NTT's QA Systems for NTCIR QAC-1

Yutaka Sasaki Hideki Isozaki Tsutomu Hirao Koji Kokuryou Eisaku Maeda
NTT Communication Science Laboratories, NTT Corp.
2-4, Hikari-dai, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan
{sasaki,isozaki,hirao,kokuryou,maeda}@cslab.kecl.ntt.co.jp

Abstract

In this paper, we describe two Question Answering systems that participated in the NTCIR QAC task. The first system, SAIQA-Ii, follows a standard word-distance approach. This system was designed to output the top five candidates and participated in TASK-1. The second system, SAIQA-Is, employs a logic-based approach. This system was designed to output only the most likely candidates and participated in TASK-1 through TASK-3. The evaluation results showed that SAIQA-Ii's word-distance approach was more suitable for TASK-1 than SAIQA-Is's; however, SAIQA-Is's logic-based approach was promising in TASK-2 and TASK-3.

Keywords: *Question Answering, NTCIR, QAC*

1 Introduction

In this paper, we describe two types of Question Answering systems that participated in the NTCIR QAC task. NTCIR QAC-1 was designed by organizers to evaluate three aspects of Question Answering systems, as described below.

TASK-1 evaluates the ability of QA systems to answer a question by returning at most five answers for each question. There is no penalty for returning wrong answers.

TASK-2 evaluates the ability of QA systems to show only correct answers. In this task, returning wrong answers causes low performance in precision.

TASK-3 evaluates the ability of QA systems to answer a series of questions given in the context of interaction.

The number of given questions were 200 in TASK-1. The same 200 questions were used in TASK-2. In TASK-3, 80 questions were provided, where a half of them were from the 200 questions and the other half were context questions.

The reason why we developed two systems is that we are very interested in investigating two distinct approaches: a word-distance approach and a logic-based approach.

The word-distance approach takes a *sequence-of-words* perspective with semantic constraints by a large-scale Japanese word taxonomy. On the other hand, the logic-based approach recognizes logical semantics of questions and articles and finds logical matches between questions and sentences.

The first system, *SAIQA-Ii*, follows the standard *word-distance* approach that was taken by our previous QA systems [11, 12]. It uses an SVM-based named entity recognizer [8] that attained F=90% for IREX-NE's GENERAL task [9].

The second system, *SAIQA-Is*, employs a logic-based approach, following the research on the FALCON system of Harabagiu et al. [1]. This system was designed to output only the most likely candidates.

In the following sections, we are going to explain our two QA systems, SAIQA-Ii and SAIQA-Is, in detail.

2 SAIQA-Ii

SAIQA-Ii follows a standard *word-distance* approach: once a question is given, each answer candidate in a retrieved paragraph is evaluated by using its distances to important words in the question.

Question Analysis First, the question is analyzed by using a set of hand-crafted rules to determine its expected answer type. We developed an answer type taxonomy based on named entity classes defined by IREX and *Goi-Taikei* (or *A Japanese Lexicon*) [3].

Both IREX's named entity classes and Goi-Taikei have problems. Although IREX defines only eight classes, we need more answer types. Moreover, IREX's ARTIFACT class covers not only product names but also laws, titles, and awards. We cannot use such a broad class as an answer type. LOCATION and ORGANIZATION are also confusing. Schools are sometimes used as LOCATION but at other time

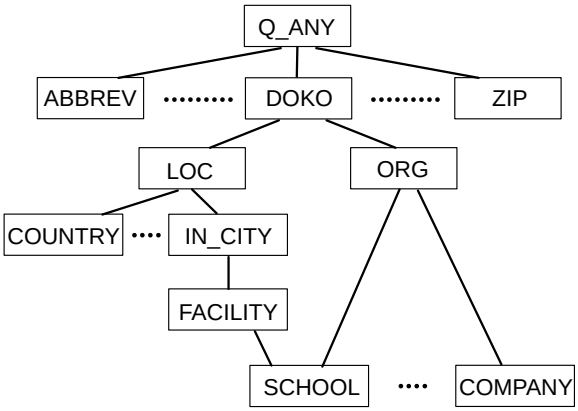

Figure 1. SAIQA-Ii's answer type taxonomy

as ORGANIZATION. When the system is asked for a name of a school, the system has to examine both LOCATION and ORGANIZATION. Furthermore, we should distinguish mountains, lakes, countries, cities, etc.

