# A* Search Algorithm for Question Answering

Tatsunori MORI Tomohiro OHTA Katsuyuki FUJIHATA Ryutaro KUMON
Graduate School of Environment and Information Sciences, Yokohama National University
79-7 Tokiwadai, Hodogaya, Yokohama 240-8501, Japan
{mori,tomo_o,fujihata,kumon}@forest.eis.ynu.ac.jp

## Abstract

*In this paper, we propose a method to introduce an A* search control into sentential matching mechanism for question-answering systems, in order to reduce the response time while the accuracy of the answer is preserved. The question answering is a new technology to retrieve not relevant documents but the answer(s) directly by combining several methodology including IR and IE. One of the essential processes is the sentential matching between the user's query and each sentence in documents. In general, in order to obtain matching score precisely in higher resolution, we need some processes with higher computational costs. We therefore introduce an A* search in which both the processing cost and the resolution of matching score are took into account simultaneously.*

*According to the experiments in NTCIR3 QAC1 Task1, the system with the controled search is about 3.4–8.5 times faster than the system with no control.*

## 1 Introduction

Technology of Question Answering (QA) is widely noticed as an advanced style of fusion of Information Retrieval (IR) and Information Extraction (IE). QA systems give us not relevant documents but the answers of question. For example, if we submit the question "Who is the president of Japan?", the system will answer the phrase 'Jun-ichiro Koizumi'. The typical QA systems accept factual questions about *who*, *when*, *where*, *what* and *how* (4W1H), and find answer candidates by IR techniques like passage retrieval and IE techniques like Named Entity (NE) extraction [12, 13, 14].

Answers for 4W1H questions may classified into two groups. First group is the answers for *how* and consists of numerical expressions like distance, time, and so on. Second group corresponds to the answers for 4W, and consists of proper names like person name and time expressions like date.

As for the former type of question, Fujihata et al.[2] proposed a method to extract numerical expressions along with their related expressions by dependency analysis. They also showed that the combination of those information works well to find answers about numbers.

As for the later type of question, many researchers proposed methods to find answers according to the consistency between the type of question and the type of NE in target documents[1, 4, 11, 3, 8]. However, if only the type of NE is used as an evidence for answer, the quality of answer estimation may not be satisfactory. In order to improve the accuracy, Murata et al.[7] proposed a method that takes account of sentential similarity between a question and each of sentences in target texts according to part of speech (POS) information and syntactic information, as well as information about NE. However, this method has a drawback in terms of calculation cost, because it treats all of possible expressions in documents equally, and it may be a very time-consuming task.

To cope with the problem about response time, Prager et al.[9, 10] proposed the method called *predictive annotation*. The system performs a set of preprocessing like NE spotting on the target documents, and annotates the documents with extracted information in advance. This proposal can be regarded as a way to reduce the cost of calculation in the response time. The method, however, supposes that all of target documents are under the control of the system. It therefore is not applicable for the documents on WWW, which cannot be preprocessed usually. Moreover, some of preprocessors take very long time. For example, although an NE spotter based on Support Vector Machines (SVM) is very acculate, it is not suitable for preprocessing all of a large set of documents equally because it is very slow[1].

In this paper, we propose a method to reduce the calculation cost of the sentential matching by introducing the A* search in order to process the most promising candidates firstly and delay the processing of other candidates. The proposed method can settle the tradeoff between the cost of calculation and the accuracy of answering. Moreover, we can use slow analyzers like an SVM-based NE spotter in question answering because it does not need preprocessing for all of doc-

[1]The NE spotter proposed by Yamada et al. [15] takes about 0.9 second to process one sentence in Japanese newspaper articles with a Pentium III 933MHz computer. Since a set of Japanese newspaper articles for one year consits of about 160 million sentences, it takes about a half year to process all of newspapaer articles in 10 years.

uments of knowledge source.

