Proceedings of NTCIR-4, Tokyo, April 2003 - June 2004

NTT's Question Answering System for NTCIR QAC2

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

Abstract

In order to retrieve best documents for finding answers, we developed a robust proximity search engine. It efficiently finds relevant passages. In addition, our search engine has two disjunction operators: `or` *and* `or2`*. The former works just like addition, whereas the latter works just like logical disjunction. The operator* `or2` *is used to introduce synonyms and antonyms of a query term. The search engine also outputs the distribution of query terms in retrieved documents. Our QA system based on the search engine showed a good performance for QAC2 task1: MRR = 0.607 and Top 5 = 0.738.*

Keywords*: question answering, information retrieval*

1 Introduction

Our question answering system SAIQA[1] follows a standard architecture for open-domain question answering. First, the system analyzes a given question and then determines an expected answer type. Second, relevant documents (or passages) are retrieved. Third, answer candidates are extracted and ranked. The system employs ALTJAWS, a morphological analyzer based on Nihongo Goi-Taikei [5].

Following QAC1 [11], we discovered various problems with SAIQA-Ii.

- SAIQA-Ii's rule-based question analyzer employed a simple right-to-left longest match. It turned out that the analyzer fails when unexpected adverbial expressions are inserted into a question. In addition, the number of answer types was not sufficient.
- SAIQA-Ii's search engine ranked paragraphs by using a variant of the standard TF-IDF method. It turned out that short passages that contain important query terms are sometimes split into different paragraphs. For instance, a news article's headline is a paragraph, and the next sentence is another paragraph. Because of this, the search engine sometimes failed to find a very important passage. Another problem is the length of a paragraph: since longer paragraphs tend to contain more query terms, they were preferred to shorter paragraphs. Introduction of synonyms and antonyms for the search caused yet another problem, because a paragraph that contains only one original query term and its synonyms sometimes obtains a better score than another paragraph that contains two or more different query terms.
- SAIQA-Ii's answer evaluation module employed a Hanning window for density calculation. The density is calculated by using the number of words between the candidate and query terms found in the retrieved document. If an answer candidate is next to a long noun phrase such as an organization name, the distance between the candidate and query terms are overestimated. Verbs are often followed by auxiliary verbs (*Jodoushi*) and suffix words (*Setsubigo*). This also increases the distance unintentionally.

Through examining these problems, we have developed a new system, *SAIQA-QAC2*. First, we present its search engine, which uses a new passage evaluation algorithm. We, then, describe other improvements.

2 Proximity-based Document Retrieval Engine

Since it turned out that paragraphs are unreliable, we gave up using the paragraph retrieval system. It is well known that the dense distribution of query terms is a good hint for finding answers. Many QA systems use fixed-size windows such as 300 words or three sentences. However, if the window size is too small, the

[1]SAIQA stands for 'System for Advanced Interactive Question Answering'.

window will not cover many query terms, and if the window size is too large, passage scores will not reflect the density distribution. Therefore, we do not want to fix the window size.

During our research, we came upon Sadakane's proximity search algorithm [9]. According to this method, each document is evaluated by the length of the shortest passage that contains all query terms. (Shorter is better.)

However, it turned out that the system often failed to find relevant documents in the following cases.

- No documents contain all query terms.
- One or more query terms are far from other query terms.

Harabagiu et al.[2] used various feedback methods to avoid similar problems. However, we do not employ feedback because it complicates the system's behavior.

Here, we decided to use disjunctions of query terms instead of their conjunctions. If one asks "Who is the President of the US?," the question is converted to a search query `President or US` instead of `President and US`. The disjunction operator `or` works just like addition of IDFs. In the first pass, documents are ranked by the sum of IDFs. In the second pass, documents are reranked by best passage scores.

The second pass is similar to the answer evaluation process. At this stage, however, we do not have answer candidates, therefore, we will have to evaluate all possible passages. Suppose that a document D contains N_D words. Then, the number of passages in a document D is $O({N_D}^2)$. In order to evaluate these passages, we have to enumerate all query terms in the passages. Therefore, a naive implementation will be $O({N_D}^3)$. Such a system would be very slow because N_D can be very large. In this paper, we present a more efficient algorithm to get the best passage score in the document.