On the other hand, Goi-Taikei classifies words into three thousand semantic categories. A word is often associated with two or more semantic categories, and some of them are for rare cases. The semantic categories for rare cases lead to misclassification. In addition, the detailed taxonomy demands exact semantic matches, and correct answers do not necessarily satisfy the constraint. For instance, 'poison' and 'drug' are clearly distinguished, but some drugs can be used as poison. Therefore, we developed our own taxonomy for question answering. Figure 1 shows a part of this taxonomy. The taxonomy has more than 80 answer types, such as PERSON, MALE, FEMALE, LOCATION, ORGANIZATION, DATE, and LENGTH. Answer types form a DAG (directed-acyclic graph) instead of a list or a tree. We used the DAG structure because some answer types belong to two or more upper nodes. For instance, SCHOOL belongs to both FACILITY and ORGANIZATION. Each node in the DAG represents a named entity class or a constraint posed by the existence of a certain clue in the question.

The question analyzer also normalizes each word into a standard form. For instance, *'megane'* is a Japanese word for glasses. It is written in three different ways: Kanji (眼鏡), Katakana (メガネ), and Hiragana (めがね). Although they look quite different, they represent the same thing. Therefore, they are replaced by their standard form. Verbs are also replaced by their standard forms.

Paragraph Retrieval Second, the system ranks relevant paragraphs by using a TF-IDF-based OR search. Here, aliases, abbreviations, and antonyms are also searched. In order to answer a question such as 'Who is Clinton's wife?', the system searches not only *Clinton* and *wife* but also *husband* and *marry*.

In some cases, substrings of expected answers are also used in this search. When a question requests a URL, we can expect that the answer contains a word such as `http` or `www`. In such cases, these words are added to the query.

Answer Extraction Third, answer candidates are extracted from the retrieved paragraphs. For each question, the system uses an appropriate subset of the following mechanisms to get answer candidates.

1. Named Entity Recognition

   When a question expects a named entity defined for IREX [9] (ORGANIZATION, PERSON, LOCATION, ARTIFACT, DATE, TIME, MONEY, PERCENT), the system reads the output of an external SVM-based named entity recognizer [8].

2. Numerical Expression Recognition

   When a question expects other numerical expressions such as WEIGHT, LENGTH, and TEMPERATURE, the system reads the output of an external numerical expression recognizer that classifies numerical expressions into 50 classes.

3. Semantic Match for Long Noun Phrases

   When a question expects a technical term, long noun phrases are detected and then classified by using semantic categories of head words. For instance, 'mechishirin taisei oushoku budou kyuukin (メチシリン耐性黄色ブドウ球菌, Methicillin Resistant Staphylococcus Aureus)' is classified as a germ because of the head word 'kyuukin (micrococcus)'.

4. Semantic Match for Single Nouns

   When a question expects a name of an animal, the answer can be a proper noun or a common noun. In such cases, proper nouns and common nouns in the paragraphs are regarded as candidates. Their semantic categories are used to select inappropriate candidates.

5. Katakana Word Sequence Extraction

   Some technical terms and proper nouns are represented by a *katakana* word sequence. Therefore, katakana word sequences are important candidates.

6. Unknown Katakana Word Extraction

   Some technical terms and proper nouns are represented in a single *katakana* word that is unknown to our morphological analyzer. Therefore, unknown katakana words are also important candidates.