## 2 A Question Answering System

### 2.1 System Overview

The overview of our system is shown in Figure 1. It basically follows the architectures of recent question answering systems, but the sentential matching module is enhanced with the A* search mechanism, which is newly proposed in this paper. The system consists of four modules: a question analyzer, a search engine, a passage extractor and a sentential matcher. In the rest of this section, we will describe each of modules briefly. The sentential matching mechanism, which is the core of our scheme, will be discussed in the next section.

### 2.2 Question Analyzer

As clues for finding answers, the module extracts the two types of information from questions.

First information is the list of *keywords*, which should be highly related to answers. Since content words in a question tend to appear along with the answer in target documents, we regard them as keywords. More precisely, we regard nouns as keywords for document retrieval, while for passage retrieval and sentential matching we adopt verbs and adjectives as well as nouns. Note that stop words like formal nouns, e.g. Japanese words '*koto*' and '*mono*', are eliminated from the keyword list.

Second information is the type of question such as 'Person', 'Location', 'Money' and so forth. It can be consider as a restriction on the kind of answers. If the system can obtain the question type, IE techniques like NE spotter can be used to narrow the answer candidates effectively. Many of question types are detectable from surface expressions in questions, like interrogatives, adverbs, and their compound expressions. For example, if the question:

(1) Kojien daigohan-ha *itsu* hatsubai-sare-mashi-taka.
KOJIEN fifth edition-TOP *when* pulish-PAS-POL-PAST
*When* was the fifth edition of KOJIEN published?

is given, then the system may conclude that the type of question is *Date* or *Time*.

### 2.3 Search Engine

This module retrieves documents related to keywords, which are obtained by the question analyzer. It is our original search engine and is based on an ordinary tf*idf method for term weighting and the vector space model for calculating similarity between a list of keywords and a document.

### 2.4 Passage Extractor

The candidates of answers are usually small parts of documents. Since systems may take a long time to process sentential matching, which is the following subprocess, retrieved documents should be broken down into smaller units like passages. The passage extractor segments each document into small passages and selects suitable passages related to keywords.

Our passage retrieval module processes retrieved documents as follows:

1. By sliding a window of passage through documents, obtain a set of passages. The window size is given as number of sentences, and is three in our experiment.
2. For each passage, check the number of (distinct) keywords, then give the score to the passage according to the number.
3. If some clue words related to the question type appear in a passage, then give some additional score to the passage.
4. Sort passages by their scores and obtain the n-best passages.

### 2.5 Sentential Matcher

The input for this module is a set of sentences in retrieved passages. The module gives a matching score to each morpheme in the input, except functional words like particles (i.e. case markers in Japanese) and symbols. The matching score represents the fitness of each morpheme for the answer. Like other QA systems, we suppose that the fitness of each morphem can be measured by the similarity, or matching, between the context of the morpheme and the question, which is calculated in the following two steps:

1. For each morphem, suppose it to be an answer, namely, bind the morpheme to the interrogative of the question.
2. Under the above restriction, calculate the similarity between the context of the morpheme, namely the other part of the target sentence having the morpheme, and the question sentence except the interrogative.

Finally, the compound noun having the morpheme of the highest score will be extracted as the most plausible answer candidate.

For the fitness of matching, several types of measures have been proposed so far. Many of QA systems utilize some combination of them. For example, similarity between dependency structures of a question and a target sentences is widely used as matching score. We can also treat the semantic matching in the similarity calculation by introducing paraphrasing mechanisms. The fitness of matching between a morpheme of answer candidate and an interrogative would be measured by the NE type of the morpheme which may be extracted by some NE spotter.