Another problem is that of *paraphrasing.* Since ALTJAWS' dictionary has normal forms of nouns, "girisha" (Greece) is normalized to "girishia." The Japanese word "megane" (eyeglasses) in hiragana is normalized to "megane" in Chinese characters. However, this normal-form data is far from complete. For instance, "bei" and "beikoku" are alternative expressions of "amerika" or the United States of America, but ALTJAWS does not have their normal forms. In addition, antonyms are useful for finding paraphrases. (e.g., Ann is the wife of Bob = Bob is the husband of Ann.) Therefore, we want to add related terms to a query, but it is sometimes detrimental to add related terms. Therefore, we add an extra logical operator `or2` to indicate alternatives.

2.1 An efficient document-scanning algorithm

Here, we present a more efficient algorithm that is a modification of Sadakane's algorithm. This algorithm scans a document and finds the best-scored passage.

A passage is represented by a pair of integers $[l, r]$ where l is the location of the first word of the passage and r is the location of the last word. Passage p's score is denoted $S(p)$. If passage p_0 is contained in passage p_1 and p_0 is not equal to p_1, we denote this as $p_0 \sqsubset p_1$. Query terms are denoted $q_1, \ldots, q_k$. $w[q_j]$ is a predefined non-negative score of q_j. In addition, $Q(p)$ is the set of all query words that appear in passage p. $\mathtt{PL}[q_j][h]$ stands for the h-th position of q_j in D.

Since shorter is better, we assume the following monotonicity for efficiency.

If $p_0 \sqsubset p_1$ and $Q(p_0) = Q(p_1)$ holds, $S(p_0) > S(p_1)$ holds.

For instance, the next definition satisfies this assumption for $\beta > 0$.

$$S([l, r]) = \exp(-\beta(r - l)) \sum_{q_j \in Q([l,r])} w[q_j]$$

Here, we call this "Decayed IDF" or DIDF.

Passage p is *Q-minimal* if and only if there is no passage p' that satisfies $p' \sqsubset p$ and $Q(p') = Q(p)$. Our algorithm finds *Q-minimal* passages in $[1, N]$.

The algorithm scans from left to right, with the initial window being the minimum passage that covers the leftmost positions of all query terms that appear in the document D. Suppose D has $k_D(= |Q([1, N])|)$ query terms.

- For each query term $q_j \in Q([1, N])$, obtain its position list $\mathtt{PL}[q_j]$.
- Merge and sort all $\mathtt{PL}[q_j]$s to make a single list `all`. The number of elements in `all` is denoted n_D. The i-th word in `all` is denoted `allQ[i]` and its position is denoted `allP[i]`.

Then, the window moves right. Figure 1 shows an example. In this example, $Q([1, N])$ is $\{a, b, c, d, e\}$ and non-query terms are disregarded. The initial position of the window is indicated by "( )" in Step 0. In Step 1, the leftmost element "a" is removed and the next position of "a" is employed instead. In Step 2, the leftmost element "c" is removed and the next position of "c" is employed. In this way, the window moves right until it reaches the end of `all`.

Each location of the window contains different passages. However, we consider only *prefix passages* that are prefixes of the window, with Table 1 shows. Since other passages in the window will be covered later, they are outside the scope of consideration for now. The number of Q-minimal prefix passages is at most k_D.

Step	`all`													
0	(**a**	c	**a**	d	b	a	d	e)	c	a	d	b	c	b
1	a	(**c**	a	d	b	a	d	e)	**c**	a	d	b	c	b
2	a	c	(**a**	d	b	**a**	d	e	c)	a	d	b	c	b
3	a	c	a	(**d**	b	a	**d**	e	c)	a	d	b	c	b
4	a	c	a	d	(**b**	a	d	e	c)	a	d	**b**	c	b
5	a	c	a	d	b	(**a**	d	e	c	**a**	d	b)	c	b
6	a	c	a	d	b	a	(**d**	e	c	a	**d**	b)	c	b
7	a	c	a	d	b	a	d	(**e**	c	a	d	b)	c	b

"( )" indicates the location of the window.

Figure 1. Movement of the window

Table 1. Prefix passages in a window

	query terms	Q-minimal	last elem.
Window	a c d b c d e		
Prefix passages	**a**	Yes	**a**
	a **c**	Yes	**c**
	a c **d**	Yes	**d**
	a c d **b**	Yes	**b**
	a **c** d b **c**	No	
	a c **d** b c **d**	No	
	a c d b c d **e**	Yes	**e**

Suppose the number is larger than k_D. Then, prefix passages P1 and P2 have the same query term q_j at the end of the passages because D has only k_D different query terms. Without loss of generality, we can assume that P1 is a prefix of P2. Since P2 has two q_js, the last element of P2 is redundant. That is, P2 is not Q-minimal. Therefore, the number of Q-minimal prefix passages is at most k_D. □

Accordingly, each prefix passage in a window can be identified by its last element, a fact we can use for efficient updates.

