Japanese IE System and Customization Tool

Chikashi Nobata
Department of Information Science
University of Tokyo
Science Building 7. Hongou 7-3-1
Bunkyo-ku, Tokyo 113 Japan
nova@is.s.u-tokyo.ac.jp

Satoshi Sekine, Roman Yangarber
Computer Science Department
New York University
715 Broadway, 7th floor
New York, NY 10003, USA
{sekine|roman}@cs.nyu.edu

1 Introduction

This paper reports on the development of the Japanese Information Extraction system and the Japanese Information Extraction customization tool. These systems are based on the corresponding English systems, the Proteus Information Extraction system (Grishman 97) and the Proteus Extraction Tool (Yangarber and Grishman 97), developed at NYU. In this paper, we will, in particular, describe the differences between the Japanese systems and English systems.

2 The Japanese IE System

Figure 1 shows the overall structure of the Japanese IE system. The system consists of a cascade of modules with their attendant knowledge bases, with the input text document passed through the pipeline of modules. Because Japanese sentences have no spaces between the tokens, we first have to run a morphological analyzer in order to segment each sentence into tokens. Although an alternative is to use the input sequence of characters as it is, this may cause a serious problem when we apply patterns because of ambiguities. The first module, *morphological analysis* is responsible for breaking the sentence into tokens. We used JUMAN (Matsumoto et al. 97) for this purpose. It also provides the part-of-speech information for each token, which will be used in the next module.

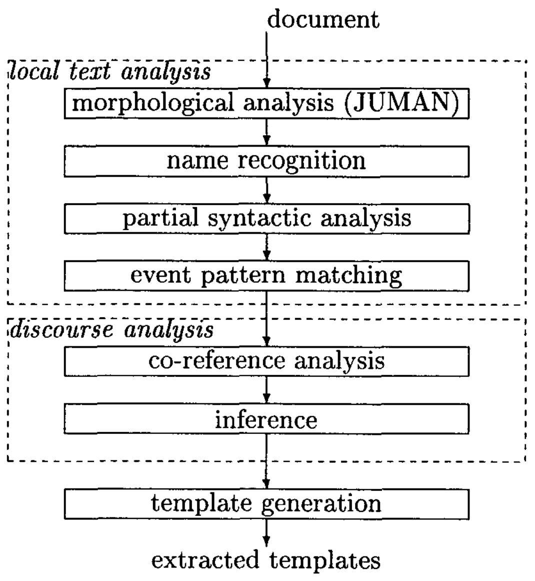

Figure 1: Japanese Proteus IE system architecture

The second module is responsible for extracting named entities, like organization, person, location names, time expressions, and numeric expressions. This is different from the English system, which uses the pattern matching mechanism for named entity detection. In the Japanese system, this is done by an independent program which uses a decision tree technique to assign tags of named entity information to the input text (Sekine 98).

The next two modules employ regular expression *pattern matching*, applying patterns of successively increasing complexity to the document. These are essentially the same as those of the English system. The patterns – the regular expressions with their associated actions, – are encoded in a special pattern specification language, and are compiled and stored in a separate pattern base. Pattern matching is a form of deterministic, bottom-up partial parsing. The numbers of patterns between the two systems are slightly different mainly due to the difference in the methods of detecting named entities.

Language	Event pattern	Other pattern	Total
English	51	187	238
Japanese	39	64	103

Table 1: Number of patterns

The following explains the pattern matching component using some simplified examples of Japanese patterns. There are two phases in the pattern matching component. The first phase uses patterns to identify small syntactic units, such as noun and verb phrases. Thus, e.g., in order to analyze the example sentence in Appendix A, this phase will employ a pattern such as:

np(*person*) np(*position*)

which matches an ordered pair of noun phrases (*np*) whose lexical heads belong to the semantic class "person" and "position", respectively. The pattern's action will subsume the matched text into a new, larger constituent, with the side effect that the *entity* corresponding to the "person" noun phrase will acquire a slot named "position", linked to the position entity.

The second phase called *event patterns* applies domain-specific and scenario-specific patterns, to resolve higher-level syntactic constructions (apposition, prepositional phrase attachment), conjunction, and clause constructions. The actions triggered by the patterns specify operations on the *logical form* representation of the sentence, which evolves as the pattern matching progresses. The logical form contains the descriptions of the entities, relationships, and events discovered so far by the analysis. For example, there is a pattern

np(*person*) `GA` np(*position*) `NI` vp(`SHOUKAKU-SURU`)

(here, the Japanese characters are written in `typewriter` type-face) This is an event pattern and it constructs several relationships regarding the person, position and the predicate `SHOUKAKU-SURU` (to promote).

The subsequent phases operate on the logical form built in the pattern matching phases. *Reference resolution* links anaphoric pronouns to their antecedents, and resolves other co-referring expressions. We create some Japanese-specific rules for abbreviations and any other equivalent expressions.

The Japanese system uses the corresponding English components for unification of partial event structures and template generation.

