HowtogetaChineseName(Entity): Segmentation and Combination Issues

Hongyan Jing **Radu Florian** **Xiaoqiang Luo**
Tong Zhang **Abraham Ittycheriah**
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
{hjing,raduf,xiaoluo,tzhang,abei}@us.ibm.com

Abstract

When building a Chinese named entity recognition system, one must deal with certain language-specific issues such as whether the model should be based on characters or words. While there is no unique answer to this question, we discuss in detail advantages and disadvantages of each model, identify problems in segmentation and suggest possible solutions, presenting our observations, analysis, and experimental results. The second topic of this paper is classifier combination. We present and describe four classifiers for Chinese named entity recognition and describe various methods for combining their outputs. The results demonstrate that classifier combination is an effective technique of improving system performance: experiments over a large annotated corpus of fine-grained entity types exhibit a 10% relative reduction in F-measure error.

1 Introduction

Named entity (NE) recognition has drawn much attention in recent years. It was a designated task in a number of conferences, including the Message Understanding Conferences (MUC-6, 1995; MUC-7, 1997), the Information Retrieval and Extraction Conference (IREX, 1999), the Conferences on Natural Language Learning (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003), and the recent Automatic Content Extraction Conference (ACE, 2002).

A variety of algorithms have been proposed for NE recognition. Many of these algorithms are, in principle, language-independent. However, when applying these algorithms to languages such as Chinese and Japanese, we must deal with certain language-specific issues: for example, should we build a character-based model or a word-based model? how do word segmentation errors affect NE recognition? how should word segmentation and NE recognition interact with each other? Besides word segmentation related issues, Chinese does not have capitalization, which is a very useful feature in identifying NEs in languages such as English, Spanish, or Dutch. How does the lack of features such as capitalization affect the performance?

In the first part of this paper, we discuss these language-specific issues in Chinese NE recognition. In particular, we use a hidden Markov model (HMM) system as an example, and discuss various issues related to applying the HMM classifier to Chinese. The HMM classifier is similar to the one described in (Bikel et al., 1999).

In the second part of this paper, we investigate the combination of a set of diverse NE recognition classifiers. Four statistical classifiers are combined in the experiments, including the above-mentioned hidden Markov model classifier, a transformation-based learning classifier (Brill, 1995; Florian and Ngai, 2001), a maximum entropy classifier (Ratnaparkhi, 1999), and a robust risk minimization classifier (Zhang et al., 2002).

The remainder of this paper is organized as follows: Section 2 describes the experiment data, Section 3 discusses specific issues related to Chinese NE recognition, Section 4 presents the four classifiers and approaches to combining these classifiers.

2 Data

We used three annotated Chinese corpora in our experiments.

The IBM-FBIS Corpus

The Foreign Broadcast Information Service (FBIS) offers an extensive collection of translations and transcriptions of open source information monitored worldwide on diverse topics such as military affairs, politics, economics, and science and technology. The IBM-FBIS corpus consists of approximately 3,000 Chinese articles obtained from FBIS (about 3.2 million Chinese characters in total). This corpus was tagged by a native Chinese speaker with 32 NE categories, such as *person, location, organization, country, people, date, time, percentage, cardinal, ordinal, product, substance,* and *salutation*. There are approximately 300,000 NEs in the entire corpus, 16% of which are labeled as *person*, 16% as *organization*, and 11% as *location*.

The IBM-CT Corpus

The Chinese Treebank (Xia et al., 2000), available from Linguistic Data Consortium, consists of a 100,000 word (approximately 160,000 characters) corpus annotated with word segmentation, part-of-speech tags, and syntactic bracketing. It includes 325 articles from Xinhua newswire between 1994 and 1998. The same Chinese annotator who worked on the above IBM-FBIS data also annotated the Chinese Treebank data with NE information, henceforth the IBM-CT corpus, using the same 32 categories as mentioned above.