Figure 2 shows the proximity search algotithm whose time complexity to process D is $O(k_D n_D + n_D \log n_D)$, which is less than $O({n_D}^2)$. We can make it faster by using Q-minimality. In Table 2, we do not have to calculate scores of long passages that start with"a c d a" because it violates Q-minimality. This algorithm was incorporated into LISTA, an XML-based search engine [3].

2.2 Alternatives of a query term

Some query terms have several alternatives. For example, in the search to answer the question,

Who is the father of Miu Sakamoto?

we can use various alternative terms such as *child*, *daughter*, *son*, *first-son*, *second-son*, and *first-*

```
# Initialization of the window
win = []; # window
DS = 0; # document score
foreach q in Q([1,N]) {
   insert(PL[q][0],win);
}
# The window moves right.
for (L := 0; L <= n; L := L+1) {
  ql := allQ[L]; # leftmost term
  pl := allP[L]; # its position
  WS := 0; # sum of weights
  foreach R in (win) { # passage
     qr := allQ[R];
     pr := allP[R];
     WS := WS + w[qr];
     S  := WS * exp(-beta*(pr-pl));
     if (DS < S) { DS := S; }
  }
  shift(win);
  shift(PL[ql]);
  if (PL[ql] != []) {
    insert(PL[ql][0],win);
  }
}
```

`insert(e,L)` inserts an integer `e` into the integer list `L` that is sorted in ascending order.
`shift(L)` removes the first element of `L`.
`DS` is the best passage score in the document.

Figure 2. An algorithm for the proximity search engine

Table 2. More efficient evaluation
We do not have to evaluate prefix passages that start with "a c d a."

	query terms	Q-minimal	last elem.
Window	a c d a c d e		
Prefix passages	**a**	Yes	**a**
	a **c**	Yes	**c**
	a c **d**	Yes	**d**
	a c d **a**	No	
	a c d **a** c	No	
	a c d **a** c d	No	
	a c d **a** c d e	No	**e**

daughter. Introduction of these terms is sometimes detrimental because articles that have nothing to do with Miu Sakamoto may obtain good scores. Therefore, we introduce a new operator `or2`.

Figure 3 shows the output of the engine for a question:

Who is the wife of President Clinton?

Each line corresponds to a document and has four elements. The first element is the best passage score (DS). The second element is the document's name. The third element is a list of query terms, their IDFs, and their positions. The forth element is a list of best passages and their scores.

The question analyzer extracts three query terms: `wife`, `president`, and `clinton`. Then, the QA system adds `husband` to `wife` as an alternative query term. Therefore, the QA system sends a query "`(wife or2 husband) or president or clinton`" to the search engine. The new disjunction `A or2 B` identifies `B` with `A` and merges `PL[B]` into `PL[A]`. Extended A's IDF is set to the minimum of A's IDF and B's IDF.

If we use "`wife or husband`," a passage such as "`Your wife/husband is ...`" obtains a good score because this passage contains two query terms: `wife` and `husband`. On the other hand, if we use "`wife or2 husband`," the system identifies `husband` with `wife`. Therefore, this passage contains only one query term: `wife`, and the repetition of a query term does not increase the passage score at all. In this way, `or2` reduces problems caused by introducing alternative terms.

According to Figure 3, the best-scored document 990804244 contains `clinton`, `president`, and `husband`.

3 Other improvements

3.1 Question analysis

SAIQA-Ii used a right-to-left longest pattern matcher. It fails when unexpected adverbial expressions are inserted. In the next question, the matcher failed because an adverb *oyoso* (approximately) was unexpected.

Rule: daigaku wa ikutsu → Number of Universities
Question: Nihon ni daigaku wa *oyoso* ikutsu arimasu ka?
(Approximately how many universities are there in Japan?)

Although some adverbial expressions are long, this problem can be solved when we use a parser or a dependency analyzer. Our abduction-based QA system, SAIQA-Is [10], used a Japanese-to-English translation system ALT-J/E [6] for parsing. Here, we have employed a simpler approach. We implemented a new question analyzer that finds the question word *ikutsu* (how many) and the focus word *daigaku* (university) separately. Now, adverbial expressions do not affect question analysis.

