Japanese Named Entity Recognition based on a Simple Rule Generator and Decision Tree Learning

Hideki Isozaki
NTT Communication Science Laboratories
2-4 Hikaridai, Seika-cho, Souraku-gun, Kyoto
619-0237, Japan
isozaki@cslab.kecl.ntt.co.jp

Abstract

Named entity (NE) recognition is a task in which proper nouns and numerical information in a document are detected and classified into categories such as person, organization, location, and date. NE recognition plays an essential role in information extraction systems and question answering systems. It is well known that hand-crafted systems with a large set of heuristic rules are difficult to maintain, and corpus-based statistical approaches are expected to be more robust and require less human intervention. Several statistical approaches have been reported in the literature. In a recent Japanese NE workshop, a maximum entropy (ME) system outperformed decision tree systems and most hand-crafted systems. Here, we propose an alternative method based on a *simple rule generator* and *decision tree learning*. Our experiments show that its performance is comparable to the ME approach. We also found that it can be trained more efficiently with a large set of training data and that it improves readability.

1 Introduction

Named entity (NE) recognition is a task in which proper nouns and numerical information in a document are detected and classified into categories such as person, organization, location, and date. NE recognition plays an essential role in information extraction systems (see MUC documents (1996)) and question answering systems (see TREC-QA documents, http://trec.nist.gov/). When you want to know the location of the Taj Mahal, traditional IR techniques direct you to relevant documents but do not directly answer your question. NE recognition is essential for finding possible answers from documents. Although it is easy to build an NE recognition system with mediocre performance, it is difficult to make it reliable because of the large number of ambiguous cases. For instance, we cannot determine whether "Washington" is a person's name or a location's name without the necessary context.

There are two major approaches to building NE recognition systems. The first approach employs hand-crafted rules. It is well known that hand-crafted systems are difficult to maintain because it is not easy to predict the effect of a small change in a rule. The second approach employs a statistical method, which is expected to be more robust and to require less human intervention. Several statistical methods have been reported in the literature (Bikel et al., 1999; Borthwick, 1999; Sekine et al., 1998; Sassano and Utsuro, 2000).

IREX (Information Retrieval and Extraction Exercise, (Sekine and Eriguchi, 2000; IRE, 1999)) was held in 1999, and fifteen systems participated in the formal run of the Japanese NE excercise. In the formal run, participants were requested to tag two data sets (GENERAL and ARREST), and their scores were compared in terms

of F-measure, i.e., the harmonic mean of 'recall' and 'precision' defined as follows.

- recall = x/(the number of correct NEs)
- precision = x/(the number of NEs extracted by the system)

  where x is the number of NEs correctly extracted and classified by the system.

GENERAL was the larger test set, and its best system was a hand-crafted one that attained F=83.86%. The second best system (F=80.05%) was also hand-crafted but enhanced with transformation-based error-driven learning. The third best system (F=77.37%) was Borthwick's ME system enhanced with hand-crafted rules and dictionaries (1999). Thus, the best three systems used quite different approaches.

In this paper, we propose an alternative approach based on a *simple rule generator* and *decision tree learning* (RG+DT). Our experiments show that its performance is comparable to the ME method, and we found that it can be trained more efficiently with a large set of training data. By adding in-house data, the proposed system's performance was improved by several points, while a standard ME toolkit crashed.

When we try to extract NEs in Japanese, we encounter several problems that are not serious in English. It is relatively easy to detect English NEs because of capitalization. In Japanese, there is no such useful hint. Proper nouns and common nouns look very similar. In English, it is also easy to tokenize a sentence because of inter-word spacing. In Japanese, inter-word spacing is rarely used. We can use an off-the-shelf morphological analyzer for tokenization, but its word boundaries may differ from the corresponding NE boundaries in the training data. For instance, a morphological analyzer may divide a four-character expression `OO-SAKA-SHI-NAI` into two words `OO-SAKA` (= Osaka) and `SHI-NAI` (= in the city), but the training data would be tagged as `<LOCATION>OO-SAKA-SHI</LOCATION>NAI` (= in `<LOCATION>`Osaka City `</LOCATION>`). Moreover, *unknown words are often divided excessively or incorrectly* because an analyzer tries to interpret a sentence as a sequence of known words.