Table 1. SAIQA-Is's question types

	Category	Tag name	Example
1.	Person name	PERSON, AUTHOR, SINGER	*Smith, Kenzaburo Oe*
2.	Location name	LOCATION, COUNTRY, STATE, CITY, SEA, PLAIN, MOUNTAIN, ISLAND, RIVER, LAKE, PARK	*Mt. Fuji*
3.	Facility	FACILITY (location and artifact)	*Tokyo Tower*
4.	Organization name	ORGANIZATION, GOVERNMENT, COMPANY, SCHOOL	*SONY, Ministry of Health*
5.	Artifact	PRODUCT, TITLE (title of a work), DISEASE, LAW, PACT, SUBSTANCE, DRUG, EVENT, RAILWAY, ROAD, AWARD	*Mac,"Gone with the Wind" COP-3*
6.	Date and time	DATE, CENTURY, YEAR, DAY, MONTH TIME, HOUR, MINUTE, SECOND	*May 1st, Christmas 7 PM*
7.	Term, period	PERIOD, PERIOD_YEAR, PERIOD_MONTH, PERIOD_DAY PERIOD_HOUR, PERIOD_MINUTE, PERIOD_SECOND	*five days, 8.78 seconds*
8.	Numerical	MONEY, PERCENT, NPERSON, NLOCATION, NORGANIZATION, NPRODUCT, PHONE, ZIP, ADDRESS, LENGTH, SQUARE, VOLUME, WEIGHT, AGE, NTH POINT, SPEED, FREQUENCY, TEMPERATURE	*seven people two companies*
9.	Misc	PTITLE (person's title), EMAIL, URL, QUOTE, DISEASE, FOOD, LANGUAGE	*President O-157, Spanish*

7. Quoted String Extraction

   Since quotation marks are used to emphasize terms, quoted strings are also important candidates. However, a person's speech is also quoted. Therefore, the quoted string and neighbor words are examined semantically or syntactically to avoid selection of wrong candidates.

8. List Element Extraction

   When a question expects a country name, we can use the output of the named entity recognizer to get country names in the retrieved paragraphs. However, the output contains too many wrong candidates such as city names and state names. Instead of the output, we use a country name list to find only country names.

9. Description Extraction

   When a question expects a short description of a word, the system tries to find a neighbor word sequence. In order to answer a question such as 'What is DVD?', the system finds a pattern such as 'DVD (Digital Versatile Disk)', and 'Digital Versatile Disk' is returned as a candidate.

Answer Evaluation Fourth, each answer candidate is evaluated by using the distance to each important word in the given question. Here, we use the Hanning window [7] for the density calculation. This part also has a set of rules to reject unlikely candidates. If the question requests an island, answer candidates that do not look like a name of an island are rejected.

In QAC, detailed answers are preferred. When a question requests a person's name, the system should answer his/her full name unless the full name is not given in the relevant documents. Therefore, when a candidate looks too short, the system searches for a longer name and registers it for the final output.

3 SAIQA-Is

This section explains another QA system named SAIQA-Is. Figure 2 shows the block diagram of SAIQA-Is. An overview is as follows:

1. Receive a question Q.
2. Analyze the question Q and obtain question type QT.
3. Morphologically analyze Q and set the content words to keyword KW.
4. If the question is a subsequent question in a series of context questions, uses previous articles in F, otherwise retrieve top N articles w.r.t. the keywords and set them to F.
5. Analyze the question and obtain a logical form of question QL. If the question is a context question, resolve references in the question by using previous questions.
6. The costed unification finds the most plausible answers by unifying question logical form QL and article logical forms of the articles retrieved in Step 3.
7. If the stopping criteria is satisfied, output the answers; otherwise paraphrase the question, then go to Step 2.

