Japanese Q/A System using A* Search and Its Improvement
— Yokohama National University at QAC2 —

Tatsunori MORI
Graduate School of Environment and Information Sciences, Yokohama National University
79-7 Tokiwadai, Hodogaya, Yokohama 240-8501, Japan
mori@forest.eis.ynu.ac.jp

Abstract

In NTCIR3 QAC1, we proposed a method to introduce an A* search control into sentential matching mechanism for Japanese question answering systems, in order to reduce the turn-around time while keeping the accuracy of the answers. Using the method, we do not have to any preprocessing on a document database, and we may use any IR systems with writing a simple wrapper program. There however is a problem that the accuracy was not so high and MRR was about 0.3.

In order to improve the accuracy, we therefore propose several measures of degree of sentence matching and a variant of voting method. Both of them can be integrated with our scheme of controlled search. By using those techniques, the system achieves a higher MRR, 0.489, in NTCIR4 QAC2.

Keywords: *A* search, Sentence Chaining, Dependency vector, Pseudo voting method.*

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 factoid 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 [17, 14].

Many researchers proposed methods to find answers according to the consistency between the type of question and the type of NE or numeric expression in passages retrieved by an IR system[1, 16, 5, 6, 12]. The consistency of type itself, however, is not a very strong support, because extracted passages may contain several NEs of a same type.

Most of QA systems accordingly gives each answer candidate *matching score* that expresses the degree of consistency between the question and a context of an answer candidate. For example, many QA systems adopt distance (or nearness) between an answer candidate and keywords in a segment as matching score.

More sophisticated methods have been employed to improve the accuracy of scoring answer candidates. Harabagiu et al.[4], Murata et al.[11], Sasaki et al.[15] and Kuwahara et al.[7] proposed methods that parse each sentences in retrieved passages and make logical forms to perform sentential matching, inference and paraphrase. Those methods require time-consuming subprocesses, like POS tagging, parsing, NE spotting and so on. If all of such subprocesses are applied to all of retrieved passages, the processing time will be very long and the system will be of no practical use.

To cope with the problem about response time, Prager et al.[13] proposed the method called *predictive annotation*. The system performs time-consuming subprocesses on document databases in advance, and annotates the documents with extracted information. The method, however, supposes that all of documents are under the control of the system. It therefore is not applicable for the documents on WWW. Moreover, some of preprocessors take very long time. For example, although a Japanese NE spotter based on Support Vector Machines (SVM) works very precisely, it is not suitable for preprocessing all of a large document database exhaustively because it is very slow.

On the other hand, Lee et al.[8] adopt Lexico-Semantic Patters to find candidates suitable for a question type and to score them. Although the method works very quickly, it require a huge set of patterns according to domains. Since it also requires the result of morphological analysis[1], the scheme of predictive annotation is still indispensable.

In order to cope with these problems, we have proposed a method to introduce an A* search control into sentential matching mechanism for a Japanese QA system, in order to reduce the turn-around time while keeping the accuracy of the answers, without any preprocessing on documents[10]. The system processes the most promising candidates firstly and delays the processing of other candidates. Thus, we may use accurate but time-consuming analyzers like an SVM-based NE recognizer. The MRR[2] of the system, however, is still low (about 0.3) as we have reported in [10].

In this paper, in order to improve the accuracy, we propose several measures of degree of sentence matching and a variant of voting method. Both of them can be integrated with our scheme of controlled search. We also examine the effectiveness of the newly introduced techniques at NTCIR4 QAC2 Subtask 1.

[1]Since Japanese sentences are not segmented by spaces, we have to identify word boundary firstly in order to perform POS tagging. Japanese morphological analyzers usually perform segmentation and POS tagging simultaneously.

[2]See Section 7.2.

2 System Overview

The overview of the proposed system is shown in Figure 1. It has the following features.

Firstly, several new techniques for sentential matching are introduced to improve the accuracy of scoring answer candidates (see Section 3). Secondly, the A* search algorithm with two-stage scoring function has been introduced in our previous work to reduce the turn-around time while keeping the accuracy of the answers, without any preprocessing on document databases (see Section 4).

In the rest of this section, we describe each of modules in the system.