Throughout this paper, the typewriter-style font is used for Japanese, and hyphens indicate character boundaries. Different types of characters are used in Japanese: hiragana, katakana, kanji, symbols, numbers, and letters of the Roman alphabet. We use 17 character types for words, e.g., `single-kanji`, `all-kanji`, `all-katakana`, `all-uppercase`, `float` (for floating point numbers), `small-integer` (up to 4 digits).

2 Methodology

Our RG+DT system (Fig. 1) generates a *recognition rule* from each NE in the training data. Then, the rule is refined by decision tree learning. By applying the refined recognition rules to a new document, we get NE candidates. Then, non-overlapping candidates are selected by a kind of longest match method.

2.1 Generation of recognition rules

In our method, each tokenized NE is converted to a recognition rule that is essentially a sequence of part-of-speech (POS) tags in the NE. For instance, `OO-SAKA-GIN-KOU` (= Osaka Bank) is tokenized into two words: `OO-SAKA:all-kanji:location-name` (= Osaka) and `GIN-KOU:all-kanji:common-noun` (= Bank), where `location-name` and `common-noun` are POS tags. In this case, we get the following recognition rule. Here, '`*`' matches anything.

```
*:*:location-name,
   *:*:common-noun
 -> ORGANIZATION
```

However, this rule is not very good. For instance, `OO-SAKA-WAN` (= Osaka Bay) follows this pattern, but it is a location's name. `GIN-KOU` and `WAN` strongly imply `ORGANIZATION` and `LOCATION`, respectively. Thus, the last word of an NE is often a head that is more useful than other words for the classification. Therefore, we register the last word into a *suffix dictionary* for each non-numerical NE class (i.e., `ORGANIZATION`, `PERSON`, `LOCATION`, and `ARTIFACT`) in order to accept only reliable candidates. If the last word appears in two or more different NE, we call it a *reliable NE suffix*. We register only reliable ones.

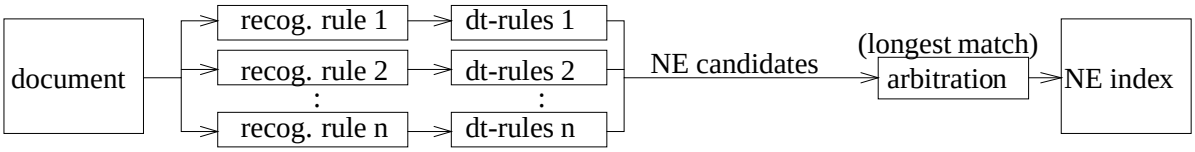

Figure 1: Rough sketch of RG+DT system

In the above examples, the last words were common nouns. However, the last word can also be a proper noun. For instance, we will get the following rule from `<ORGANIZATION>OO-SAKA-TO-YO-TA</ORGANIZATION>` (= Osaka Toyota) because Japanese POS taggers know that `TO-YO-TA` is an organization name (a kind of proper noun).

```
*:*:location-name, *:*:org-name
  -> ORGANIZATION,0,0
```

Since Yokohama Honda and Kyoto Sony also follow this pattern, the second element `*:*:org-name` should not be restricted to the words in the training data. Therefore, we do not restrict proper nouns by a suffix dictionary, and we do not restrict numbers either.

In addition, the first or last word of an NE may contain an NE boundary as we described before (`SHI</LOCATION>NAI`). In this case, we can get `OO-SAKA-SHI` by removing no character of the first word `OO-SAKA` and one character of the last word `SHI-NAI`. Accordingly, this modification can be represented by two integers: `0,1`.

Furthermore, one-word NEs are different from other NEs in the following respects.

- The word is usually a proper noun, an unknown word, or a number; otherwise, it is an exceptional case.
- The character type of a one-word NE gives a useful hint for its classification. For instance, `all-uppercase` words (e.g., IOC) are often classified as `ORGANIZATION`.

Since unknown words are often proper nouns, we assume they are tagged as `misc-proper-noun`. If the training data contains `<ORGANIZATION>I-O-C</ORGANIZATION>` and `I-O-C` (= IOC) is an unknown word, we will get `I-O-C:all-uppercase:misc-proper-noun`.

