A Decision Tree Method for Finding and Classifying Names in Japanese Texts

Satoshi Sekine Ralph Grishman Hiroyuki Shinnou
Computer Science Department
New York University
715 Broadway, 7th floor
New York, NY 10003, USA
[sekine|grishman]@cs.nyu.edu

Department of Systems Engineering
Ibaraki University
Nakanarusawa, 4-12-1
Hitachi, Ibaraki, 316, Japan
shinnou@lily.dse.ibaraki.ac.jp

Abstract

This paper describes a system which uses a decision tree to find and classify names in Japanese texts. The decision tree uses part-of-speech, character type, and special dictionary information to determine the probability that a particular type of name opens or closes at a given position in the text. The output is generated from the consistent sequence of name opens and name closes with the highest probability. This system does not require any human adjustment. Experiments indicate good accuracy with a small amount of training data, and demonstrate the system's portability. The issues of training data size and domain dependency are discussed.

1 Introduction

For some NLP applications, it is important to identify "named entities" (NE), such as person names, organization names, time, date, or money expressions in the text. For example, in information extraction systems, it is crucial to identify them in order to provide the knowledge to be extracted, and in machine translation systems, they are useful for creating translations of unknown words or for disambiguation. However, it is not easy to identify these names, because they involve unknown words, and hence the strategy of listing candidates won't work. Also, it is sometimes hard to determine the category of proper nouns, like distinguishing a person name from a company name. These phenomena are often different from domain to domain. One domain may use a special pattern which is not found in other domains.

In this paper, we will present a supervised learning system which finds and classifies named entities in Japanese newspaper texts. Recently, several systems have been proposed for this task, but many of them use hand-coded patterns. Creating these patterns is laborious work, and when we adapt these systems to a new domain or a new definition of named entities, it is likely to need a large amount of additional work. On the other hand, in a supervised learning system, what is needed to adapt the system is to make new training data. While this is also not a very easy task, it would be easier than creating complicated rules. For example, based on our experience, 100 training articles can be created in a day.

There also have been several machine learning systems applied to this task. However, these either 1) partially need hand-made rules, 2) have parameters which must be adjusted by hand, or 3) do not perform well by fully automatic means. Our system does not work fully automatically and also needs special dictionaries, but performs well and does not have parameters to be adjusted by hand. We will discuss one of the related systems in a later section.

The issue of training data size will be discussed based on experiments using different sizes of training data. In order to demonstrate the portability of our system, we ran the system on a new domain with a new type of named entity. The experiment shows that the portability of the system is quite good and the performance is satisfactory.

2 Task

The task is to find and classify several types of named entity items in texts, shown in Table 1. We use the task definition provided in the MET-2 guidelines (Multilingual Entity Task; the formal definition will be published in May 1998). "Executive position" is a new category which is used in the portability experiment only and is not part of the MET definition.

There are some idiosyncratic definitions. For

Named Entity	Examples
Organization	Matsushita, Matsushita Electric Industrial Co.Ltd.
Person	Mr. Matsushita, Mike
Location	U.S.A., Matsushita
Position	President, Professor
Date	March 5, 21st century
Time	12:09, noon, morning
Money	100,000 yen, 1 ECU
Percent	10%, a quarter

Table 1: Named Entities (NE)

example, a sub-organization expression like "Executive Staff" should be identified only when it follows an organization in proper expression. So, in the expression "Defense Ministry's Executive Staff", "Executive Staff" should be identified; however, it should not be identified if it appears alone in a sentence. Also, a country expression at the head of an organization should be identified if it is expressed by one Chinese character, but it should not when it is expressed by Katakana characters. Although we find some idiosyncratic definitions in the guidelines, we will use them, because there are such difficulties in nature and we can't easily find another reasonable definition.

3 Algorithm

In this section, the algorithm of the system will be presented. There are two phases, one for creating the decision tree from training data (training phase) and the other for generating the tagged output based on the decision tree (testing phase). We use a Japanese morphological analyzer, JUMAN (JUMAN, 1997) and a program package for decision trees, C4.5 (Quinlan, 1993). We use three kinds of feature sets in the decision tree:

- Part-of-speech tagged by JUMAN
  We define the set of our categories based on its major category and minor category.

- Character type information
  Character type, like Kanji, Hiragana, Katakana, alphabet, number or symbol, etc. and some combinations of these.