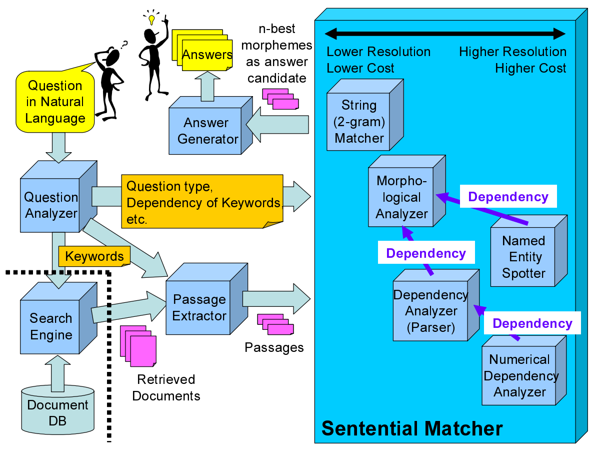

Figure 1. System Overview

2.1 Question Analyzer

The question analyzer receives a question from a user and extracts the following information: 1) results of morphological analysis and syntactic parsing, 2) a list of keywords with weights of importance (see also Section 3), 3) a question type, and 4) dependency structures of numerical expressions. Here, we define the term *Keywords* as content words in a given question. The question type, like PERSON, LENGTH and so on, is estimated by a set of hand-crafted patterns. If the question type is detected as numerical expression, a numerical expression extractor[3] is applied. It extracts a triplet (*object*, *attribute*, *numeric+unit*) for each numerical expression in a sentence. For example, it outputs the triplet (*Tokyo Tower*, *height*, *333m*), when the sentence “The height of Tokyo Tower is 333m.” is given (in Japanese).

2.2 Passage Extractor

Since the information related to a question is usually in a very small part of document, the passage extractor segments each document into small passages and selects suitable passages related to keywords. In our experiment, we defined one passage as a sequence of three sentences, according to Murata et. al.[11].

2.3 Sentential Matcher

The input for this module is a set of sentences in retrieved passages. We call these sentences *retrieved sentences*. The module treats each morpheme as an answer candidate (AC) and gives it a matching score. The matching score represents the fitness of each answer candidate for the answer. Like other QA systems, the scores ara calculated by the following two steps:

1. For each answer candidate, firstly suppose that it is an answer, and link up the answer candidate AC to an interrogative Q in a question.

2. Under the above condition, calculate the matching score according to the similarity between the context of AC and the question sentence except Q.

As the matching score, we adopt a linear combination of the several sub-scores described in Section 3.

The output of this module is a list of n-best morphemes with scores.

2.4 Answer Generation

A morpheme of an answer candidate is either a word or a part of a longer compound word. In the latter case, the system finds a compound word including the answer candidate, and outputs it.

3 Improving Accuracy of Japanese Sentential Matcher

According to the present situation of NLP techniques, there should be some trade-off relation between expressiveness of data structure and robustness of analysis in each processing techniques. Thus, we suspect that it is difficult to make QA systems high-precision with one monolithic method. For example, the ‘distance’ scheme, in which the degree of sentential matching is represented as a function of distance between an answer candidate and other keywords, is very robust, but it is rude approximation. On the other hand, logical forms can express the situation of matching in detail, but it is difficult to extract logical forms robustly and precisely. Consequently we think it is desirable that QA systems employ multiple complementary methods so as to have a variety of expressiveness and robustness.

We therefore introduce a composite matching score shown in Formula (1), which is a linear combination of the following sub-scores for an answer candidate AC in the i-th retrieved sentence L_i with respect to a question sentence L_q having an interrogative Q: 1) the matching score in terms of 2-gram, $Sb(AC, L_i, L_q)$, 2) the matching score in terms of keyword, $Sk(AC, L_i, L_q)$, 3) the matching score in terms of dependency between an answer candidate and a keyword, $Sd(AC, L_i, L_q)$, and 4) the matching score in terms of question type, $St(AC, L_i, L_q)$.

$$\begin{aligned} S(AC, L_i, L_q) = & \\ & Sb(AC, L_i, L_q) + Sk(AC, L_i, L_q) \\ & + Sd(AC, L_i, L_q) + St(AC, L_i, L_q) \end{aligned} \qquad (1)$$

In the rest of the section, we firstly introduce a method called 'sentence chaining' in order to apply sentential matching to a series of sentences. Then we describe each of sub-scores.