By considering these facts, we modify the above rule generation. That is, we replace every word in an NE and its character type by '`*`' to get the left-hand side of the corresponding recognition rule *except the following cases*.

A word that contains an NE boundary If the first or last word of the NE contains an *NE boundary* (e.g, `SHI</LOCATION>NAI`), the word is not replaced by '`*`'. The number of characters to be deleted is also recorded in the right-hand side of the recognition rule.

One-word NE The following exceptions are applied to one-word NEs. If the word is a proper noun or a number, its character type is not replaced by '`*`'. Otherwise, the word is not replaced by '`*`'.

The last word of a longer NE The following exceptions are applied to the last word of a non-numerical NE that is composed of two or more words when the word is neither a proper noun nor a number. If the last word is a reliable NE suffix (i.e., it appears in two or more different NEs in the class), its information (i.e., the last word, its character type, and its POS tag) is registered into a suffix dictionary for the NE class. The last word of the recognition rule must be an element of the suffix dictionary. Unreliable NE suffixes are not replaced by '`*`'. Suffixes of numerical NEs (i.e., `DATE`, `TIME`, `MONEY`, `PERCENT`) are not replaced, either.

Now, we obtain the following recognition rules from the above examples.

```
*:all-uppercase:misc-proper-noun
  -> ORGANIZATION,0,0.
*:*:location-name,
    SHI-NAI:*:common-noun
  -> LOCATION,0,1.
```

```
*:*:location-name,
   *:*:common-noun
 -> ORGANIZATION,0,0.
```

The first rule extracts CNN as an organization. The second rule extracts YOKO-HAMA-SHI (= Yokohama City) from YOKO-HAMA-SHI-NAI (= in Yokohama City). The third rule extracts YOKO-HAMA-GIN-KOU (= Yokohama Bank) as an organization. Note that, in this rule, the second element (*:*:common-noun) is constrained by the suffix dictionary for ORGANIZATION because it is neither a proper noun nor a number. Hence, the rule does not match YOKO-HAMA-WAN (= Yokohama Bay). If the suffix dictionary also happens to have KOU-KOU:all-kanji:commmon-noun (= senior high school), the rule also matches YOKO-HAMA-KOU-KOU (= Yokohama Senior High School).

IREX introduced <ARTIFACT> for product names, prizes, pacts, books, and fine arts, among other nouns. Titles of books and fine arts are often long and have atypical word patterns. However, they are often delimited by a pair of symbols that correspond to quotation marks in English. Some atypical organization names are also delimited by these symbols. In order to extract such a long NE, we concatenate all words within a pair of such symbols into one word. We employ the first and last word of the quoted words as extra features. In addition, we do not regard the quotation symbols as adjacent words because they are constant and lack semantic meaning.

When a large amount of training data is given, thousands of recognition rules are generated. For efficiency, we compile these recognition rules by using a hash table that converts a hash key into a list of relevant rules that have to be examined. We make this hash table as follows. If the left-hand side of a rule contains only one element, the element is used as a hash key and its rule identifier is appended to the corresponding rule list. If the left-hand side contains two or more elements, the first two elements are concatenated and used as a hash key and its rule identifier is appended to the corresponding rule list. After this compilation, we can efficiently apply all of the rules to a new document. By taking the first two elements into consideration, we can reduce the number of rules that need to be examined.

2.2 Refinement of recognition rules

Some recognition rules are not reliable. For instance, we get the following rule when a person's name is incorrectly tagged as a location's name by a POS tagger.

```
*:all-kanji:location-name
 -> PERSON,0,0
```

Therefore, we have to consider a way to refine the recognition rules.