In general, in order to obtain matching score precisely in higher resolution, we need some processes

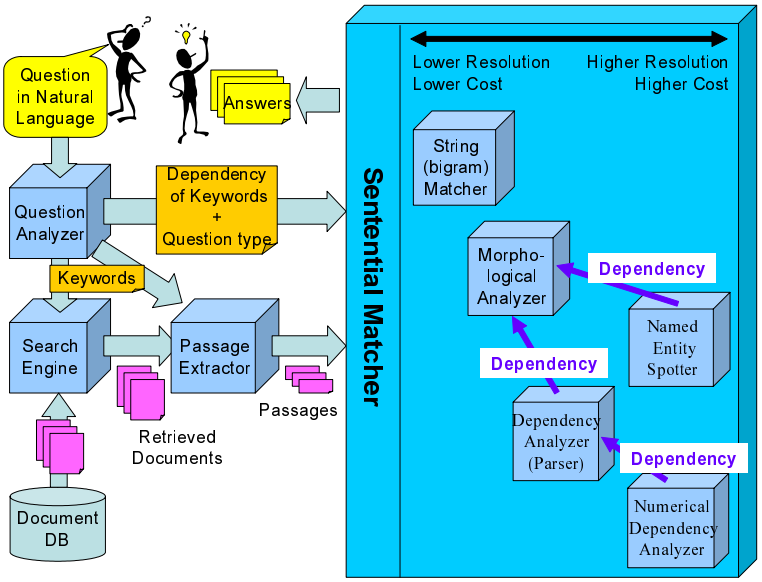


**Figure 1. System Overview**

with higher computational costs like morphological analysis, dependency analysis, NE extraction and so on. The resolution in matching score and the calculation cost are in a trade-off relation. As described in the following section, we therefore propose a method to minimize the total computational cost of sentential matching by controlling execution of processes.

Currently we adopt a fundamental matching score, which is calculated as the weighted sum of following four matching scores with the two steps described before.

1. Matching in terms of character.

   The number of character bi-grams in the question sentence which also appear in the target sentence.

2. Matching in terms of keywords.

   The number of keywords in the question sentence which also appear in the target sentence[2]. If a matched keyword has a same following particle, some extra score is added because the same particle means that the keyword plays the same syntactic role in the target sentence as well as in the question.

3. Matching in terms of dependency structure.

   The weighted sum of the sub-scores obtained from the following dependency information in target sentences.

   (a) Distance of dependency of the candidate morpheme on a certain keyword.

   (b) Distance of dependency of a certain keyword on the candidate morpheme.

   (c) Number of keywords that depend on the predicate on which the candidate morpheme also depends.

   To put it briefly, by this scoring, a candidate morpheme is given higher score, if the candidate and its context cover larger part of the question with smaller dependency structure.

4. Matching in terms of question type.

   Consistency between question type, which obtained from the interrogative of question, and the semantic information of the candidate morpheme. As semantic information, the type of NE and the information of numerical expressions are used[3].

## 3 Sentential Matching with Controlled Search

### 3.1 A* Search Algorithm for Sentential Matching

As described in Section 1, it is very inefficient that all of processes are uniformly performed for every answer candidate. Since one sub-process in sentential matching may have dependency on some of other sub-processes, the calculation of matching score for a candidate can be broken down into a series of execution of several sub-processes in order of dependency. We therefore propose a sentential matching method with a controlled search, which does not execute all of processes for every candidate equally, but processes the most promising candidate firstly. By introducing the control, the higher the estimated score of a candidate

[2] In the evaluation by QAC1 formal run, we take account of not only the target sentence, but also the preceding sentences in the passage window. See Section 4.

[3] In the evaluation by QAC1 dry run, we did not use the extractor of numerical expressions. See Section 4.

is, the earlier all of its sub-processes will finish. Thus, the control works effectively in deriving not only the most plausible answer candidate, but also the n-best candidates.

All of sub-processes for each candidate can be represented as a path of the search tree shown in Figure 2, if we suppose the correspondence shown in Table 1.

In the A$^*$ search, the estimated score $\hat{f}(c,n)$ of the candidate $c$ at the processing step $n$ are represented as the following summation of the exact score $S_1(c,n)$ obtained until the step $n$ and the score $\hat{S}_2(c,n)$, which is an estimated score of the rest of sub-processes.

$$\hat{f}(c,n) \quad = \quad S_1(c,n) + \hat{S}_2(c,n) \qquad (1)$$

The candidate with the highest score will be processed in the next turn. In order to find the best answer, the estimated score has to satisfy the condition (2).

$$\hat{S}_2(c,n) \quad \geq \quad S_2(c,n) \qquad (2)$$

If the requirement is not satisfied, there is no guarantee that the search will find the best answer. In such a situation, the search is called not 'A$^*$ search' but 'A search'.