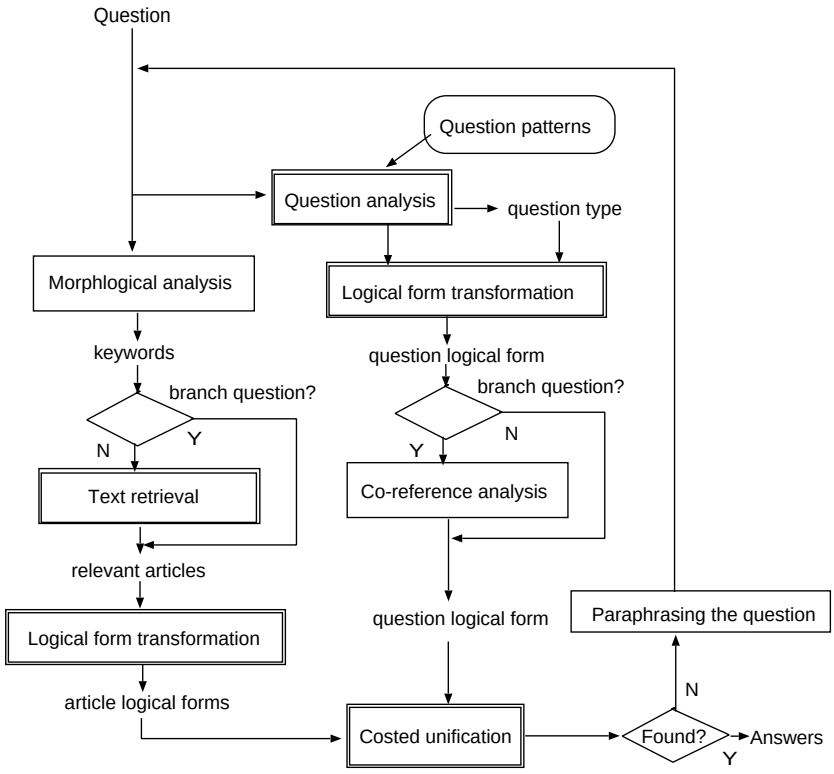

Figure 2. SAIQA-Is's block diagram

For TASK-1, SAIQA-Is finds 20 answers and returns the top five answers with the lower unification costs. For TASK-2 and TASK-3, it returns the first-ranked answers including answers with tie costs.

Question Analysis A question sentence is mapped to question types (Table 1) using a pattern-matching approach.

Question patterns, expressed in regular expressions, were compiled manually based on a human analysis of a large set of questions. Each question type has its own question patterns, and a question sentence is classified to a question type if one of the question patterns matches the question.

For example, the question type of the question "Where is Key West?"will be determined as LOCATION by using the pattern "Where is → LOCATION". However, if the question is "Which state is Key West located in?" will be determined as a STATE question type by using the pattern "Which state is → STATE".

Text Retrieval Let KW be keywords in a question sentence. The text retrieval module just finds the top N articles from two years of Mainich Newspaper articles by using the keywords KW. An index, which is an association list from all words to the articles where the words appears, was build at the system development stage.

The weight of a word $weight(w)$ is computed as follows:

$$weight(w) \stackrel{\text{def}}{=} \frac{1}{\log(df+2)}, \quad (1)$$

where df is the number of articles in which the word w occurs.

The score of an article $fscore(d)$ of an article d is:

$$fscore(d) \stackrel{\text{def}}{=} \sum_{w \in KW \cap words(d)} weight(w), \quad (2)$$

where $words(d)$ returns the set of all words in d. The top N articles are returned as the result of text retrieval according to the $fscore$.

Logical Form A logical form is an expression that consists of *atomic form (atoms)* combined with *the logical conjunction* and *the logical disjunction*. The logical conjunction is represented shown by the comma (,) and the logical disjunction is as semicolon (;).

An atom is one of the forms:

- *NE*(Var:'$word$')
- *question_type*(Var:Z)
- '*semantic_ category*'(Var:'*word*')

- w(Var:'$word$')
- R(Var,Var)

For simplicity in implementation, variable names start with X if the logical form is made from an article and starts with Y from a question. Z is reserved as the variable to which answers are to be unified. Predicate R represents binary relations between two variables and means that two words with the variables have a (syntactic) dependency in a sentence. Semantic categories are given according to the semantic attribute system of *Goi-Taikei* [3].

Note that usually the logical conjunction has stronger connectivity than disjunction, but in our logical form, disjunction is stronger than conjunction. Note also that in our form, we do not distinguish verbs; we treat verbs the same as nouns and adjectives. The relations between a verb and a subject or an object are represented by Predicate R. This is because important case roles, such as an agent, tend to be abbreviated in Japanese even though the sentence appears a in newspaper article.

Logical Form Transformation Given a sentence s, the logical form transformation generates a logical form of s.