By applying each recognition rule to the untagged training data, we can obtain NE candidates for the rule. By comparing the candidates with the given answer for the training data, we can classify them into positive examples and negative examples for the recognition rule. Consequently, we can apply decision tree learning to classify these examples correctly. We represent each example by a list of features: words in the NEs, h preceding words, k succeeding words, their character types, and their POS tags. If we consider one preceding word and two succeeding words, the feature list for a two-word named entity (w_0, w_1) will be w_{-1}, c_{-1}, p_{-1}, w_0, c_0, p_0, w_1, c_1, p_1, w_2, c_2, p_2, w_3, c_3, p_3, pn, where w_{-1} is the preceding word and w_2 and w_3 are the succeeding words. c_i is w_i's character type and p_i is w_i's POS tag. pn is a boolean value that indicates whether it is a positive example. If a feature value appears less than three times in the examples, it is replaced by a dummy constant. We also replace numbers by dummy constants because most numerical NEs follow typical patterns, and their specific values are often useless for NE recognition.

Here, we discuss handling short NEs. For example, NO-O-BE-RU-SHOU-SEN-KOU-I-IN-KAI (= the Nobel Prize Selection Committee) is an organization's name that contains a person's name NO-O-BE-RU (= Nobel) and an artifact name NO-O-BE-RU-SHOU (= Nobel Prize), but <PERSON>NO-O-BE-RU</PERSON> and <ARTIFACT>NO-O-BE-RU-SHOU</ARTIFACT> are incorrect in this case. If the training data contain NO-O-BE-RU as both positive and negative examples of a person's name, the decision tree learner will be confused. They are rejected because there is a longer named entity

and overlapping tags are not allowed. We do not have to change our knowledge that Nobel is a person's name. Therefore, we remove such negative examples caused by longer NEs. Consequently, the decision tree may fail to reject <PERSON> NO-O-BE-RU</PERSON>, but it will disappear in the final output because we use a longest match method for arbitration.

For readability, we translate each decision tree into a set of production rules by `c4.5rules` (Quinlan, 1993). Throughout this paper, we call them *dt-rules* (Fig. 1) in order to distinguish them from recognition rules. Thus, each recognition rule is enhanced by a set of dt-rules. The dt-rules removes unlikely candidates.

2.3 Arbitration of candidates

Once the refined rules are generated, we can apply them to a new document. This obtains a large number of NE candidates (Fig. 1). Since overlapping tags are not allowed, we use a kind of *left-to-right longest match method*. First, we compare their starting points and select the earliest ones. If two or more candidates start at the same point, their ending points are compared and the longest candidate is selected. Therefore, the candidates overlapping the selected candidate are removed from the candidate set. This procedure is repeated until the candidate set becomes empty.

The rank of a candidate starting at the x-th word boundary and ending at the y-th word boundary can be represented by a pair $(x, -y)$. The beginning of a sentence is the zeroth word boundary, and the first word ends at the first word boundary, etc. Then, the selected candidate should have the *minimum* rank according to the *lexicographical ordering* of $(x, -y)$. When a candidate starts or ends within a word (e.g., `SHI-NAI`), we assume that the entire word is a member of the candidate for the definition of $(x, -y)$.

According to this ordering, two candidates can have the same rank. One of them might assert that a certain word is an organization's name and another candidate might assert that it is a person's name. In order to apply the most frequently used rule, we extend this ordering by $(x, -y, -z_r)$, where z_r is the number of positive examples for the rule r.

2.4 Maximum entropy system

In order to compare our method with the ME approach, we also implement an ME system based on Ristad's toolkit (1997). Borthwick's (1999) and Uchimoto's (2000) ME systems are quite similar but differ in details. They regarded Japanese NE recognition as a classification problem of a word. The first word of a person name is classified as PERSON-BEGIN. The last word is classified as PERSON-END. Other words in the person's name (if any) are classified as PERSON-MIDDLE. If the person's name is composed of only one word, it is classified as PERSON-SINGLE. Similar labels are given to all other classes such as `LOCATION`. Non-NE words are classified as OTHER. Thus, every word is classified into 33 classes, i.e., $\{$ORGANIZATION, PERSON, LOCATION, ARTIFACT, DATE, TIME, MONEY, PERCENT$\}$ $\times$ $\{$BEGIN, MIDDLE, END, SINGLE$\}$ $\cup$ $\{$OTHER$\}$. For instance, the words in "President <PERSON> George Herbert Walker Bush </PERSON>" are classified as follows: President = OTHER, George = PERSON-BEGIN, Herbert = PERSON-MIDDLE, Walker = PERSON-MIDDLE, Bush = PERSON-END.