One of the simplest way to estimate the score of a candidate is to adopt the maximum value in possible scores. By the estimation, we can always obtain the best answer because the condition (2) is satisfied. The estimation, however, cannot prune hopeless candidates effectively because the estimated scores of many candidates remain high. Thus, we introduce a more precise approximation of score of the next step and calculate the estimated score $\hat{S}_2(c,n)$ by the summation of the following values:

1. More precise approximation of score for the next step
2. Maximum score of each step after the next step

If the approximation score for the next step can be calculated with some small additional cost, we can prune candidates with low possibility in earlier stage. With the two-leveled approximation, the estimated score $\hat{f}(c,n)$ in the processing step $n$ of the candidate $c$ is revised as follows:

$$\hat{f}(c,n) \quad = \quad S_1(c,n) + \hat{S}_2^a(c,n) + \hat{S}_2^b(c,n) \quad (3)$$

where $\hat{S}_2^a(c,n)$ and $\hat{S}_2^b(c,n)$ are the approximated score of next step and the summation of maximum scores of steps after the next step, respectively. Since the value $\hat{S}_2^a(c,n)$ is the essential in this equation, we will discuss how to calculate it in the following section. Note that the search with the estimated score (3) may be not an A$^*$ search but an A search, because some of approximation methods would violate the condition (2) as described below.

## 3.2 Approximated Scores of Sentential Match

The estimated score $\hat{S}_2^a(c,n)$ should be derived by some estimation method with lower computational cost than the exact matching process. Moreover, the estimation of score should be more accurate than estimation by maximum value. In this section, we will consider the methods to calculate $\hat{S}_2^a(c,n)$ according to the guideline described above. If we have such estimation methods, the A$^*$ search can be performed with the algorithm shown in Figure 3.

### 3.2.1 Approximated Score of Keyword Matching

The score of keyword match is calculated as the weighted sum of the following values:

1. Number of keywords in target sentence excepting the answer candidate.
2. Number of pairs of keyword and particles in target sentence excepting the answer candidate.

While keywords and the following particles in a question have been already extracted by the question analyzer, we need a morphological analysis to judge how many keywords and their following particles appear in the target sentence[4].

The approximation of score is required to be performed with lower cost before morphological analysis of target sentences. We propose the following approximations, which uses string match only, for the values described above, respectively. Note that we give (approximated) scores to each *character* in a target sentence, because we do not have information about word boundaries at this stage.

1. Number of string match of a keyword in the target sentence excepting the target character.
2. Number of string match of a keyword and a particle in the target sentence excepting the target character.

These approximations satisfy the condition (2).

### 3.2.2 Approximated Score of Matching of Dependency Structure

As described in Section 2.5, the score of match of dependency structure is represented as the weighted sum of the following values.

1. Distance of dependency of the candidate on a certain keyword.
2. Distance of dependency of a certain keyword on the candidate.

[4] Since word boundaries do not appear explicitly in Japanese sentences, one of main purposes of morphological analysis for Japanese sentences is the segmentation of sentences into words.

**Table 1. Correspondence of Sentential Matching with Search Tree**

| | |
|---|---|
| Leaf | Candidate morpheme all of whose sub-processes have finished. |
| Score of leaf | Its matching score |
| Inner node | Candidate morpheme that is in the middle of calculating matching score. |
| Score of inner node | Its (estimated) matching score |
| Edge | One sub-process for a certain candidate. |