- Special Dictionaries
  Lists of entities created based on JUMAN dictionary entries, lists found on the Web or based on human knowledge. Table 2 shows the number of entities in each dictionary. Organization name has two types of dictionary; one for proper names and the other for general nouns. An example of the latter case is "Executive Staff", mentioned before.

Entity	name-prefix	name	name-suffix
Org.	9	7018/49	96
Person	0	17851	7
Loc.	0	14863	61
Position	0	75	0
Date	24	198	29
Time	2	25	5
Money	22	0	39
Percent	0	99	3

Table 2: Special Dictionary Entry

Creating the special dictionaries is not very easy, but it is not very laborious work. The initial dictionary was built in about a week. In the course of the system development, in particular while creating the training corpus, we added some entities to the dictionaries.

The decision tree gives an output for the beginning and the ending position of each token. It is one of the 4 possible combinations of opening, continuation and closing for each named entity type, or having no named entity. shown in Table 3. When we have 8 named entity types, there are 33 kinds of output. For example, if an organization name covers three words, A, B and C,

output	beginning of token	ending of token	token is
OP-CL	opening	closing	NE itself
OP-CN	opening	cont.	starting NE
CN-CN	cont.	cont.	middle of NE
CN-CL	cont.	closing	ending NE
none	none	none	not NE

Table 3: Five types of Output

and the next word D has no named entity, then we will have the following data:

```
A : org-OP-CN
B : org-CN-CN
C : org-CN-CL
D : none
```

Note that there is no overlapping or embedding of named entities. An example of real data is shown in Appendix A.

There could be a problem, in the testing phase, if we just use the deterministic decision created by the tree. Because the decisions are made locally, the system could make an inconsistent sequence of decisions overall. For example, one token could be tagged as the opening of an organization, while the next token might be tagged as the closing of person name. We can think of several strategies to solve this problem (for example, the method adopted by (Bennett et al. 1997) will be described in a later section), but we used a probabilistic method.

There will usually be more than one tag in the leaf of a decision tree. At a leaf we don't just record the most probable tag; rather, we keep the probabilities of the all possible tags for that leaf. In this way we can salvage cases where a tag is part of the most probable globally-consistent tagging of the text, even though it is not the most probable tag for this token, and so would be discarded if we made a deterministic decision at each token. Note that, we did not apply smoothing technique, which might be able to avoid the data sparseness problem. More about the probabilistic method will be explained in the next section.

Training Phase

First, the training sentences are segmented and part-of-speech tagged by JUMAN. Then each token is analyzed by its character type and is matched against entries in the special dictionaries. One token can match entries in several dictionaries. For example, "Matsushita" could match the organization, person and location dictionaries.

Using the training data, a decision tree is built. It learns about the opening and closing of named entities based on the three kinds of information of the previous, current and following tokens. The three types of information are the part-of-speech, character type and special dictionary information described above.

Testing Phase

In the testing phase, the first three steps, token segmentation and part-of-speech tagging by JUMAN, analysis of character type, and special dictionary look-up, are identical to that in the training phase. Then, in order to find the probabilities of opening and closing a named entity for each token, the properties of the previous, current and following tokens are examined against the decision tree. Appendix B shows two example paths in the decision tree. For each token, the probabilities of 'none' and the four combinations of answer pairs for each named entity type are assigned. For instance, if we have 7 named entity types, then 29 probabilities are generated.

Once the probabilities for all the tokens in a sentence are assigned, the remaining task is to discover the most probable consistent path through the sentence. Here, a consistent path means that for example, a path can't have `org-OP-CN` and `date-OP-CL` in a row, but can have `loc-OP-CN` and `loc-CN-CL`. The output is generated from the consistent sequence with the highest probability for each sentence. The Viterbi algorithm is used in the search; this can be run in time linear in the length of the input.

4 Example

Appendix A shows an example sentence along with three types of information, part-of-speech, character type and special dictionary information, and information of opening and closing of named entities. Appendix B shows two example paths in the decision tree. For the purpose of demonstration, we used the seventh and eighth token of the example sentence in Appendix A. Each line corresponds to a question asked by the tree nodes along the path. The last line shows the probabilities of named entity information which have none-zero probability. This instance demonstrates how the probability method works. As we can see, the probability of none for the seventh token (`Isuraeru` = Israel) is higher than that for the opening of organization (0.67 to 0.33), but in the eighth token (`Keisatsu` = Police), the probability of closing organization is

much higher than none (0.86 to 0.14). The combined probabilities of the two consistent paths are calculated. One of these paths makes the two tokens an organization entity while along the other path, neither token is part of a named entity. The probabilities are higher in the first case (0.28) than that in the latter case (0.09), So the two tokens are tagged as an organization entity.