The IEER data

The National Institute of Standard and Technology organized the Information Extraction – Entity Recognition (IEER) evaluation, which involves entity recognition from textual information sources in both English and Mandarin. The Mandarin training data consists of approximately 10 hours of broadcast news transcripts comprised of approximately 390 stories. The test data also contains transcripts of broadcast news[1]. The training data includes approximately 170,000 characters and the test data includes approximately 6,500 characters. Ten categories of NEs were annotated, such as *person, location, organization, date, duration,* and *measure*.

[1] Other types of test data were also used in IEER evaluation, including newswire text and real automatic speech recognition transcripts, but we did not use them in our experiments.

3 Language-Specific Issues in Chinese NE Recognition

Chinese does not have delimiters between words, so a key design issue in Chinese NE recognition is whether to build a character-based model or a word-based model. In this section, we use a hidden Markov model NE recognition system as an example to discuss language-specific issues in Chinese NE recognition.

3.1 The Hidden Markov Model Classifier

NE recognition can be formulated as a classification task, where the goal is to label each token with a tag indicating whether it belongs to a specific NE or is not part of any NE. The HMM classifier used in our experiments follows the algorithm described in (Bikel et al., 1999). It performs sequence classification by assigning each token either one of the NE types or the label “O” to represent “outside any NE”. The states in the HMM are organized into regions, one region for each type of NE plus one for “O”. Within each of the regions, a statistical language model is used to compute the likelihood of words occurring within that region. The transition probabilities are smoothed by deleted interpolation, and the decoding is performed using the Viterbi algorithm.

3.2 Character-Based, Word-Based, and Class-Based Models

To build a model for identifying Chinese NEs, we need to determine the basic unit of the model: character or word. On one hand, the word-based model is attractive since it allows the system to inspect a larger window of text, which may lead to more informative decisions. On the other hand, a word segmenter is not error-prone and these errors may propagate and result in errors in NE recognition.

Two systems, a character-based HMM model and a word-based HMM model, were built for comparison. The word segmenter used in our experiments relies on dictionaries and surrounding words in local context to determine the word boundaries. During training, the NE boundaries were provided to the word segmenter; the latter is restricted to enforce word boundaries at each entity boundary. Therefore, at training time, the word boundaries are consistent with the entity boundaries. At test time, however, the segmenter could create words which do not agree with the gold-standard entity boundaries.

Corpus	Model	Prec	Rec	$F_{\beta=1}$
IBM-FBIS	Character	74.36%	80.24%	77.19
	Word	72.46%	75.97%	74.17
	Class	72.74%	76.20%	74.43
IEER	Character	74.57%	78.01%	76.25
	Word	77.51%	65.22%	70.83
	Class	77.21%	64.36%	70.20

Table 1: Performance of the character-based HMM model, the word-based HMM model, and the class-based HMM model. *(The precision, recall, and F-measure presented in this table and throughout this paper are based on correct identification of all the attributes of an NE, including boundary, content, and type.)*

The performance of the character-based model and the word-based model are shown in Table 1. The two corpora used in the evaluation, the IBM-FBIS corpus and the IEER corpus, differ greatly in data size and the number of NE types. The IBM-FBIS training data consists of 3.1 million characters and the corresponding test data has 270,000 characters. As we can see from the table, for both corpora, the character-based model outperforms the word-based model, with a lead of 3 to 5.5 in F-measure. The performance gap between two models is larger for the IEER data than for the IBM-FBIS data.

We also built a class-based NE model. After word segmentation, class tags such as *number, chinese-name, foreign-name, date,* and *percent* are used to replace words belonging to these classes. Whether a word belongs to a specific class is identified by a rule-based normalizer. The performance of the class-based HMM model is also shown in Table 1. For the IBM-FBIS corpus, the class-based model outperforms the word-based model; for the IEER corpus, the class-based model is worse than the word-based model. In both cases, the performance difference between the word-based model and the class-based model is very small. The character-based model outperforms the class-based model in both tests.