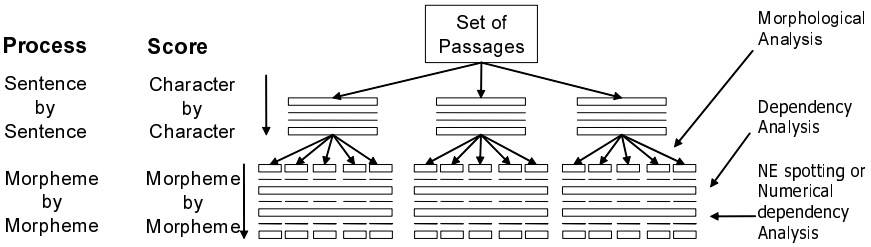


**Figure 2. Search Tree of Sentential Matching**

3. Number of keywords that depend on the predicate on which the candidate also depends.

Since the dependency analysis is performed after the morphological analysis, in this stage we use the result of morphological analysis to obtain the approximated score for each candidate morpheme by the following approximation:

1. Distance between the candidate morpheme and a certain keyword[5].

2. Number of keywords that *probably* depend on the predicate on which the candidate also *probably* depends. We suppose that the candidate and keywords *probably* depend on the nearest following verb.

Note that these approximation may not satisfy the condition (2), because this approximated dependency analysis may make a mistake.

#### 3.2.3 Approximated Score of Matching of Question Type

Matching of question type is judged according to the output of the NE spotter or the numerical expression extractor. The NE spotter searches for named entities in target sentences, and in our system it finds person names, organization names and location names in the output of morphological analyzer. On the other hand, the numerical expression extractor spots a triplet (*object*,*attribute*,*numeric+unit*) for each numerical expression in sentences. For example, if the sentence "The height of Tokyo Tower is 333m." is given (in Japanese), it outputs the triplet (*Tokyo Tower*, *height*, *333m*).

The appropriateness of candidate is judged in terms of consistency between the question type, which is obtained from the type of interrogative, and the type of candidate morpheme, which is extracted by the NE spotter or the numerical expression extractor. More precisely, the matching score of question type is calculated as follows:

1. If the question is type of numeric expression, the score is given according to whether the numerical triplet of the candidate in the target sentence is consistent with the triplet extracted from the question, or not.

2. Otherwise, the score is given according to whether the type of NE for the candidate is consistent with the question type, or not.

Now, let us consider the approximation of those scores. Firstly, in the case of the numerical expressions we use the following approximation, which is performed by pattern matching.

1. The score is given according to whether the pattern 'number+unit' is in the target sentence, or not.

This approximation satisfy the condition (2).

Secondly, in the other cases, we use the following approximation based on the result of morphological analysis.

1. The score is given according whether the semantic information[6] for the candidate is consistent with the question type, or not.

### 3.3 Generation of Answers

If the final score $f(c, n)$ of the candidate $c$ is greater than other estimated scores ($\hat{f}(c, n)$), the candidate is selected as the most plausible answer. Since the candidate is a morpheme and usually one constituent of a

[5]The unit of distance is morpheme.

[6]Many of Japanese morphological analyzers output some detailed POS information like semantic categories as well as basic POS information like grammatical categories.