In addition, we increased the number of answer types: SAIQA-Ii had about 80 types, whereas SAIQA-QAC2 has 189.

3.2 Answer extraction

Although we added about one hundred answer types, we changed Named Entity Recognizers very little. We used an SVM-based Named Entity Recognizer [7] that achieved F = 90% for IREX general task [12]. Although SVM is often criticized for its inefficiency, this recognizer runs at 62 KB/sec on a 2.8-GHz Pentium 4 Linux PC. We did not change this module at all. This recognizer finds only eight IREX named entities: ORGANIZATION, PERSON, LOCATION, ARTIFACT, DATE, TIME, MONEY and PERCENT. As for numerical expressions, we wrote 300+ rules for 53 answer types. This recognizer was slightly improved.

We also use name lists (or gazetteers). For instance, prefectures in Japan can be enumerated, and such lists are useful to improve accuracy for some answer types. However, we used only three lists (COUNTRY, PREFECTURE, and STATE).

Candidates for other answer types are dynamically generated. For instance, MMDD (month & day) is a subclass of DATE. MMDD inherits answer candidates from DATE but rejects candidates that do not include "month" or "day." Answer candidates for BIRD, REPTILE, etc. are extracted by using Nihongo Goi-Taikei's word senses. Goi-Taikei classifies 350,000 words into about 3,000 categories.

```
score   document  term(IDF),positions passage[score]
16478 990804244 clinton(1.91),83:president(1.43),7,85:husband(1.70),82,282 82-85[16478],...
16478 990717256 clinton(1.91),241,459,603:president(1.43),242:husband(1.70),239 239-242[16478],
16071 980316J1TYEUB0400010 clinton(1.91),3,25,175,395:president(1.43),5,27,43,88,116,...
15911 980819276 clinton(1.91),17,62:president(1.43),19,63,105,181,211,247,266:husband(1.70),..
```

Question: Who is the wife of President Clinton?

Figure 3. Example output of the proximity search engine

Candidates are obtained by using several methods such as:

- Inheritance: Since it is not clear whether a hospital is an organization or a location, we introduce a new answer type, HOSPITAL. The next rule means that HOSPITAL inherits candidates from ORGANIZATION and LOCATION.

  ```
  HOSPITAL -> ORGANIZATION LOCATION
  ```

- Complex noun phrases: For instance, RULE's candidates are complex noun phrases that end with a word whose sense is "statute," "regulation," "law," or "treaty" according to Nihongo Goi-Taikei.

- Quoted expressions: The next sentence indicates that "Yukiguni" is a novel.

  His novel "Yukiguni" was ...

- Unknown words: It is difficult to classify unknown foreign words such as "farfalle." These words are accepted by FOOD or INSECT as candidates.

Then, a filter is used to reject inappropriate candidates. For instance, the next rule means that any candidate that ends with the suffix "iin" (clinic) is accepted as HOSPITAL.

```
iin   :   HOSPITAL
```

Since HOSPITAL is a subclass of FACILITY, a clinic is also accepted as a candidate of FACILITY. However, it is not accepted as a candidate of AQUARIUM, which is not an upper class of HOSPITAL. Figure 4 shows this part of the answer type taxonomy.

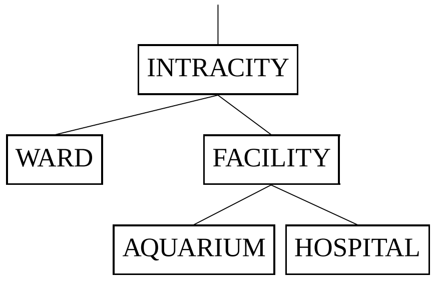


Figure 4. A part of the answer type taxonomy

3.3 Answer ranking

We tried to keep our scoring function as simple as possible to make the system comprehensible. Candidate c's score is given by a weighted Hanning window function[2] [4] defined as follows:

$$\text{score}(c) = \sum_{q_j \in Q} w[q_j] H(d(c, q_j)),$$

where $d(c, q_j)$ is the distance between c and the nearest position of q_j, and

$$H(d) = \begin{cases} \frac{1}{2}(\cos(\pi\, d/W) + 1) & \text{if } 0 \leq d \leq W, \\ 0 & \text{otherwise.} \end{cases}$$

SAIQA-Ii used the number of words as the distance, whereas SAIQA-QAC2 used the number of *bunsetsu*s. A bunsetsu is a basic linguistic unit in the Japanese language. Several words are contained in a bunsetsu. The bunsetsu distance seems more natural than the word distance. It is not clear what is a word in Japanese because no inter-word space is used. On the other hand, bunsetsu is relatively clear and corresponds to semantic roles.