A more careful analysis indicates that although the word-based model performs worse than the character-based model overall in our evaluation, it performs better for certain NE types. For instance, the word-based model has a better performance for the *organization* category than the character-based model in both tests. While the character-based model has an F-measure of 65.07 (IBM-FBIS) and 64.76 (IEER) for the *organization* category, the word-based model achieves F-measure scores of 69.14 (IBM-FBIS) and 72.38 (IEER) respectively. One reason may be that organization names tend to contain many characters, and since the word-based model allows the system to analyze a larger window of text, it is more likely to make a correct guess. We can integrate the character-based model and the word-based model by combining the decisions from the two models. For instance, if we use the decisions of the word-based model for the *organization* category, but use the decisions of the character-based model for all the other categories, the overall F-measure goes up to 76.91 for the IEER data, higher than using either the character-based or word-based model alone. Another way to integrate the two models is to use a hybrid model – starting with a word-based model and backing off to character-based model if the word is unknown.

3.3 Granularity of Word Segmentation

We believe that one main reason for the lower performance of the word-based model is that the word granularity defined by the word segmenter is not suitable for the HMM model to perform the NE recognition task. What exactly constitutes a Chinese word has been a topic of major debate. We are interested in what is the best word granularity for our particular task.

To illustrate the word granularity problem for NE tagging, we take person names as an example. Our word segmenter marks a person's name as one word, consistent with the convention used by the Chinese treebank and many other word segmentation systems. While this may be useful in other applications, it is certainly not a good choice for our NE model. Chinese names typically contain two or three characters, with family name preceding first name. Only a limited set of characters are used as family names, while the first name can be any character(s). Therefore, the family name is a very important and useful feature in identifying an NE in the *person* category. By combining the family name and the first name into one word, this important feature is lost to the word-based model. In our tests, the word-based model performs much worse for the *person* category than the character-based model. We believe that, for the purpose of NE recognition, it is better to separate the family name from the first name in word segmentation, although this is not the convention used in the Chinese treebank.

Other examples include the segmentation of words indicating dates, countries, locations, percentages, measures, and ordinals. For instance, "July 4th" is expressed by four characters *"7th month 4th day"* in Chinese. The word segmenter marks the four characters as a single word; however, the second and the last character are actually good features for indicating date, since the dates are usually expressed using the same structure (e.g., "March 25th" is expressed by *"3rd month 25th day"* in Chinese). For reasons similar to the above, we believe that it is better to separate characters representing "month" and "day", rather than combining the four characters into one word. A similar problem can be observed in English with tokens such as "61-year-old man" if one is interested in identifying a person's age, in which case 'year' and 'old' are good features for predication.

The above analysis suggests that a better way to apply a word segmenter in an NE system is to first adapt the segmenter so that the segmentation granularity is more appropriate to the particular task and model. As a guideline, characters that are good features for identifying NEs should not be combined with other characters into word. Additional examples include characters expressing "percent" and characters representing monetary measures .

3.4 The Effect of Segmentation Errors

Word segmentation errors can lead to mistakes in NE recognition. Suppose an NE consists of four characters (C_1, C_2, C_3, C_4), if the word segmentation merges C_1 with a character preceding it, then this NE cannot be correctly identified by the word-based model since the boundary will be incorrect. Besides inducing NE boundary errors, incorrect word segmentation also leads to wrong matchings between training examples and testing examples, which may result in mistakes in identifying entities.

We computed the upper bound for the word-based model for the IBM-FBIS test presented in Table 1. The upper bound of performance is computed by dividing the total number of NEs whose boundaries are also recognized as boundaries by the word segmenter by the total number of NEs in the corpus, which is the precision, recall, and also the F-measure. For the IBM-FBIS test data in Table 1, the upper bound of the word-based model is 95.7 F-measure.