1. If s is from an article, the prefix of variable Var is fixed to X; otherwise if s is a question, the prefix is fixed to Y.
2. Separate a sentence s into *bunsetsu* (or Japanese phrases).
3. Find unit sentences from a sequence of bunsetsu.
4. Separate each bunsetsu into words.
5. Find named entities in word-segmented s.
6. Morphologically analysis s and find part-of-speech and semantics categories of each word in s.
7. Decide dependencies between words.
8. If a word w is a content word with a semantics category SC, create an atom of the form 'SC'(Var:'w'), where Var is a fresh variable. If SC has several semantic categories, create 'SC_1'(Var:'w') ;...; 'SC_n'(Var:'w').
9. If a word w is a named entity NE, create an atom of the form 'ne'(Var:'w'), where Var is a fresh variable. If NE has several named entities, create 'NE_1'(Var:'w') ;...; 'NE_n'(Var:'w').
10. If a word w is a question word, such as who, $where$, $how - many$, create an atom of the form 'QT'(Z:'w'), where Var is a fresh variable and QT is the question type. If QT has several question types, create 'QT_1'(Var:'w') ;...; 'QT_n'(Z:'w').
11. If a word w is a content word without semantics categories, create an atom of the form w(Var:'w'), where Var is a fresh variable.
12. Add R(Var_1,Var_2) if there is a dependency between the words with Var_1 and Var_2.

For example, "Where is the capital city of Japan?" is represented as the following logical form.

```
COUNTRY(Y1:'Japan'), R(Y1,Y2),
'city'(Y2:'capital city'),
LOCATION(Y3:Z); ORGANIZATION(Y3:Z),
R(Y2,Y3)
```

Since the word "where" (*doko*) indicates both location and organization in Japanese, the logical form contains LOCATION and ORGANIZATION with variable Z.

For TASK-2, dependency information in a parallel expression is represented while keeping several possibilities. If there is an expression, "A, B, C ", there is a dependency not only A to B but also A to C. This is because it is preferable that A, B, C have the same cost if they are extracted as answers. This makes it possible to list up multiple answers to a question.

Distance in a Question Logical Form As a preliminary for describing the costed unification procedure, we present the definition of a distance in a question logical form. The definition is based on the assumption that the nearer words are to question words, the more important the words are in a question.

The distance of word w is defined as follows:

1. If the word is with variable Z, its distance is 1.
2. The distance of word w is the minimum number of relations R(V_1,V_2) that are needed to transitively reach from the variable with w to the variable with Z.

Costed Unification Ideally, answers are the words that are unified to answer variable Z during exact unification of question logical form and article logical forms. It is, however, very rare that a real answer is described with the same semantic structure as the semantic structure of the question. For example, there is a case that "When will DELL release a new PC?" should be matched with "The release date of DELL's new PC is next Friday". Therefore, the unification allows some incomplete matching controlled by a unification cost. This approach is based on *interpretation as abduction* [2].

Table 2. Scores of SAIQA-Ii and SAIQA-Is

	TASK-1 MRR	TASK-2 F	TASK-3 AFM	TASK-3 F
SAIQA-Ii	⋆⋆⋆⋆	—	—	—
SAIQA-Is	⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆⋆	⋆⋆⋆⋆

1. Separate a question logical form QL into clauses at the positions of comma and set the clauses to QC.
2. For each article logical form AL, do the following steps.
   (a) In the following steps, find a matching with AL for all c in QC.
   (b) Separate c into atoms P with semicolons.
   (c) For each p in P, match p with an atom in AL in the following manner.
      i. Find an atom in AL with the same predicate as p.
      ii. Variables X_i can match any variables Y_j. If it matches, bind X_j to Y_i.
      iii. Variable Z can be bound to terms.
      iv. Two atoms with the same semantic category with distinct words match with some costs.
   (d) If there is nothing that matches with p, add cost $100/d$, where d is the distance of p in QL. Then continue a unification process.
   (e) After trying all matching with atoms in QC, output terms bound to Z with the lowest (or lower) cost(s).

Paraphrasing SAIQA-Is paraphrases a question if there is no answer found in TASK-2,3 or there is room to return more answers in TASK-1. It has a database of paraphrases that maps an expression to another similar expression. For example, "World Cup" is described as "W-Cup" in newspaper articles. Newspaper correspondents tends to use shorter terms if there is a way to express the same term short. Therefore, paraphrasing has an effect of adjusting a question to the writing style of newspaper articles as well as expanding the expression.

Co-reference Analysis SAIQA-Is gives co-reference tags along with NE tags. Co-reference tag names are defined as REF+NE tag names. For instance, REFPERSON indicates a reference to a person name. Examples of REFPERSON include *he, she, they, and the manager*.

The current implementation of SAIQA-Is's co-reference analysis is very naive. It just links a term with a REF+X tag to the nearest precedent NE tag X. For examples, it links a word with REFPERSON to the nearest word with the PERSON tag.