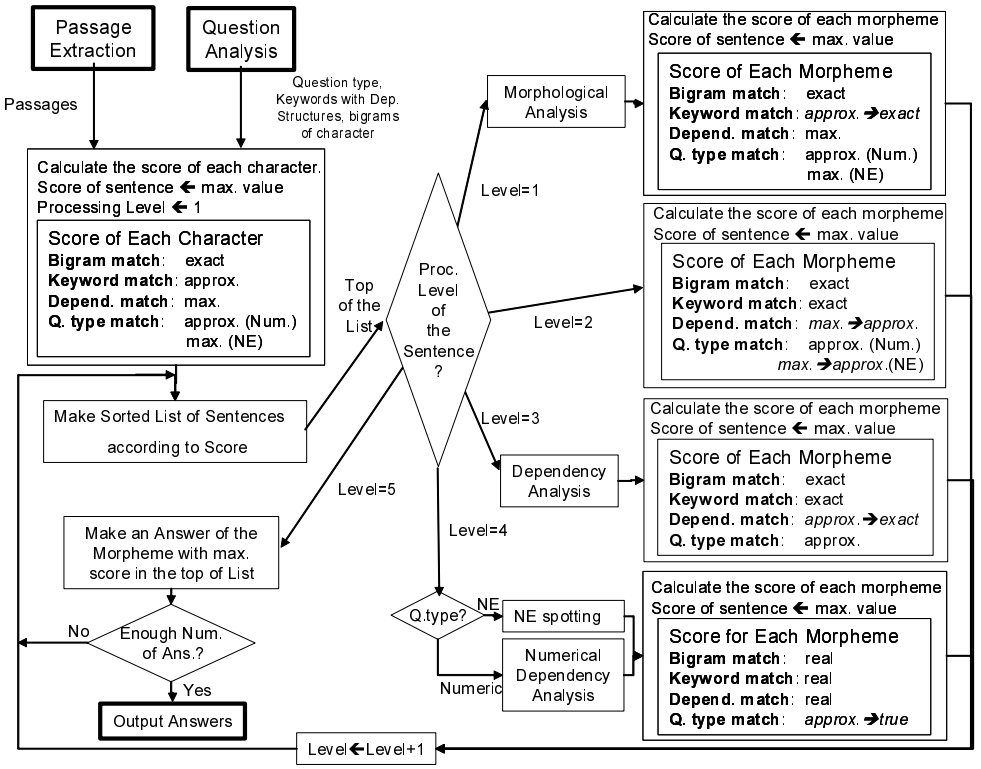


**Figure 3. Overview of Sentential Matching Algorithm with A* search**

compound noun, we select the series of nouns including the candidate as answer. If the candidate is marked as one of NE, then the series of the same type of NE is extracted.

When we need secondary or lower-ranked candidates, the search is continued until needed number of answers are obtained. Since some of matching process do not satisfy the A* condition (2), the score of the candidate obtained previously is may smaller than the score of the candidate obtained subsequently. Thus, if we would have plural answers, the answers should be sorted according to their final scores.

## 4 Experimental Result

In this section, we describe the experimental results of proposed QA system. We participated in NTCIR 3 QAC Task 1 (the dry run and the formal run), and conducted the following experiments under the condition shown in Table 2.

1. We compared the average time to calculate the approximated score $\hat{S}_2^a(n)$ with the average time to calculate the exact score $S_2(n)$. The result is shown in Table 4.

2. Table 5 and Table 6 show MRR[7] and the average of execution time in the sentential matching module for each of the following systems, when the five best answers are requested.

   (a) System with no search control.

   (b) System with an A* search controlled by maximum scores only.

   (c) System with an A* search controlled by approximated scores and maximum scores.

   Note that MRR values in those tables are based on not the official answer set supplied by QAC task organizers but our manual check, which is slightly looser judgment under the criterion that users can obtain the right answer from the output, because in this paper we do not want to commit ourself to errors in the last process in which a final answer are made from the most plausible candidate.

3. We also compared the system (c) with the system (b) in terms of the number of calls of each subprocess.

In those experiments, the 100 best documents are retrieved by IR system, and the passage extractor selected the 10 best passages from the documents. Each passage consists of a series of three sentences. Table 3 shows a part of scores for sub-processes we adopted. Note that we increase the matching score of question type for the formal run, because the match seems to be dominant in selecting the right answers.

[7] The accuracy of QA systems is usually evaluated with the value called '*Mean Reciprocal Rank (MRR)*'. It is the average of reciprocal rank of the highest right answer.

**Table 2. Condition of Experiment**

| | |
|---|---|
| Morphological analyzer | JUMAN 3.61 [6] |
| Dependency Analyzer | KNP 2.0b6 [5] |
| NE spotter | SVM-based NE spotter originally proposed by Yamda et al.[15] |
| Numerical Expression Extractor (formal run only) | System by Fujihata et al.[2] |
| Questions | 50 questions for QAC dry run and 200 questions for QAC formal run |
| Target documents (Knowledge Resource) | Mainichi Shimbun Newspaper Articles in 1998 and 1999 |
| Computer<br>for dry run<br>for formal run | <br>CPU: UltraSPARC-II 450MHz, Main memory:1GByte<br>CPU: Pentium III 1GHz × 2, Main memory:2GByte |

## 5 Discussion

With the result shown in Table 4, we can confirm that each approximation can be performed within much shorter time than the derivation of exact score.