We also did the following experiment to measure the effect of word segmentation errors: we gave the boundaries of NEs in the test data to the word segmenter and forced it to mark entity boundaries as word boundaries. This eliminates the word segmentation errors that inevitably result in NE boundary errors. For the IBM-FBIS data, the word-based HMM model achieves 76.60 F-measure when the entity boundaries in the test data are given, and the class-based model achieves 77.77 F-measure, higher than the 77.19 F-measure by the character-based model in Table 1. For the IEER data, the F-measure of the word-based model improves from 70.83 to 73.74 when the entity boundaries are given, and the class-based model improves from 70.20 to 72.47.

This suggests that with the improvement in Chinese word segmentation, the word-based model may achieve comparable or better performance than the character-based model.

3.5 Lexical Features

Capitalization in English gives good evidence of names. Our HMM classifier for English uses a set of word-features to indicate whether a word contains all capitalized letters, only digits, or capitalized letters and period, as described in (Bikel et al., 1999). However, Chinese does not have capitalization. When we applied the HMM system to Chinese, we retained such features since Chinese text also include digits and roman words (such as in product or company names). In an attempt to investigate the usefulness of such features for Chinese, we removed them from the system and observed very little difference in overall performance (0.4 difference in F-measure).

3.6 Sensitivity to Corpus and Training Size Variation

To test the robustness of the model, we trained the system on the 100,000 word IBM-CT data and tested on the same IBM-FBIS data. The character-based model achieves 61.36 F-measure and the word-based model achieves 58.40 F-measure, compared to 77.19 and 74.17, respectively, using the 20 times larger IBM-FBIS training set. This represents an approximately 20% relative reduction in performance when trained on a related yet different and considerably smaller training set. We plan to investigate further the relation between corpus type and size and performance.

4 Classifier Combination

This section investigates the combination of a set of classifiers for NE recognition. We first introduce the classifiers used in our experiments and then describe the combination methods.

4.1 The Classifiers

Besides the HMM classifier mentioned in the previous section, the following three classifiers were used in the experiments.

4.1.1 The Transformation-Based Learning (fnTBL) 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.

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 (similar to Snow (Dagan et al., 1997), Maximum Entropy, decision trees, etc); it is error–driven, thus directly minimizes the ultimate evaluation measure: the error rate. The TBL toolkit used in this experiment is described in (Florian and Ngai, 2001).

4.1.2 The Maximum Entropy Classifier (MaxEnt)

The model used here is based on the maximum entropy model used for shallow parsing (Ratnaparkhi, 1999). A sentence with NE tags is converted into a shallow tree: tokens not in any NE are assigned an "O" tag, while tokens within an NE are represented as constituents whose label is the same as the NE type. For example, the annotated sentence "I will fly to (LOCATION New York) (DATEREF tomorrow)" is represented as a tree "(S I/O will/O fly/O to/O (LOCATION New/LOCATION York/LOCATION) (DATEREF tomorrow/DATEREF))". Once an NE is represented as a shallow tree, NE recognition can be realized by performing shallow parsing.

We use the tagging and chunking model described in (Ratnaparkhi, 1999) for shallow parsing. In the tagging model, the context consists of a window of five tokens (including the token being tagged and two tokens to its left and two tokens to its right) and two tags to the left of the current token. Five groups of feature templates are used: token unigram, token bigram, token trigram, tag unigram and tag bigram (all within the context window). In the chunking model, the context is limited to a window of three subtrees: the previous, current and next subtree. Unigram and bigram chunk (or tag) labels are used as features.

4.1.3 The Robust Risk Minimization (RRM) Classifier

This system is a variant of the text chunking system described in Zhang et al. (2002), where the NE recognition problem is regarded as a sequential token-based tagging problem. We denote by $\{w_i\}$ $(i = 0, 1, \ldots, m)$ the sequence of tokenized text, which is the input to our system. In token-based tagging, the goal is to assign a class-label t_i to every token w_i.