Co-reference information is represented using variables in a logical form. That is, variables of words that are linked by a co-reference relation are unified to an identical variable. By the co-reference analysis, "PERSON(X_1:'Smith'),..., REFPERSON(X_2:'he'),R(X_2,X_3)" is changed to "PERSON(X_1:'Smith'),..., REFPERSON(X_1:'he'), R(X_1,X_3)".

4 Results

Table 2 shows the evaluation results of our QA systems. Stars ($\star$) rate the performance that each system achieved for each task. Five stars indicates excellent performance and one star means poor performance. MRR stands for Mean Reciprocal Rank, which gives an average of points for all answers, where $1/n$ point is given for each correct answer at the rank n. F means F-measure, which is a balanced measure of recall and precision. AFM is an average of F-measures.

The evaluation results showed that SAIQA-Ii's word-distance approach was more suitable for TASK-1 than SAIQA-Is's; however, SAIQA-Is's logic-based approach is promising in TASK-2 and TASK-3. Furthermore, TASK-3 was more difficult to complete because of the many omissions and abbreviations in the context questions.

In TASK-3, it was interesting to us that five context questions were correctly answered even though their main questions were not correctly answered.

5 Related Work

SAIQA-Ii's approach is a common approach in the QA community. Most of the systems participating in TREC QA-Tracks answered questions based on English news wires by using a word-distance measure. Some systems also used parse information to reflect more appropriate word-distances.

SAIQA-Is's approach is based on *Interpretation as Abduction* [2]. The idea of this approach is that interpretation is regarded as a process of abductively adding missing links between words with costs as well as relaxing condition of links with costs. FALCON QA system [1] uses this approach in an English QA system and marked excellent results in the TREC QA-Tracks. However, to our best knowledge, there is no Japanese QA system that uses an "Interpretation as Abduction" approach. From our experience, we feel that because the Japanese language is grammatically looser than English, it is more difficult to decide costs or penalties of ambiguous unification between a question logical form and an article logical form.

Most of the participants at the TREC QA Tracks did not employ a machine learning approach, although some systems did [4, 5, 6]. The first two papers used the maximum entropy method for question analysis, and the third paper applied a method of answer extraction. Another research effort that adopted machine leaning for answer extraction was presented by [10]. Suzuki et al. [13] applied Support Vector Machines (SVM) to the answer selection of QA. We believe the costed unification could also be augmented by a machine learning method.

6 Concluding Remarks

Our two systems showed good performance in each task for which we designed them. Through QAC tasks, it was confirmed that SAIQA-Ii's word-distance based approach is suitable for TASK-1 and that SAIQA-Is's logic-based approach is appropriate for TASK-2 and TASK-3.

Our future work includes applying Question Biased Text Summarization (QBTS) [7] to answer "why" and "how" questions and applying machine learning methods to the next QAC tasks.

Acknowledgment

We would like to thank all members of the Knowledge Processing Group, NTT Communication Science Laboratories.

References

[1] S. Harabagiu, D. Moldovan, R. Mihalcea M. Pasca, R. Bunescu, M. Surdeanu, R. G irju, V. Rus, and P. Morarescu, Falcon: Boosting knowledge for answer engines, Proc. of Ninth Text REtrieval Conference (TREC 9), pp. 479–488, 2000.

[2] Jerry R. Hobbs, Mark E. Stickel, Douglas Appelt, and Paul Martin, Interpretation as Abduction, Technical Report 499, AI Center, SRI International, 1990.

[3] S. Ikehara, M. Miyazaki, S. Shirai, A. Yokoo, H. Nakaiwa, K. Ogura, Y. Oyama, and Y. Hayashi, *Goi-Taikei — A Japanese Lexicon*, volume 1, Iwanami Publishing, 1997. (in Japanese)

[4] Abraham Ittycheriah, Martin Franz, Wei-Jing Zhu, and Adwait Ratnaparkhi, IBM's Statistical Question Answering System, *Proc. of TREC-9*, 2000.

[5] Abraham Ittycheriah, Martin Franz, Wei-Jing Zhu, and Adwait Ratnaparkhi, Question Answering Using Maximum-Entropy Components, *Proc. of NAACL 2001*, 2001.