3 Customization Tool

There is not much to mention about the porting of the English Customization tool (PET) to Japanese. There were programming level difficulties because its window system is developed using GARNET, a graphical system for LISP, which does not support the Japanese language.

4 Evaluation

Table 2 shows the result of the information extraction experiment for English and Japanese. The English results are based on the formal MUC-6 experiment. The Japanese experiment is conducted using Nikkei newspaper articles on the same domain, "executive succession events". We spent about two months of one person's part-time labor to develop the patterns. Actually, the developer created the patterns at the same time as he developed the Japanese system. It achieves a slightly better score. This may due to the fact that he spent more time than that spent for the English rule creation (one month), or due to the tendency we found that there is a typical document style for this kind of articles. The latter is not clear as we have not investigated the English articles.

5 Future Work

Structural generalization

In the English system, there is a facility to generalize a pattern based on structural variation. For example, an active mood pattern can be generalized to a passive pattern or a relative clause pattern. A similar mechanism might be useful in Japanese. Japanese is known as a free word-order language. In principle, a pattern can be generalized based on this property of Japanese, but there are some exceptions and some heuristics should be considered.

Reference resolution for zero pronoun

In Japanese, subject or object nouns can sometimes be omitted. It is a difficult problem to recover the noun, but we found several cases where such technologies are needed to achieve better performance.

6 Acknowledgments

The English Proteus Information Extraction system, on which this work is based, was developed by Professor Ralph Grishman. We would like to gratefully acknowledge his assistance in the development of the Japanese system.

References

Ralph Grishman:
Information extraction: Techniques and challenges. In Maria Teresa Pazienza, editor, *Information Extraction.* Springer-Verlag, Lecture Notes in Artificial Intelligence, Rome, 1997.

Yuuji Matsumoto, Sadao Kurohashi, Osamu Yamaji, Yuu Taeki and Makoto Nagao: Japanese

Language	Corpus	# of articles	Recall	Precision	F-measure
English	training	100	61	74	67.01
	test	51	47	70	56.40
Japanese	training	87	64	71	67.32
	test	90	57	70	62.61

Table 2: Evaluation

morphological analyzing System: JUMAN *Kyoto University and Nara Institute of Science and Technology* 1997

Satoshi Sekine: NYU:Description of The Japanese NE System Used for MET-2 In *Proc. MUC-7* Verginia, USA, 1998.

Roman Yangarber and Ralph Grishman: Customization of Information Extraction Systems In *Proc. Int'l Workshop on Lexically Driven Information Extraction*, pages 1–11, Frascati, Italy, 1997.

Appendix A:
===========

日興国際投資顧問は二十四日付けで、中西孝雄専務を社長に昇格させる人事を発表した。小林忠雄社長は代表権を持つ会長に昇格する。

Position	Company	Person	Status	On_the_Job	Reason
専務	日興国際投資顧問	中西孝雄	OUT	YES	Reassignment
社長	日興国際投資顧問	中西孝雄	IN	NO	Reassignment
社長	日興国際投資顧問	小林忠雄	OUT	YES	Reassignment
会長	日興国際投資顧問	小林忠雄	IN	NO	Reassignment

Appendix B:
===========

```
(defpattern jpn-decision4
  "np-sem(C-company) 'は' j-at-date? j-at-board-of-directors?
   np-sem(C-person) np-sem(C-position) 'を'
   np-sem(C-position) 'に'
   jnoun-appoint 'さ' 'せる' jnoun-jinji 'を' jverb-decision:
  verb-tense=future,
  person-at=5.attributes,
  company-prev-at=1.attributes,
  person-prev-at=6.attributes,
  company-at=1.attributes,
  position-at=8.attributes")

(defclausefun when-j-appoint (phrase-type)
  (let ((person-at (binding 'person-at))
        (person-entity (essential-entity-bound 'person-at 'C-person))
        (company-prev-entity (entity-bound 'company-prev-at))
        (position-prev-entity (entity-bound 'position-prev-at))
        (company-entity (entity-bound 'company-at))
        (position-entity (entity-bound 'position-at))
        (reason-symbol 'REASSIGNMENT)
        (verb-span (binding 'verb-span))
        (tense-verb-word (first-word-bound 'verb-span))
        (tense-verb-symbol)
        (new-event)
        )
    (when (null position-entity)
          (error "~%when-j-appoint:cannot output appoint event.~%"))
    (not-an-antecedent position-entity)
    (setq tense-verb-symbol (intern tense-verb-word))
    (format *trace-output* "~%~S ~S" verb-span tense-verb-symbol)

    (setq new-event
                (assert-event
                 :predicate 'appoint
```

```
             :person person-entity
             :position position-entity
             :company company-entity
             :reason reason-symbol
             :verb verb-head
             :verb-tense verb-tense))
(when position-prev-entity
        (assert-event
         :predicate 'leave-job
         :person person-entity
         :position position-prev-entity
         :company company-entity
         :reason reason-symbol
         :verb verb-head
         :verb-tense verb-tense)))
```