In our system, this is achieved by estimating the conditional probability $P(t_i = c|x_i)$ for every possible class-label value c, where x_i is a feature vector associated with token i. The feature vector x_i can depend on previously predicted class-labels $\{t_j\}_{j \leq i}$, but the dependency is typically assumed to be local. Given such a conditional probability model, in the decoding stage, we estimate the best possible sequence of t_i's using a dynamic programming approach.

In our system, the conditional probability model has the following parametric form:

$$P(t_i = c|x_i, \{t_{i-\ell}, \ldots, t_{i-1}\}) = T(w_c^T x_i + b_c),$$

where $T(y) = \min(1, \max(0, y))$ is the truncation of y into the interval $[0, 1]$. w_c is a linear weight vector and b_c is a constant. Parameters w_c and b_c can be estimated from the training data.

This classification method is based on approximately minimizing the risk function. The generalized Winnow method used in (Zhang et al., 2002) implements such a *robust risk minimization* method.

4.1.4 Performance of the Classifiers

We compared the performance of the four classifiers by training and testing them on the same data sets. We divided the IBM-FBIS corpus into three subsets: 2.8 million characters for training, 330,000 characters for development testing, and 330,000 characters for testing. Table 2 shows the results of each classifier for the development test set and the evaluation set. The RRM and fnTBL classifiers are the best performers for the test set, followed by MaxEnt. The HMM classifier lags behind by around 6

	Development Test			Test		
	Precision	Recall	F-measure	Precision	Recall	F-measure
Baseline1	17.52%	24.92%	20.58	14.04%	20.52%	16.67
Baseline2	77.86 %	45.26%	57.24	71.06%	40.22%	51.37
HMM	71.38%	80.92%	75.85	71.73%	77.55%	74.53
MaxEnt	83.82%	78.08%	80.85	85.43%	75.22%	80.00
fnTBL	76.85%	81.55%	80.08	82.06%	80.68%	81.36
RRM	82.83%	81.06%	81.89	84.53%	78.64%	81.48

Table 2: Baselines and performance of the four classifiers.

points in F-measure from the best system. The presented results are for character-based models.

For comparison, we also computed two baselines: one in which each character is labeled with its most frequent label (Baseline1 in Table 2), and one in which each entity that was seen in training data is labeled with its most frequent classification (Baseline2 in Table 2 - this baseline is computed using the software provided with the CoNLL-2003 shared task (Tjong Kim Sang and De Meulder, 2003)).

4.2 Combination

The four classifiers differ in multiple dimensions, making them good candidates for combination. We explored various ways to combine the results from the four classifiers.

4.2.1 Combination algorithms

For the first part of the experimental setup, we consider the following classification framework: given the (probabilistic) output $Pr_i\left(\cdot|w\right)$ of n classifiers $\mathcal{C}_1, \ldots, \mathcal{C}_n$, the classifier combination problem can be viewed a probability interpolation problem – compute the class probability distribution conditioned on the joint classifier output:

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

where C_i is the i$^{\text{th}}$ classifier's output, w is an observable context (e.g., a word trigram) and f is a combination function. A commonly used combining scheme is through linear interpolation of the classifiers' class probability distributions:

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

The weights $\lambda_i\left(w\right)$ encode the importance given to classifier i in combination, for the context 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 context w is C_i. These parameters from Equation (2) can be estimated, if needed, on development data.

Table 3 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 $P_i\left(C|w, C_i\right) = \delta\left(C, C_i\right)$, δ being the Kronecker operator:

$$\delta\left(x, y\right) = \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases}$$

In other words, 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 (i.e., it is selected as the classification output). However, this procedure may lead to ties, where some classifications receive an identical number of votes – one usually resorts to randomly selecting one of the tied candidates in this case – Table 3 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 $\lambda_i\left(w\right) = 1 - P_i\left(error\right)$. In this method, presented in Table 3 as *weighted voting*, better performing classifiers will have a higher impact on the final classification.