In Table 5 and Table 6, we can see that the proposed method (c) is faster from 3.4 times (formal run) to 8.5 times (dry run) than (a), while MRR is almost unchanged[8]. This result shows that the A* search in QA system works effectively and we can cut down the computational cost drastically.

By comparing the method (c) with the method (b), we can see that the introduction of approximated scores into A* search is effective. The proposed method (c) is faster from 1.4 times (formal run) to 1.7 times (dry run) than the method (b), which uses only maximum scores. Since the process of NE extraction takes longer time than other processes, the total execution time has a tendency to depend on the number of execution time of NE extraction. However, in Table 7 and Table 8, we can confirm that the number of execution time is decreased equally in each sub-process.

Improvement by the A* search depends on the score distribution of sub-processes. Since we set the score of question type matching in the formal run by a larger value than one in the dry run, scores of many candidates remain high until the exact calculation of question type and consequently the gain in execution time of the formal run is smaller than the dry run.

Note that the MRR of the method (c) may be lower than the method (a) as shown in Table 6. When the A* condition (2) is not satisfy in the method (c), the right answer may be estimated at lower score and other (incorrect) candidates may be relatively ranked in higher positions. If we can find some other approximations that always satisfy the condition, we will obtain the same MRR for the method (a) and the method (c). The method (b) always satisfy the condition, MRR of (b) should be the same value as (c)[9].

[8]Since main part of our system is implemented with the programming language Perl, which is an interpreter, the absolute execution time is long.

[9]To the contrary, you may notice that MRR value of the method (b) is smaller than (a) in Table 6. It happened because the order of candidates with the same score can varies in the list of final answer.

## 6 Conclusion

In this paper, we propose a method of A* search for sentential matching in QA system. According to the experiments in NTCIR3 QAC1 Task1, the system with the controled search is about 3.4–8.5 times faster than the system with no control. It shows that our method can settle the trade-off between the cost of calculation and the accuracy of answering under the given computational resource.

On the other hand, in terms of accuracy of answers, there is room for improvement in our sentential matching method. The average of MRR for all participating systems is 0.303, while the official MRR of our system for formal run is 0.296. We have to consider introduction of other viewpoints other than ours. In our future works, we therefore would like to consider the improvement of accuracy including refinement of sentence matching and optimize the weight of each score.

## References

[1] E. Breck, J. Burger, L. Ferro, D. House, M. Light, and I. Mani. A Sys Called Qanda. In *Proceedings of the eighth Text Retrieval Conferene (TREC 8)*, pages 499–506, 1999.

[2] K. Fujihata, M. Shiga, and T. Mori. Extraction of numerical expressions by constraints and default rules of dependency structure. SIG Notes 2001-NL-145, Information Processing Society of Japan, Sept. 2001. (In Japanese).

[3] H. Kazawa, H. Isozaki, and E. Maeda. NTT Question Answering System in TREC 2001. In *Proceedings of the tenth Text Retrieval Conferene (TREC 10)*, pages 415–422, 2001.

[4] H. Kazawa and T. Kato. Question answering using semantic constraint on answers. SIG Notes 2000-NL-140, Information Processing Society of Japan, Nov. 2000.

[5] S. Kurohashi. *Japanese Syntactic Parsing System KNP version 2.0 b6 Instruction Manual*, 1998. (in Japanese).

[6] T. Kurohashi and M. Nagao. *Japanese Morphological Analysis System JUMAN version 3.6 Manual*. Kyoto University, 1998. (in Japanese).

[7] M. Murata, M. Utiyama, and H. Isahara. Question answering system using similarity-guided reasoning.