We use the following features for each word in the training data: the word itself, h preceding words, k succeeding words, their character types, and their POS tags. By following Uchimoto, we disregard words that appear fewer than five times and other features that appear fewer than three times.

Then, the ME-based classifier gives a probability for each class to each word in a new sentence. Finally, the *Viterbi* algorithm (see textbooks, e.g., (Allen, 1995)) enhanced with *consistency checking* (e.g., PERSON-END should follow PERSON-BEGIN or PERSON-MIDDLE) determines the best combination for the entire sentence.

We generate the word boundary rewriting rules as follows. First, the NE boundaries inside a word are assumed to be at the nearest word boundary outside the named entity. Hence, `SHI</LOCATION>NAI` is rewritten as `SHI-NAI</LOCATION>`. Accordingly, `SHI-NAI` is classified as LOCATION-END. The original NE boundary is recorded for the pair `SHI-NAI`/LOCATION-END, If `SHI-NAI`/LOCATION-END

is found in the output of the Viterbi algorithm, it is rewritten as SHI</LOCATION>NAI. Since rewriting rules from rare cases can be harmful, we employ a rewriting rule only when the rule correctly works for more than 50% of the word/class pairs in the training data.

3 Results

Now, we compare our method with the ME system. We used the standard IREX training data (CRL NE 1.4 MB and NERT 30 KB) and the formal run test data (GENERAL and AREST). When human annotators were not sure, they used <OPTIONAL POSSIBILITY=...> where POSSIBILITY is a list of possible NE classes. We also used 7.4 MB of in-house NE data that did not contain optional tags. All of the training data (all = CRL NE+NERT+in-house) were based on the Mainichi Newspaper's 1994 and 1995 CD-ROMs. Table 1 shows the details. We removed an optional tag when its possibility list contains NONE, which means this part is accepted without a tag. Otherwise, we selected the majority class in the list. As a result, 56 NEs were added to CRL NE.

For tokenization, we used chasen 2.2.1 (http://chasen.aist-nara.ac.jp/). It has about 90 POS tags and large proper noun dictionaries (persons = 32,167, organizations = 16,610, locations = 67,296, miscellaneous proper nouns = 26,106). (Large dictionaries sometimes make the extraction of NEs difficult. If OOSAKA-GIN-KOU is registered as a single word, GIN-KOU is not extracted as an organization suffix from this example.) We tuned chasen's parameters for NE recognition. In order to avoid the excessive division of unknown words (see Introduction), we reduced the cost for unknown words (30000 $\rightarrow$ 7000). We also changed its setting so that an unknown word are classified as a misc-proper-noun.

Then, we compared the above methods in terms of the averaged F-measures by 5-fold cross-validation of CRL NE data. The ME system attained 82.77% for $(h, k) = (1, 1)$ and 82.67% for $(2, 2)$. The RG+DT system attained 84.10% for $(h, k) = (1, 2)$, 84.02% for $(1, 1)$, and 84.03% for $(2, 2)$. (Even if we do not use C4.5, RG+DT attained 81.18% for $(1, 2)$ by removing bad templates with fewer positive examples than negative ones.) Thus, the two methods returned similar results. However, we cannot expect good performance for other documents because CRL NE is limited to January, 1995.

	CRL NE (Jan.'95)	all ('94-'95)	GENERAL ('99)	ARREST ('99)
ORG	3676+13	26725	361	74
PERSON	3840+4	23732	338	97
LOCATION	5463+38	32766	413	106
ARTIFACT	747	4890	48	13
DATE	3567+1	18497	260	72
TIME	502	3177	54	19
MONEY	390	3016	15	8
PERCENT	492	2783	21	0
TOTAL	18677+56	115586	1510	389

Table 1: Data used for comparison

Figure 2 compares these systems by using the formal run data. We cannot show the ME results for the large training data because Ristad's toolkit crashes even on a 2 GB memory machine. According to this graph, the RG+DT system's scores are comparable to those of the ME system. When all the training data was used, RG+DT's F-measure for GENERAL was 87.43%. We also examined RG+DT's variants. When we replaced character types of one-word NEs by '*', the score dropped to 86.79%. When we did not replace any character type by '*' at all, the score was 86.63%. RG+DT/n in the figure is a variant that also applies suffix dictionary to numerical NE classes.

