However, Equation (2) allows for classifiers to give partial credit to alternative classifications, through the probability $P_i\left(C|w,C_i\right)$. In our experiments, this value is computed through 5-fold cross-validation on the training data. The space of possible choices for C, w and C_i is large enough to make the estimation unreliable, so we use two approximations, named *Model 1* and *Model 2* in Table 2: $P_i\left(C|w,C_i\right) = P_i\left(C|w\right)$and $P_i\left(C|w,C_i\right) = P_i\left(C|C_i\right)$, respectively. On the development data, the former estimation type obtains a lower performance than the latter.

In a last experiment using only features extracted from the training data, we use the RRM method to compute the function f in Equation (1), allowing the system to select a good performing combination of features. At training time, the system was fed the output of each classifier on the cross-classified data, the part-of-speech and chunk boundary tags. At test time, the system was fed the classifications of each system trained on the entire training data, and the corresponding POS and chunk boundary tags. The result obtained rivals the one obtained by model 2, both displaying a 17% reduction in F-measure error[4], indicating that maybe all sources of information have been explored and incorporated.

The RRM method is showing its combining power when additional information sources are used. Specifically, the system was fed additional feature streams from a list of gazetteers and the output of two other named entity systems trained on 1.7M words annotated with 32 name categories. The RRM system alone obtains an F-measure of 92.1, and can effectively integrate these information streams with the output of the four classifiers, gazetteers and the two additional classifiers into obtaining 93.9 F-measure, as detailed in Table 4, a 21% reduction in F-measure error. In contrast, combination model 2 obtains only a performance of 92.4, showing its limitations

[4] Measured as $100 - F$.

English devel	precision	recall	$F_{\beta=1}$
LOC	96.59%	95.65%	96.12
MISC	90.77%	87.42%	89.06
ORG	90.85%	89.63%	90.24
PER	96.08%	97.12%	96.60
overall	94.26%	93.47%	**93.87**

English test	precision	recall	$F_{\beta=1}$
LOC	90.59%	91.73%	91.15
MISC	83.46%	77.64%	80.44
ORG	85.93%	83.44%	84.67
PER	92.49%	95.24%	93.85
overall	88.99%	88.54%	**88.76**

German devel	precision	recall	$F_{\beta=1}$
LOC	83.19%	72.90%	77.71
MISC	83.20%	42.18%	55.98
ORG	83.64%	61.80%	71.08
PER	87.43%	67.02%	75.88
overall	84.60%	61.93%	**71.51**

German test	precision	recall	$F_{\beta=1}$
LOC	80.19%	71.59%	75.65
MISC	77.87%	41.49%	54.14
ORG	79.43%	54.46%	64.62
PER	91.93%	75.31%	82.80
overall	83.87%	63.71%	**72.41**

Tab. 4: Results on the devel and test sets in English and German

in combining diverse sources of information.

German poses a completely different problem for named entity recognition: the data is considerably sparser. Table 3 shows the relative distribution of unknown words in the development and test corpora. We note that the numbers are roughly twice as large for the development data in German as they are for English. Since the unknown words are classed by most classifiers, this results in few data points to estimate classifier combinations. Also, specifically for the German data, traditional approaches which utilize capitalization do not work as well as in English, because all nouns are capitalized in German.

For German, in addition to the entity lists provided, we also used a small gazetteer of names (4500 first and last names, 4800 locations in Germany and 190 countries), which was collected by browsing web pages in about two person-hours. The average classifier performance gain by using these features is about 1.5F for the *testa* data and about .6F for the *testb* data.

5 Conclusion

In conclusion, we have shown results on a set of both well-established and novel classifier techniques which improve the overall performance, when compared with the best performing classifier, by 17-21% on the English task. For the German task, the improvement yielded by classifier combination is smaller. As a machine learning method, the RRM algorithm seems especially suited to handle additional feature streams, and therefore is a good candidate for classifier combination.

References

A. Berger, S. Della Pietra, and V. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Daniel M. Bikel, Richard L. Schwartz, and Ralph M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231.

A. Borthwick. 1999. *A Maximum Entropy Approach to Named Entity Recognition*. Ph.D. thesis, New York University.

E. 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.

J. N. Darroch and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480.

R. Florian and G. Ngai, 2001. *Fast Transformation-Based Learning Toolkit*. Johns Hopkins University, http://nlp.cs.jhu.edu/˜rflorian/fntbl/documentation.html.

R. Florian. 2002a. Named entity recognition as a house of cards: Classifier stacking. In *Proceedings of CoNLL-2002*, pages 175–178.

R. Florian. 2002b. *Transformation Based Learning and Data-Driven Lexical Disambiguation: Syntactic and Semantic Ambiguity Resolution*. Ph.D. thesis, Johns Hopkins University. Chapter 5.3, pages 135–142.

Abraham Ittycheriah, Martin Franz, and Salim Roukos. 2001. IBM's statistical question answering system – trec-10. *TREC-10 Proceedings*, pages 258–264.

F. Jelinek. 1997. *Statistical Methods for Speech Recognition*. MIT Press.

E. F. Tjong Kim Sang, W. Daelemans, H. Dejean, R. Koeling, Y. Krymolowsky, V. Punyakanok, and D. Roth. 2000. Applying system combination to base noun phrase identification. In *Proceedings of COLING 2000*, pages 857–863.

H. van Halteren, J. Zavrel, and W. Daelemans. 2001. Improving accuracy in word class tagging through the combination fo machine learning systems. *Computational Linguistics*, 27(2):199–230.

A. J. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13:260–267.

T. Zhang and D. Johnson. 2003. A robust risk minimization based named entity recognition system. In *Proceedings of CoNLL-2003*.

T. Zhang, F. Damerau, and D. Johnson. 2002. Text chunking based on a generalization of winnow. *Journal of Machine Learning Research*, 2:615–637, March.