NYU: DESCRIPTION OF THE JAPANESE NE SYSTEM USED FOR MET-2

Satoshi Sekine
Computer Science Department
New York University
715 Broadway, 7th floor
New York, NY 10003, USA
sekine@cs.nyu.edu

INTRODUCTION

In this paper, experiments on the Japanese Named Entity task are reported. We employed a supervised learning mechanism. 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 and maybe additional small work. While this is also not a very easy task, it would be easier than creating complicated patterns. 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 3) do not perform well by fully automatic means or 4) need a huge training data. Our system does not work fully automatically, but performs well with a small training corpus and does not have parameters to be adjusted by hand. We will discuss one of the related systems later..

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 (running phase). We use a Japanese morphological analyzer, JUMAN [6] and a program package for decision trees, C4.5 [7]. 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
  List of entities created based on JUMAN dictionary entries, lists distributed by SAIC for MUC, lists found on the Web or based on human knowledge. Table 1 shows the number of entities in each dictionary[1]. Organization name has two types of dictionary; one for proper names and the other for general nouns which should be tagged when they co-occur with proper names. Also, we have a special dictionary which contains words written in Roman alphabet but most likely these are not an organization (e.g. TEL, FAX). We made a list of 93 such words.

[1] Some of the lists are available at [8]

Entity	prefix	name	suffix
Org.	14	10076/49	175
Person	0	17672	82
Loc.	0	14903	60
Date	24	199	29
Time	2	24	5
Money	15	0	39
Percent	0	99	3

Table 1: Special Dictionary Entries

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 during creating the training corpus, we added some entities to the dictionaries.

The decision tree gives an output for each token. It is one of the 4 possible combinations of opening, continuation and closing a named entity, and having no named entity, shown in Table 2. In this paper, we

Output	beginning	ending
`OP-CL`	opening of NE	closing of NE
`OP-CN`	opening of NE	cont. of NE
`CN-CN`	cont. of NE	cont. of NE
`CN-CL`	cont. of NE	closing of NE
`none`	none	none

Table 2: Five types of Output

will use two different sets of terms in order to avoid the confusion between positions relative to a token and regions of named entities. The terms `beginning` and `ending` are used to indicate positions, whereas `opening` and `closing` are used to indicate the start and end of named entities. Note that there is no overlapping or embedding of named entities. An example of real data is shown in Figure 1.

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.

If we just use the deterministic decision created by the tree, it could cause a problem in the running phase. 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 by [2] will be described in a later section), but we used a probabilistic method.

The instances in the training corpus corresponding to a leaf of the decision tree may not all have the same tag. 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.

subsectionRunning Phase

In the running 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. Figure 2 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.

EXAMPLE

Figure 1 shows an example sentence along with three types of information, part-of-speech, character type and special dictionary information, and given information of opening and closing of named entities. Figure 2 shows two example paths in the decision tree. For the purpose of demonstration, we used the first and second token of the example sentence in Figure 1. 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 more than 0.0 probability. This instance demonstrates how the probability method works. As we can see, the probability of none for the first token (`Isuraeru` = Israel) is higher than that for the opening of organization (0.67 to 0.33), but in the second 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.

Token	ISURAERU	KEISATSU	NI	YORU	TO	,	ERUSAREMU
POS	PN-loc	N	postpos	V	postpos	comma	PN-loc
Char.type	Kata	Kanji	Hira	Hira	Hira	Comma	Kata
Special Dict.	loc	org-S	-	-	-	-	loc
NE answer	org-OP-CN	org-CN-CL	-	-	-	-	loc-OP-CN

Token	SHI	HOKUBU	DE	26	NICHI	GOGO	,
POS	N-suf	N	postpos	number	N-suf	N	comma
Char.type	Kanji	Kanji	Hira	Num	Kanji	Kanji	Comma
Special Dict.	loc-S	-	-	-	date-S	time,time-P	-
NE answer	loc-CN-CL	-	-	date-OP-CN	date-CN-CL	time-OP-CL	-

Figure 1: Sentence Example

RESULTS

We will report results of five experiments described in Table 3. Here, "Training data", "Dry run data" and "Formal run data" are the data distributed by SAIC, and "seefu data" is the data created by Oki, NTT data and NYU (available through [8]). Note that all Training, Dry run and seefu data are in the topic of

```
ISURAEKU  (first 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  (second 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
```

Figure 2: Decision Tree Path Example

vehicle crash, and only the Formal run data is on the topic of space craft launch. Numbers in the brackets indicate the number of articles.