5 Experiments

In this section, the experiments will be described. We chose two domains for the experiments. One is the vehicle accident report domain. Newspaper articles in the domain report accidents of vehicles, like car, train or airplane. The other is the executive succession domain, articles in this domain report succession events of executives, like president, vice president or CEO. We have 103 training articles in the accident domain, which contain 2,368 NE's and 11 evaluation articles which were hidden from the developer. In the evaluation articles, there are 258 NE items (58 organization, 30 person, 100 location, 47 date, 21 time and 2 money expressions). Also, we have 70 training articles, which contain 2,406 NE's and 17 evaluation articles in the succession domain. In the evaluation articles, there are 566 NE items (113 organization, 114 person, 67 location, 183 position, 77 date, 1 time, 9 money and 2 percent expressions).

5.1 Accident Report Domain

First, we will report on the experiment on the accident domain. Basically, this is the initial target domain of the system.

The result is shown in Table 4. The F-scores based on recall and precision are shown. 'Recall' is the percentage of the correct answers among the answers in the key provided by human. 'Precision' is the percentage of the correct answers among the answers proposed by the system. 'F-score' is a measurement combining the two figures. See (Tipster2, 1996) for more "detail" definition of F-score, recall and precision. They are compared with the results produced by JUMAN's part-of-speech information and the average scores in MET1, reported in (Tipster2, 1996). The result from JUMAN is created based on JUMAN version 3.3's output alone[1]. When it identifies a sequence of locations, persons or other proper nouns, then we tag the sequence with location, person or organization, respectively. The MET1 evaluation was conducted on completely different texts and on a different domain, so it is not directly comparable, but since the task definitions are almost the same, we believe it gives a rough point of comparison. Note that for the MET1 evaluation, there were about 300 training articles compared to our 100 training articles. Also, they did not report the scores by each individual participant.

Entity	Our score	JUMAN only	MET1 ave. score
Org.	86	56	73
Person	91	63	77
Loc.	87	51	82
Date	96	-	94
Time	91	-	93
Money	100	-	95
Percent	-	-	96
Overall	85	-	-

Table 4: Result in Accident Report Domain

We believe these results are quite good and indicate the capability of our system. In terms of execution time, the training phase takes about 5 minutes, of which JUMAN and the decision tree creation take most of the time. It takes less than a minute to create the named entity output, and again JUMAN takes the bulk of the time.

5.2 Issue of Training Size

It is quite nice that we can get this level of performance with only about 100 training articles. It is interesting to investigate how much training data is needed to achieve a good performance. We created 8 small training sets of different size, and ran the system using these training data. Note that we used the same dictionaries for all the experiments, which were generated by several means including the items in the entire training data. Table 5 shows the results. The size of the training set is indicated by the number of articles and the number of NE in the training data. It is amazing that the performance is not greatly degraded even with 9 articles. Also, even with

[1] Latest version may have better performance than the results reported here. Also remember that the definitions are different.

only one article, our system can achieve 68 F-score. Actually, the three sets of 1-article training data were created from each article in the 3-article training data, and we can see that the performance using the 3-article training data is mainly derived from the high performance single article. So, we believe that once you have a good coverage dictionaries and some amount of standard patterns in the training data, the system can achieve fairly good performance. We observed that the article which gives high performance contains a good variety of many named entities.

Size of Training	score
103 (2368)	85
69 (1586)	86
35 (721)	80
18 (384)	81
9 (216)	79
3 (59)	71
1 (23/13/23)	68/21/41

Table 5: Result for Training Data Size

5.3 Executive Succession Domain - Portability -

In general, one of the advantages of automatic learning systems is their portability. In this subsection, we will report an experiment of moving the system to a new domain, the executive succession domain. Also, in order to see the portability of the system, we add a new kind of named entity. In this domain, executive positions appear very often and it is an important entity type for understanding those articles. So, we add a new entity class, 'position'. When porting the system, only the following two changes are required.

1. Add a new dictionary
   Create a new dictionary for positions. In practice, many of them were listed in the person prefix in the previous experiment. So we separate them and add several position names which appeared in or could be inferred from the training data. This took less than an hour. Note that we did not change any other dictionaries, i.e. organization, location dictionary, etc. We believe that these dictionaries can be relatively domain independent.
2. Modify the program
   Assign a new ID number for the position entity in the decision tree program and modify the input/output routine accordingly. This also took less than an hour.