**Table 3. Sample scores for each matching**

| Checkpoint | Score |
|---|---|
| Distance of dependency of a certain keyword on the candidate. | 0∼2 |
| Number of keywords that depend on the predicate on which the candidate also depends. | # × 3 |
| NE type of the candidate is consistent with the question type<br>The triplet about the candidate is consistent with the triplet derived from the question. | 4(dry run), 10(formal run) |

**Table 4. Average execution time to derive one exact score or one approximated scores (at QAC dry run)**

| | Exact Score | Approximated Score | Ratio |
|---|---|---|---|
| Keyword Matching | 0.021 sec. | 0.0023 sec. | 0.11 |
| Matching of dependency structure | 0.15 sec. | 0.0029 sec. | 0.019 |
| Matching of question type | 11.93 sec. | 0.0003 sec. | 0.000025 |

**Table 5. MRR and Average execution time per question (at QAC dry run)**

| | MRR | Time | Ratio of time |
|---|---|---|---|
| (a) no control | 0.30 | 303.1 sec. | 8.49 |
| (b) maximum | 0.30 | 60.2 sec. | 1.69 |
| (c) approx. & maximum | 0.30 | 35.7 sec. | 1 |

**Table 6. MRR and Average execution time per question (at QAC formal run)**

| | MRR | Time | Ratio of time |
|---|---|---|---|
| (a) no control | 0.32 | 57.8 sec. | 3.42 |
| (b) maximum | 0.31 | 23.8 sec. | 1.41 |
| (c) approx. & maximum | 0.31 | 16.9 sec. | 1 |

**Table 7. Number of calls of each sub-process per one question (at QAC dry run)**

| | Morphological Analysis | Dependency Analysis | NE extraction |
|---|---|---|---|
| (b) maximum | 29.6 | 17.5 | 5.1 |
| (c) approx. & maximum | 22.8 | 3.5 | 2.4 |
| Ratio | 0.77 | 0.20 | 0.47 |

**Table 8. Number of calls of each sub-process per one question (at QAC formal run)**

| | Morphological Analysis | Dependency Analysis | NE extraction | Numerical exp. extraction |
|---|---|---|---|---|
| (b) maximum | 27.8 | 24.9 | 11.5 | 14.9 |
| (c) approx. & maximum | 23.9 | 20.2 | 7.8 | 7.6 |
| Ratio | 0.86 | 0.81 | 0.68 | 0.51 |

SIG Notes 2000-NL-135, Information Processing Society of Japan, Jan. 2000.

[8] J.-H. Oh, K.-S. Lee, D.-S. Chang, C.-W. Seo, and K.-S. Choi. TREC-10 Experiments at KAIST: Batch Filtering and Question Answering. In *Proceedings of the tenth Text Retrieval Conferene (TREC 10)*, pages 347–354, 2001.

[9] J. Prager, E. Brown, A. Coden, and D. Radev. Question-answering by predictive annotation. In *Proceedings of SIGIR 2000: 23th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 184–191, 2000.

[10] J. Prager, E. Brown, D. R. Radev, and K. Czuba. One search engine or two for question-answering. In *Proceedings of the ninth Text Retrieval Conferene (TREC 9)*, pages 235–240, 2000.

[11] Y. Sasaki, H. Isozaki, H. Taira, T. Hirao, H. Kazawa, J. Suzuki, K. Kokuryo, and E. Maeda. SAIQA: A japanese qa system based on a large-scale corpus. SIG Notes 2001-NL-145, Information Processing Society of Japan, Sept. 2001.

[12] TREC Project. Proceedings of the eighth text retreival conference TREC 8. http://trec.nist.gov/pubs/trec8/t8_proceedings.html, 1999.

[13] TREC Project. Proceedings of the eighth text retreival conference TREC 9. http://trec.nist.gov/pubs/trec9/t9_proceedings.html, 2000.

[14] TREC Project. Proceedings of the eighth text retreival conference TREC 10. http://trec.nist.gov/pubs/trec10/t10_proceedings.html, 2001.

[15] H. Yamada, T. Kudo, and Y. Matsumoto. Japanese named entity extraction using support vector machine. *IPSJ Journal*, 43(1):44–53, Jan. 2002.