Experiment	Training Data	Test Data
1) Formal run	Training data(114), seefu data(150), Dry run data(30)	Formal run data
2) Best in-house Dry run	Training data(114), seefu data(150)	Dry Run data
3) 75/25 experiment	75% of Formal run Data(75)	25% of Formal run data
4) All training + 75/25	Training data(114), seefu data(150), Dry run data(30),75% of Formal run data(75)	25% of Formal run data
5) Add planet names	same as 4)	same as 4)

Table 3: Runs

The results of Formal run and the best in-house dry-run are shown in Table 4. We can clearly tell that the recall of Named Entities (person, organization and location) are bad. This is caused by the change of the topic. For example, there are very few foreign person names written in Katakana in the training data, (as a foreign person would hardly be a victim of a crash in Japan). However, in the space craft launch, there are many foreign person names written in Katakana. This is the reason why the recall of persons is so low. Also, in the test documents, planet names, "the Sun","the Earth" or "Saturn" are tagged as locations, which could not be predicted from the training topic. We missed all such names in the formal test.

The best in-house Dry run result was achieved before the formal run without looking at the test data. So it should be regarded as an example of the performance if we know the topic of the material. We think this is satisfactory, considering that the effort we made was just preparing dictionaries and no patterns.

Table 5 shows three experiments performed after the formal run. As the topic change may degrade of the performance, we conducted experiments in which the training data includes documents in the same topic. The first experiment used 75% of the formal run data for training and the rest of the data for testing. Four such experiments were made to obtain the result for the entire corpus. The second experiment includes the training data used in the formal run in addition to the 75% of the formal run data. The table shows about 1% improvement over the formal run. This is an encouraging result, the better performance was achieved with only 75 articles on the same topic compared with 294 articles on a different topic used in the formal run. The result of the second experiment also shows a good sign that documents in a different topic helped to improve the performance. This result suggests an idea of "domain adaptation scheme". That is to have a

F-measure	Formal run		Best in-house dry run	
Entity	Recall	Precision	Recall	Precision
Org.	75	83	78	87
Person	48	74	87	90
Loc.	70	87	91	95
Date	96	95	97	91
Time	95	96	98	98
Money	90	97	100	100
Percent	90	95	88	100
Overall	75	85	87	90
F-measure	79.49		88.62	

Table 4: Result of Formal Run and the best in-house Dry run

large general corpus of tagged documents as the basis, and to add small domain specific documents to have a domain specific system. Lastly, in the third experiment, we added the planet names in the location dictionary. From the formal run result, it was clear that one of the main reasons of the performance degradation is the lack of the planet names. The addition improves 3.5% which is better than the other trials. Although there are several other obvious reasons to be fixed, the F-measure 86.34 is comparable to the best in-house Dry run experiment described before (Experiment 2;F-measure = 88.62).

Experiment	F-measure
3) 75/25 experiment	80.46
4) All training + 75/25	82.73
5) Add planet names	86.34

Table 5: Comparative Results

RELATED WORK

There have been several efforts to apply machine learning techniques to the same task [4] [3] [5] [2]. In this section, we will discuss a system which is one of the most advanced and which closely resembles our own [2]. 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 [3], which also uses a probabilistic method in their N-gram based system. This is a crucial difference which also has important consequences. Because the system of [2] 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 the 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.

DISCUSSION

First, we have to consider topic or domain dependency of the task. It is clear that in order to achieve good performance in the framework, we have to investigate dictionary entries for the task. It may or may not easy to modify the dictionary. For example, a list of foreign person name written in Katakana is not so easy to create, whereas a list of planet names is easy to find. This difficulty also exists in pattern-based methods, but in our framework it is not necessary to create domain dependent patterns.

Currently creating dictionaries is done by hand. One possibility to automatize the process 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.

REFERENCES

[1] Defense Advanced Research Projects Agency, "Proceedings of Workshop on Tipster Program Phase II" *Morgan Kaufmann Publishers* (1996)

[2] Bennett, S., Aone, C. and Lovell, C., "Learning to Tag Multilingual Texts Through Observation" *Conference on Empirical Methods in Natural Language Processing* (1997)

[3] Bikel, D., Miller, S., Schwartz, R. and Weischedel, R., "Nymble: a High-Performance Learning Name-finder" *Proceedings of the Fifth Conference on Applied Natural Language Processing* (1997)

[4] Cowie, J., "Description of the CRL/NMSU Systems Used for MUC-6" *Proceedings of Sixth Message Understanding Conference (MUC-6)* (1995)

[5] Gallippi, A., "Learning to Recognize Names Across Languages" *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)* (1996)

[6] Matsumoto, Y., Kurohashi, S., Yamaji, O., Taeki, Y. and Nagao, M., "Japanese morphological analyzing System: JUMAN" *Kyoto University and Nara Institute of Science and Technology* (1997)

[7] Quinlan, R., "C4.5: Program for Machine Learning" *Morgan Kaufmann Publishers* (1993)

[8] Sekine, S., "Homepage of data related Japanese Named Entity" *http://cs.nyu.edu/cs/projects/proteus/met2j* (1997)