In less than two hours for the system modification, and about a day's work for the preparation of the training data, the new system becomes runnable, Table 6 shows the result of the experiment. The result is quite satisfactory. However,

Entity	score
Org.	72
Person	88
Loc.	67
Position	93
Date	89
Time	100
Money	90
Percent	100
Overall	84

Table 6: Result in Executive Succession Domain

it is not as good as the result in the previous domain, in particular, for organization and location. Observing the output, we noticed domain idiosyncrasies which we had not thought of before. For example, in the new domain, there are many Chinese company names, which have the suffix "Yuugenkoushi". This is never used for Japanese company names and we don't have the suffix in our organization suffix dictionary. Another interesting example is a Chinese character "Shou". In Japanese, the character is used as a suffix of official organizations, like "Monbu-Shou" (Department of Education), but in Chinese it is used as a suffix of location names, like "Kanton-Shou" (Canton District). In the accident domain, we did not encounter such Chinese location names, so we just had the token in the organization suffix dictionary. This led to many errors in location names in the new domain. Also, we find many unfamiliar foreign location names and company names. We believe these make the result relatively worse.

5.4 Domain Dependency

As we have training and evaluation data on two different domains, it is interesting to observe the domain dependency of the system. Namely, we will see how the performance differs if we use the knowledge (decision tree) created from a different domain. We conducted two new experiments, tagging named entities for texts in the succession domain based on the decision tree created for the accident domain, and vice versa.

Table 7 shows the comparison of these results. The performance in the accident domain decreased from 85 to 71 using the decision tree of the other domain. Also, the performance decreased from 82 to 59 in the succession domain.

Test \ Train	Acc.	Suc.
Accident	85	71
Succession	59	82

Table 7: Result on Domain Dependency

The result demonstrates the domain dependency of the method used, at least for the two domains. Obviously, making a general comment based on these small experiments is dangerous, but it suggests that we should consider the domain dependency when we port the system to a new domain.

6 Related Work

There have been several efforts to apply machine learning techniques to the same task (Cowie, 1995) (Bikel et al, 1997) (Gallippi, 1996) (Bennett et al, 1997) (Borthwick et al, 1997). In this section, we will discuss a system which is one of the most advanced and which closely resembles our own (Bennett et al, 1997). A good review of most of the other systems can be found in their paper.

Their system uses the decision tree algorithm and almost the same features. However, there are significant differences between the systems. The main difference is that they have more than one decision tree, each of which decides if a particular named entity starts/ends at the current token. In contrast, our system has only one decision tree which produces probabilities of information about the named entity. In this regard, we are similar to (Bikel et al, 1997), which also uses a probabilistic method in their HMM based system. This is a crucial difference which also has important consequences. Because the system of (Bennett et al, 1997) makes multiple decisions at each token, they could assign multiple, possibly inconsistent tags. They solved the problem by introducing two somewhat idiosyncratic methods. One of them is the distance score, which is used to find an opening and closing pair for each named entity mainly based on distance information. The other is a tag priority scheme, which chooses a named entity among different types of overlapping candidates based on the priority order of named entities. These methods require parameters which must be adjusted when they are applied to a new domain. In contrast, our system does not require such methods, as the multiple possibilities are resolved by the probabilistic method. This is a strong advantage, because we don't need manual adjustments.

The result they reported is not comparable to our result, because the text and definition are different. But the total F-score of our system is similar to theirs, even though the size of our training data is much smaller.

7 Discussion

This paper has described a system which uses a decision tree to find and classify names in Japanese texts. Experiments indicate good accuracy with a small amount of training data, and demonstrate the system's portability. The issues of training data size and domain dependency were discussed.

We would like to discuss the issue of the hand created dictionaries. People might think that the hand made dictionaries play the major role in the system. It may be true, but we should remember that the experiment in the Executive Succession Domain use the same pre-exist dictionaries used in the Accident Domain. We did not modify any dictionaries used in the previous domain, we only added the dictionary for the position. Although we found some dictionary entities which should be added, the fact that we achieved good performance in the new domain by using the same dictionaries shows that dictionaries are not so domain dependent. Once we prepared the dictionaries, we might not need to modify them to a great degree. Also, Table 7 suggests that the

decision tree rules are more domain dependent rather the dictionaries.

We have several ideas in order to improve our system.