In the previously described methods, also known as voting, each classifier gave its entire vote to one classification – 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 the experiments described here, this value is computed directly on the development data. However, the space of possible choices for C, w and C_i is large enough to make the estimation

	Precision	Recall	F-measure
Best Classifier	84.53	78.64	81.48
Equal weight voting	85.2±0.03	81.7±0.02	83.3±0.02
Weighted voting	86.04	80.63	83.24
Model 1	83.07	82.10	82.58
Model 2	86.3	77.55	81.69
RRM	89.01	79.61	84.05
RRM+ flags	88.96	79.84	84.18
RRM+IOB1+IOB2	88.5	80.46	**84.29**
Oracle	*91.6*	*92.3*	*91.95*

Table 3: Classifier combination results.

unreliable, so we use two approximations, named *Model 1* and *Model 2* in Table 3: $P_i\left(C|w,C_i\right) = P_i\left(C|w_0\right)^2$ and $P_i\left(C|w,C_i\right) = P_i\left(C|C_i\right)$, respectively. Both probability distributions are estimated as smoothed relative frequencies on the development data. Interestingly, both methods underperform the equal voting method, a fact which can be explained by inspecting the results in Table 2: the fnTBL method has an accuracy (computed on development data) lower than the MaxEnt accuracy, but it outperforms the latter on the test data. Since the parameters $P\left(C|w,C_i\right)$ are computed on the development data, they are probably favoring the MaxEnt method, resulting in lower performance. On the other hand, the equal voting method does not suffer from this problem, as its parameters are not dependent on the development data. In a last set of experiments, we extend the classification framework to a larger space, in which we compute the conditional class probability distribution conditioned on an arbitrary set of features:

$$P\left(C|f_1^k\right) = f\left(\left\{P_i\left(C|w,f_1^k\right)\right\}_{i=1\ldots n}\right) \quad (3)$$

This setup allows us to still use the classifications of individual systems as features, but also allows for other types of conditioning features – for instance, one can use the output of any classifier (e.g., POS tags, text chunk labels, etc) as features.

In the described experiments, we use the RRM method to compute the function f in Equation (3), 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 development data, and, in subsequent experiments, the system was also fed a *flag* stream which briefly identifies some of the tokens (numbers, romanized characters, etc) and the output of each system in a different NE encoding scheme.

In all the voting experiments, the NEs were encoded in an IOB1 scheme, since it seems to be the most amenable to combination. Briefly, the IOB general encoding scheme associates a label with each word, corresponding to whether it begins a specific entity, continues the entity, or is outside any entity. Tjong Kim Sang and Veenstra (1999) describes in detail the IOB schemes. The final experiment also has access to the output of systems trained in the IOB2 encoding. The addition of each feature type resulted in better performance, with the final result yielding a 10% relative decrease of F-measure error when compared with the best performing system[3]. Table 3 also includes an upper-bound on the classifier combination performance - the performance of the *switch* oracle, which selects the correct classification if at least one classifier outputs it.

Table 3 shows that, at least for the examined types of combination, using a robust feature-based classifier to compute the classification distribution yields better performance than combining the classifications through either voting or weighted interpolation. The RRM-based classifier is able to incorporate heterogenous information from multiple sources, obtaining a 2.8 absolute F-measure improvement versus the best performing classifier and 1.0 F-measure gain over the next best method.

5 Related Work

Sun et al. (2002) proposes to use a class-based model for Chinese NE recognition. Specifically, it uses a character-based trigram model for the class *person*, a word-based model for the class *location*, and a more complicated model for the class *organization*. This decision is consistent with our observation that the character-based model performs better than the word-based model for classes such as *person*, but is worse for classes such as *organization*.

Sekine and Eriguchi (2000) provides an overview of Japanese NE recognition. It presents the results of 15 systems that participated in an evaluation project for Information Retrieval and Information Extraction (IREX, 1999). Utsuro et al. (2002) studies combining outputs of multiple Japanese NE systems by stacking. A second stage classifier – in this case, a decision list – is trained to combine the outputs from first stage classifiers. This is similar

[2] w_0 is the word associated with the context w.