Some articles look like entries of a dictionary; that is, an article's body is a description of a headword given in the headline. The headword is rarely mentioned in the body because it is obvious. Suppose the headword is a correct answer to a question and query terms in the body are far from the headline. The answer will not obtain a good score. However, we can detect such articles by using simple patterns. For these articles, the distances between answer candidates and query terms are assumed to be zero.

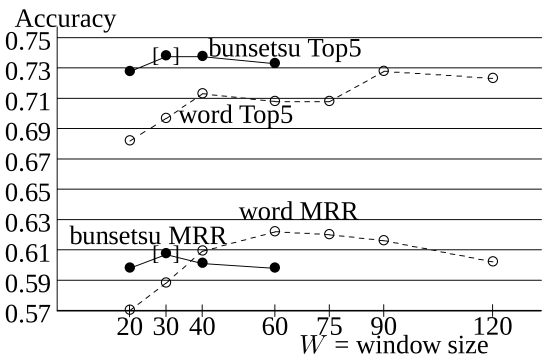
4 Results

Table 3 shows the stepwise evaluation of the system. According to this table, question analysis is essential to question answering. We observed a similar tendency when we analyzed our QAC1 system, leading us to greatly improve the question analysis module. Table 4 shows causes of failure of some questions

[2]Since our search engine also uses a distance-based score, it seems redundant to use another distance-based scoring function. This is our future work.

Table 3. Stepwise evaluation of the system

Ans. Type	Search	Answer rank	#Qs
Correct	OK	1	96
Correct	OK	2–5	44
Correct	OK	Failed	25
Correct	Failed	Failed	1
Wrong	OK	1	4
Wrong	OK	Failed	25



bunsetsu = distance is the number of bunsetsus
word = distance is the number of words
[●] = submitted version

Figure 5. Effects of distance unit and window size

that SAIQA-QAC2 could not answer. This table also shows that question analysis is essential.

Table 6 shows the performance of DIDF. #Qs is the number of answerable questions (lenient evaluation) [3]. For 76% of the questions, the first ranked documents contained correct answers.

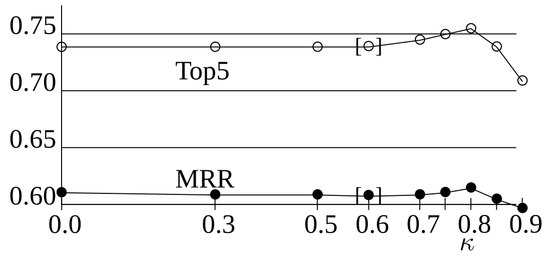
Table 5 compares Top 5 scores and MRR scores of QA systems based on different retrieval systems. when we use only top 3 or top 5 documents in a simplified version of the QA system. According to this table, DIDF performs better than MultiText [1] or IDF. We will compare DIDF with Okapi BM25 [8] and MultiText more extensively in the full paper.

Figure 5 compares bunsetsu distance and word distance for answer evaluation in terms of MRR. According to this graph, word distance gives better MRR scores while bunsetsu distance gives better Top 5 scores. When we used word distance, $W = 60$ yielded MRR = 0.622.

[3]Numbers in the old version of this manuscript was incorrect. It was evaluated by a human subject, but alternative correct answers were not checked. Here, the evaluation was automatically done by a program.

Table 4. Failure causes

006	Paraphrase or distance
007	Abbreviation
008	Anaphora to a part of an address
017	Question analysis (Wrong type for “privilege”)
024	Question analysis (Wrong type for “equipment”)
025	Question analysis (Wrong type for “predecessor”)
026	Question analysis (“Nickname” was removed)
035	Question analysis (Ambiguity of “what”)
037	Question analysis (Wrong type for “record”)
051	Answer extraction (Strip)
063	Paraphrase or distance
067	Question analysis (Ambiguity of “what”)
068	Answer extraction (An unfamiliar fish name)
075	Paraphrase (die)
078	Answer extraction (Unusual “average lifespan”)
098	Question analysis (Wrong type for “class”)
119	Question analysis (Particle *towa*)
129	Morphological analysis (Wrong part-of-speech tag)
133	Question analysis (Relative noun *koto*)
138	Named entity recognition (Chinese location name)
148	Answer extraction (3-liter car)
153	Question analysis (Unusual usage of *doko*)
154	Question analysis (Relative noun *koto*)
155	Misleading term (Character)
159	Answer evaluation (New laws)
161	Named entity recognition (Eva Braun)
162	Question analysis (Particle *towa*)
172	Question analysis (Ambiguous word *tokoro*)



[●] and [○] : submitted version.