[6] Abraham Ittycheriah, Martin Franz, and Salim Roukos, IBM's Statistical Question Answering System – TREC-10, *Proc. of TREC-10*, 2001.

[7] Tsutomu Hirao, Yutaka Sasaki, and Hideki Isozaki, An Extrinsic Evaluation for Question-Biased Text Summarization on QA Tasks, *Proc. of NAACL-2001 Summarization Workshop*, 2001.

[8] Hideki Isozaki and Hideto Kazawa, Efficient support vector classifiers for named entity recognition, *Proc. of COLING-2002*, 2002.

[9] Satoshi Sekine and Yoshio Eriguchi, Japanese named entity extraction evaluation — analysis of results — , *Proc. of 18th International Conference on Computational Linguistics*, pp. 1106–1110, 2000.

[10] Hwee T. Ng, Jennifer L. P. Kwan, and Yiyuan Xia, Question Answering Using a Large Text Database: A Machine Learning Approach, *Proc. of EMNLP 2001*, pp. 67–73, 2001.

[11] Yutaka Sasaki, Hideki Isozaki, Hirotoshi Taira, Keiichi Hirota, Hideto Kazawa, Tsutomu Hirao, Hiroyuki Nakajima, and Tsuneaki Kato, An Evaluation and Comparison of Japanese Question Answering Systems, Technical Report of IEICE, NLC-2000-24, 2000. (in Japanese)

[12] Yutaka Sasaki, Hideki Isozaki, Hirotoshi Taira, Tsutomu Hirao, Hideto Kazawa, Jun Suzuki, Kouji Kokuryou, and Eisaku Maeda, SAIQA: A Japanese QA System based on a Large-Scale Corpus, IPSJ SIG notes FI-2001-9-11, 2001. (in Japanese)

[13] Jun Suzuki, Yutaka Sasaki, and Eisaku Maeda, SVM Answer Selection for Open-Domain Question Answering, *Proc. of COLING-2002*, 2002.

Appendix

This section shows SAIQA-Is's example process of generating the logical form of a question sentence.
Question: "５月に新しい高性能パソコンを発売した会社はどこですか？" (What company released a new high performance PC in May?)

```
*Sentence separation:
1-1. ５月に新しい高性能パソコンを発売した会社はどこですか？

*Bunsetsu analysis:
1-1. ５月に
1-2. 新しい高性能パソコンを
1-3. 発売した
1-4. 会社は
1-5. どこですか？

*Unit sentence analysis:
1-1. ５月に
1-1. 新しい高性能パソコンを
1-1@ 発売した
1-2. 会社は
1-2. どこですか？

*Word separation:
1-1. ｜５月｜に｜
1-1. ｜新しい｜高性能｜パソコン｜を｜
1-1@ ｜発売｜した｜
1-2. ｜会社｜は｜
1-2. ｜どこ｜です｜か｜?|

*NE tagging:
1-1. ｜<５月>DATE｜に｜
1-1. ｜新しい｜高性能｜パソコン｜を｜
1-1@ ｜発売｜した｜
1-2. ｜会社｜は｜
1-2. ｜どこ｜です｜か｜?|

*Morphological analysis:
1-1. ｜<５月>DATE｜に｜
1-1. ｜<新しい (3106)>｜<高性能 (1250[2492,2502])>｜<パソコン (1100[971])>｜を｜
1-1@ ｜<発売 (1220[1900])>｜した｜
1-2. ｜<会社 (1100[374,428])>｜は｜
1-2. ｜どこ｜です｜か｜<?>｜

*Logical form:
1-1. DATE(X_1_1_1_1:’５月’),
1-1. ( ’物性 [2492]’(X_1_1_2_2:’高性能’); ’能力 [2502]’(X_1_1_2_2:’高性能’) ),
     R(X_1_1_2_2,X_1_1_2_3), ’コンピュータ [971]’(X_1_1_2_3:’パソコン’),
1-1@ ’売り [1900]’(X_1_1_3_1:’発売’),
     R(X_1_1_1_1,X_1_1_3_1), R(X_1_1_2_3,X_1_1_3_1),
1-2. COMPANY(X_1_2_5_1:Z); LOCATION(X_1_2_5_1:Z), R(X_1_1_3_1,X_1_2_5_1)
```