[3] Measured as $F_e = 1 - F$.

in spirit to our application of the RRM classifier for combining classifier outputs.

Classifier combination has been shown to be effective in improving the performance of NLP applications, and have been investigated by Brill and Wu (1998) and van Halteren et al. (2001) for part-of-speech tagging, Tjong Kim Sang et al. (2000) for base noun phrase chunking, and Florian et al. (2003a) for word sense disambiguation. Among the investigated techniques were voting, probability interpolation, and classifier stacking. We also applied the classifier combination technique discussed in this paper to English and German (Florian et al., 2003b).

6 Conclusion

In this paper, we discuss two topics related to Chinese NE recognition: dealing with language-specific issues such as word segmentation, and combining multiple classifiers to enhance the system performance. In the described experiments, the character-based model consistently outperforms the word-based model – one major reason for this fact is that the segmentation granularity might not be suited for this particular task. Combining four statistical classifiers, including a hidden Markov model classifier, a transformation-based learning classifier, a maximum entropy classifier, and a robust risk minimization classifier, significantly improves the system performance, yielding a 10% relative reduction in F-measure error over the best performing classifier.

Acknowledgments

We would like to specially thank Fei Xia for helpful discussions and for providing the Chinese word segmenter. We also thank Weijing Zhu for his assistance in data preparation, and Nanda Kambhatla, Todd Ward and Salim Roukos for their support and suggestions.

This work was partially supported by the Defense Advanced Research Projects Agency and monitored by SPAWAR under contract No. N66001-99-2-8916 and by the Knowledge Discovery and Dissemination Project sponsored by the National Science Foundation. The views and findings contained in this material are those of the authors and do not necessarily reflect the position of policy of the Government and no official endorsement should be inferred.

References

ACE. 2002. Automatic content extraction. http://www.ldc.upenn.edu/Projects/ACE/.

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

E. Brill and J. Wu. 1998. Classifier combination for improved lexical disambiguation. *Proceedings of COLING-ACL'98*, pages 191–195, August.

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.

I. Dagan, Y. Karov, and D. Roth. 1997. Mistake-driven learning in text categorization. In *Proceedings of EMNLP-1997*.

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, S. Cucerzan, C. Schafer, and D. Yarowsky. 2003a. Combining classifiers for word sense disambiguation. *Journal of Natural Language Engineering*, 8(4).

R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. 2003b. Named entity recognition through classifier combination. In *Proceedings of CoNLL-2003*. Edmonton, Canada.

IREX. 1999. Information retrieval and extraction exercise. http://nlp.cs.nyu.edu/irex/index-e.html.

MUC-6. 1995. The sixth message understanding conference. http://www.cs.nyu.edu/cs/faculty/grishman/muc6.html.

MUC-7. 1997. The seventh message understanding conference. http://www.itl.nist.gov/iad/894.02/related_projects/muc/proceedings/muc_7_toc.html.

A. Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34:151–178.

S. Sekine and Y. Eriguchi. 2000. Japanese named entity extraction evaluation - analysis of results. In *Proceedings of COLING-2000*, Saarbrüecken, Germany.

J. Sun, J. Gao, L. Zhang, M. Zhou, and C. Huang. 2002. Chinese named entity identification using class-based language model. In *Proceedings of COLING-2002*, Taiwan.

E. F. Tjong Kim Sang and F. De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*.

E. F. Tjong Kim Sang and J. Veenstra. 1999. Representing text chunks. In *Proceedings of EACL'99*.

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.

E. F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*. Taiwan.

T. Utsuro, M. Sassano, and K. Uchimoto. 2002. Combining outputs of multiple Japanese named entity chunkers by stacking. In *Proceedings of EMNLP-2002*, Pennsylvania.

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.

F. Xia, M. Palmer, N. Xue, M..E. Okurowski, J. Kovarik, S. Huang, T. Kroch, and M. Marcus. 2000. Developing guidelines and ensuring consistency for Chinese text annotation. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, Athens, Greece.

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