When we used tokenized CRL NE for training, the RG+DT system's training time was about 3 minutes on a Pentium III 866 MHz 256 MB memory Linux machine. This performance is much faster than that of the ME system, which takes a few hours; this difference cannot be explained by the fact that the ME system is implemented on a slower machine. When we used all of the training data, the training time was less than one hour and the processing time of tokenized GENERAL (79 KB before tokenization) was about 14 seconds.

4 Discussion

Before the experiments, we did not expect that the RG+DT system would perform very well because the number of possible combinations of POS tags increases exponentially with respect to the num-

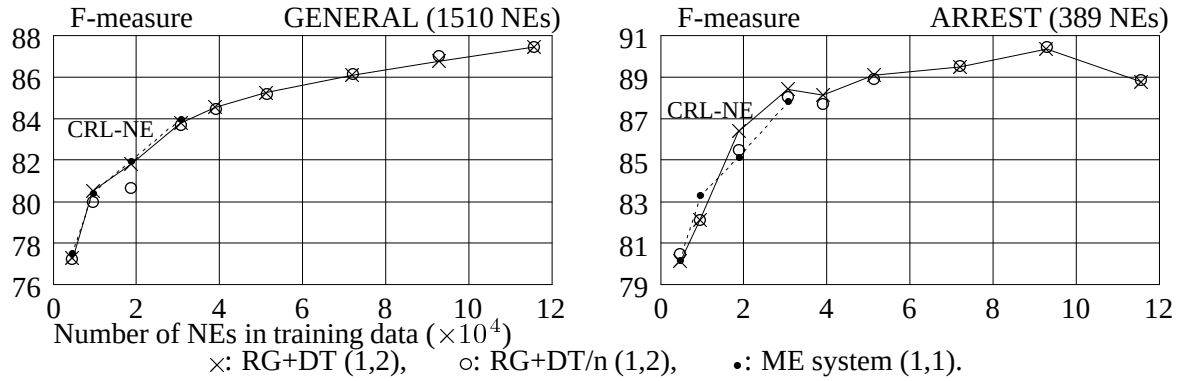

Figure 2: Comparison of RG+DT systems and Max. Ent. system

ber of words in an NE. However, the above results are encouraging. Its performance is comparable to the ME system. Why did it work so well? First, the percentage of long NEs is negligible. 91% of the NEs in the training data have at most three words. Second, the POS tags frequently used in NEs are limited.

When we compare the RG+DT method with other statistical methods, its advantage is its readability and independence of generated rules. When using cascaded rules, a small change in a rule can damage another rule's functionality. On the other hand, the recognition rules of our system are not cascaded (Fig. 1). Therefore, rewriting a recognition rule does not influence the performance of other rules at all. Moreover, dt-rules are usually very simple. When all of the training data were used, most of the RG+DT's recognition rules had a simple additional constraint that always accepts (65%) or rejects (16%) candidates. This result also implies the usefulness of our rule generator. Only 2% of the recognition rules have 10 or more dt-rules. For instance, the following recognition rule has dozens of dt-rules.

```
*:all-katakana:misc-proper-noun
  -> PERSON,0,0.
```

However, they are easy to understand as follows.

If the next word is `SHI` *(honorific), accept it.*
If the next word is `SAN` *(honorific), accept it.*
If the next word is `DAI-TOU-RYOU` *(=president), accept it.*
If the next word is `KAN-TOKU` *(=director), accept it.*
:
Otherwise, reject it.

We can explain this tendency as follows. Short NEs like 'Washington' are often ambiguous, but longer NEs like 'Washington State University' are less ambiguous. Thus, short recognition rules often have dozens of dt-rules, whereas long rules have simple constraints.

Some NE systems use decision tree learning to classify a word. Sekine's system (1998) is similar to the above ME systems, but C4.5 (Quinlan, 1993) is used instead. A similar system participated in IREX, but failed to show good performance. Borthwick (1999) explained the reason for this tendency. When he added lexical questions (e.g., whether the current word is x or not) to Sekine's system, C4.5 crashed with CRL NE. Accordingly, the decision tree systems did not directly use words as features. Instead, they used a word's memberships in their word lists.