The most crucial and most elaborate step in building up the system is creating the dictionaries. It was done by hand, because 100 training articles are not enough to acquire even prefixes and suffixes. One possibility is to use a bootstrapping method. Starting with core dictionaries, we can run the system on untagged texts, and increase the entities in the dictionaries.

Another issue is aliases. In newspaper articles, aliases are often used. The full name is used only the first time the company is mentioned (`Matsushita Denki Sangyou Kabushiki Kaisya` = Matsushita Electric Industrial Co. Ltd.) and then aliases (`Matsushita` or `Matsushita Densan` = Matsushita E.I.) are used in the later sections of the article. Our system cannot handle these aliases, unless the aliases are registered in the dictionaries.

Also, lexical information should help the accuracy. For example, a name, possibly a person or an organization, in a particular argument slot of a verb can be disambiguated by the verb. For example, a name in the object slot of the verb 'hire' might be a person, while a name in the subject slot of verb 'manufacture' might be an organization.

8 Acknowledgment

We would like to thank our colleague at NYU, in particular Mr.Andrew Borthwick and Mr.John Sterling. There comments and discussion were useful for the research.

References

Defense Advanced Research Projects Agency 1996 Proceedings of Workshop on Tipster Program Phase II *Morgan Kaufmann Publishers*

Scott Bennett, Chinatsu Aone and Craig Lovell 1997 Learning to Tag Multilingual Texts Through Observation *Conference on Empirical Methods in Natural Language Processing*

Daniel Bikel, Scott Miller, Richard Schwartz and Ralph Weischedel 1997 Nymble: a High-Performance Learning Name-finder *Proceedings of the Fifth Conference on Applied Natural Language Processing*

Andrew Borthwick, John Sterling, Eugene Agichtein and Ralph Grishman 1998 Exploiting Diverse Knowledge Sources via Maximum Entropy in Named Entity Recognition *Proceedings of the Sixth Workshop on Very Large Corpora*

Anthony Gallippi 1996 Learning to Recognize Names Across Languages *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*

Jim Cowie 1995 Description of the CRL/NMSU Systems Used for MUC-6 *Proceedings of Sixth Message Understanding Conference (MUC-6)*

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

Yuuji Matsumoto, Sadao Kurohashi, Osamu Yamaji, Yuu Taeki and Makoto Nagao 1997 Japanese morphological analyzing System: JUMAN *Kyoto University and Nara Institute of Science and Technology*

Appendix A: Example training data

Token	POS	String type	Special Dict.	Named entity answer
[	[	Sym	-	-
ERUSAREMU	PN-loc	Kata	loc	loc-OP-CL
26	number	Num	-	date-OP-CN
NICHI	N-suf	Kanji	date-S	date-CN-CL
KYODO	PN	Kanji	org	org-OP-CL
]	]	Sym	-	-
ISURAERU	PN-loc	Kata	loc	org-OP-CN
KEISATSU	N	Kanji	org-S	org-CN-CL
NI	postpos	Hira	-	-
YORU	V	Hira	-	-
TO	postpos	Hira	-	-
,	comma	Comma	-	-
ERUSAREMU	PN-loc	Kata	loc	loc-OP-CN
SHI	N-suf	Kanji	loc-S	loc-CN-CL
HOKUBU	N	Kanji	-	-
DE	postpos	Hira	-	-
26	number	Num	-	date-OP-CN
NICHI	N-suf	Kanji	date-S	date-CN-CL
GOGO	N	Kanji	time, time-P	time-OP-CL
,	comma	Comma	-	-

Appendix B: Example paths in the tree

```
ISURAERU  (seventh token)
 if current token  is a location      -> yes
 if next token     is a loc-suffix    -> no
 if next token     is a person-suffix -> no
 if next token     is a org-suffix    -> yes
 if previous token is a location      -> no
then none = 0.67, org-OP-CN = 0.33

KEISATSU  (eighth token)
 if current token  is a location      -> no
 if current token  is a organization  -> no
 if current token  is a time          -> no
 if current token  is a loc-suffix    -> no
 if next token     is a time-suffix   -> no
 if current token  is a time-suffix   -> no
 if next token     is a date-suffix   -> no
 if current token  is a date-suffix   -> no
 if current token  is a date          -> no
 if next token     is a location      -> no
 if current token  is a org-suffix    -> yes
 if previous token is a location      -> yes
then none = 0.14, org-CN-CL = 0.86
```