Figure 6. Effect of the cutoff parameter

SAIQA-QAC2 examines only high-scored documents. If a document's score is lower than κ times the best document score, it is not examined at all. This cutoff parameter was introduced for efficiency. We used $\kappa = 0.6$ in the formal run, but it turned out that $\kappa = 0.8$ gives a better score: MRR = 0.614 and Top 5 = 0.754. That is, we can discard more documents.

5 Concluding Remarks

We developed a robust proximity search engine that has two disjunction operators. Our system based on the search engine showed a good performance for QAC2 task1. We found that slight modifications of a few parameters yielded better scores.

Table 5. QA performance of different retrieval systems

		MultiText	IDF	$\beta = 10^{-2}$	0.005	10^{-3}	10^{-4}
$R = 3$	Top 5	0.651	0.641	0.662	**0.677**	0.651	0.631
	MRR	0.568	0.544	0.554	**0.581**	0.572	0.537
$R = 5$	Top 5	0.677	0.667	0.672	**0.692**	**0.692**	0.667
	MRR	0.579	0.560	0.573	0.594	**0.596**	0.571

Table 6. Performance of the proximity search engine

#Qs is the number of questions that can be answered by the top documents.
%Qs is its percentage in the 195 questions.

Document rank	1	≤2	≤3	≤5	≤10	≤20	≤100
#Qs	148	165	177	183	188	193	194
%Qs	75.9%	84.6%	90.8%	93.8%	96.4%	99.0%	99.5%

References

[1] C. L. A. Clarke and E. L. Terra. Passage retrieval vs. document retrieval for factorid question answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 427–428, 2003.

[2] S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, and P. Morarescu. The role of lexico-semantic feedback in open-domain textual question-answering. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 274–281, 2001.

[3] Y. Hayashi, G. Kikui, and J. Tomita. Searching text-rich XML documents with relevance ranking. In *Proceedings of SIGIR 2000 Workshop on XML and Information Retrieval*, 2000.

[4] T. Hirao, Y. Sasaki, and H. Isozaki. An extrinsic evaluation for question-biased text summarization on QA tasks. In *Proceedings of the Workshop on Automatic Summarization, The Second Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 61–68, 2001.

[5] S. Ikehara, M. Miyazaki, S. Shirai, A. Yokoo, H. Nakaiwa, K. Ogura, Y. Ooyama, and Y. Hayashi. *Goi-Taikei — A Japanese Lexicon (in Japanese)*. Iwanami Shoten, 1997.

[6] S. Ikehara, S. Shirai, K. Ogura, A. Yokoo, H. Nakaiwa, and T. Kawaoka. ALT-J/E: A Japanese to English machine translation system for communication with translation. In *Proceedings of IFIP World Computer Congress*, pages 80–85, 1994.

[7] H. Isozaki and H. Kazawa. Efficient support vector classifiers for named entity recognition. In *Proceedings of COLING-2002*, pages 390–396, 2002.

[8] S. E. Robertson and S. Walker. Okapi/keenbow at trec-8. In *Proceedings of the Eigth Text Retrieval Conference*, pages 151–162, 1999.

[9] K. Sadakane and H. Imai. Fast algorithms for k-word proximity search. In *IEICE Transactions on Communications/Electronics/Information and Systems*, volume E84-A, pages 312–319, 2001.

[10] Y. Sasaki. Question answering as abduction: A feasibility study at NTCIR QAC1. *IEICE Transaction on Information and Systems*, E86-D(9):1669–1676, 2003.

[11] Y. Sasaki, H. Isozaki, T. Hirao, K. Kokuryou, and E. Maeda. NTT's QA systems for NTCIR QAC-1. In *Working Notes of the Third NTCIR Workshop Meeting, Part IV: Question Answering Challenge (QAC1)*, pages 63–70, 2002.

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