Cowie (1995) interprets a decision tree deterministically and uses heuristic rewriting rules to get consistent results. Baluja's system (2000) simply determines whether a word is in an NE or not and does not classify it. On the other hand, Paliouras (2000) uses decision tree learning for classification of a noun phrase by assuming that named entities are noun phrases. Gallippi (1996) employs hundreds of hand-crafted templates as features for decision tree learning. Brill's rule generation method (Brill, 2000) is not used for NE tasks, but it might be useful.

Recently, unsupervised or minimally supervised models have been proposed (Collins and Singer, 2000; Utsuro and Sassano, 2000).

Collins' system is not a full NE system and Utsuro's score is not very good yet, but they represent interesting directions.

5 Conclusions

As far as we can tell, Japanese NE recognition technology has not yet matured. Conventional decision tree systems have not shown good performance. The maximum entropy method is competitive, but adding more training data causes problems. In this paper, we presented an alternative method based on decision tree learning and longest match. According to our experiments, this method's performance is comparable to that of the maximum entropy system, and it can be trained more efficiently. We hope our method can be applicable to other languages.

Acknowledgement

I would like to thank Yutaka Sasaki, Kiyotaka Uchimoto, Tsuneaki Kato, Eisaku Maeda, Shigeru Katagiri, Kenichiro Ishii, and anonymous reviewers.

References

James Allen. 1995. *Natural Language Understanding 2nd. Ed.* Benjamin Cummings.

Shumeet Baluja, Vibhu Mittal, and Rahul Sukthankar. 2000. Applying Machine Learning for High Performance Named-Entity Extraction. *Computational Intelligence*, 16(4).

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

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

Eric Brill. 2000. Pattern-based disambiguation for natural language processing. In *Proceedings of EMNLP/VLC-2000*, pages 1–8.

Michael Collins and Yoram Singer. 2000. Unsupervised models for named entity classification. In *Proceedings of EMNLP/VLC*.

Jim Cowie. 1995. CRL/NMSU description of the CRL/NMSU system used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference*, pages 157–166. Morgan Kaufmann.

Anthony F. Gallippi. 1996. Learning to recognize names accross lanugages. In *Proceedings of the International Conference on Computational Linguistics*, pages 424–429.

IREX Comittee. 1999. *Proceedings of the IREX Workshop (in Japanese)*.

MUC-6. 1996. *Proceedings of the Sixth Message Understanding Conference*. Morgan Kaufmann.

Georgios Paliouras, Vangelis Karkaletsis, Georgios Petasis, and Constantine D. Spyropoulos. 2000. Learning decision trees for named-entity recognition and classification. In *ECAI Workshop on Machine Learning for Information Extraction*.

J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.

Eric Sven Ristad, 1997. *Maximum entropy modeling toolkit, release 1.5 Beta.* `ftp:// ftp. cs. princeton. edu/ pub/ packages/ memt`, January.

Manabu Sassano and Takehito Utsuro. 2000. Named entity chunking techniques in supervised learning for Japanese named entity recognition. In *Proceedings of the International Conference on Computational Linguistics*, pages 705–711.

Satoshi Sekine and Yoshio Eriguchi. 2000. Japanese named entity extraction evaluation — analysis of results —. In *Proceedings of 18th International Conference on Computational Linguistics*, pages 1106–1110.

Satoshi Sekine, Ralph Grishman, and Hiroyuki Shinnou. 1998. A decision tree method for finding and classifying names in Japanese texts. In *Proceedings of the Sixth Workshop on Very Large Corpora*.

Kiyotaka Uchimoto, Qing Ma, Masaki Murata, Hiromi Ozaku, Masao Utiyama, and Hitoshi Isahara. 2000. Named entity extraction based on a maximum entropy model and transformation rules (in Japanese). *Journal of Natural Language Processing*, 7(2):63–90.

Takehito Utsuro and Manabu Sassano. 2000. Minimally supervised Japanese named entity recognition: Resources and evaluation. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, pages 1229–1236.