3.1 Sentence Chaining

The method proposed in the following subsections are for matching one question with one retrieved sentence because it is difficult to detect inter-sentential dependency precisely. Information of one question however may spread over multiple sentences. We cannot obtain correct answers by one-to-one sentential matching in this situation. We therefore propose a method to treat multiple sentences as one pseudo-sentence. Each of parse trees for sentences $L_0, \cdots, L_n$ is connected to the parse tree of the succeeding sentence, if the condition (2) is satisfied with respect to the question sentence L_q.

$$(KW(L_q) \cap \bigcup_{i=0}^{n-1} KW(L_i)) \not\subseteq (KW(L_q) \cap KW(L_n)) \quad (2)$$
$$\wedge (KW(L_q) \cap \bigcup_{i=0}^{n-1} KW(L_i)) \not\supseteq (KW(L_q) \cap KW(L_n))$$

where $KW(L_i)$ is the set of keywords appearing in L_i. Intuitively, when keywords in the question sentence are scattered all over the sentences, we connect the sentences. We call the method *sentence chaining*. In sentence chaining, the last BUNSETSU[3] segment of L_{i-1} is linked up to the BUNSETSU segment with the Japanese topic marker '*wa*' in L_i. It is a natural consequence of the fact that a topic marker expresses an old information and the antecedent should appears in the preceding sentences. In the case that L_i does not have a topic marker, the last BUNSETSU segment of L_{i-1} is connected to the first BUNSETSU segment of L_i.

3.2 Matching in terms of Keywords

The matching score in terms of keywords, Sk, is calculated according to the number of keywords shared by a question and a retrieved sentence. If a matched keyword has a same case marker[4] , some extra score is added.

Formula (3) defines the score $Sk(AC, L_i, L_q)$ for an answer candidate AC in a retrieved sentence L_i with respect to a question L_q:

$$\begin{aligned} Sk(AC, L_i, L_q) &= C_k \sum_{k \in SKW(AC, L_i, L_q)} w(k) + \\ & \quad C_c \sum_{\langle k,c \rangle \in SKWC(AC, L_i, L_q)} w(k) \qquad (3) \\ SKW(AC, L_i, L_q) &= (KW(L_i) \setminus \{AC\}) \cap KW(L_q) \\ SKWC(AC, L_i, L_q) &= (KW_{case}(L_i) \setminus \{\langle AC, c_{AC} \rangle\}) \\ & \quad \cap KW_{case}(L_q) \end{aligned}$$

where the function $KW_{case}(L)$ returns a set of keyword-case pairs in the sentence L. The constants C_k and C_c are mixing factors in the composite score of (1). c_{AC} is the case marker of AC.

In our scheme, we suppose that documents are available only by way of an external search engine. Consequently we do not use any global term weighting methods like IDF. In stead of those kinds of weighting, we introduce a term weighting method that depends only on output of a search engine. The weight of a keyword K_i is defined as the following formula:

$$w(K_i) = \frac{1}{\frac{freq(K_i, D'_i)}{N_{kd}} + 1} \qquad (4)$$

where D'_i is the set of top N_{kd} documents retrieved by a query that consists of all keywords except K_i, and $freq(K_i, D'_i)$ is the frequency of K_i in the document set D'_i. Intuitively speaking, the weight measures the importance of keyword in terms of necessity in the query. When a keyword tends to appear independently of other keywords, the keyword have higher weight.

3.3 Matching in terms of 2-gram

The matching score in terms of 2-gram is calculated according to the number of character 2-grams shared by a retrieved sentence and a question:

$$Sb(s, e, L_i, L_q) = \qquad (5)$$
$$C_b \sum_{k \in KW(L_q)} \frac{\sum_{l=s}^{e} \sum_{j=1}^{len(L_i)} bfreq(L_i, L_q, l, j)}{len(AC)}$$
$$bfreq(L_i, L_q, l, j) = \qquad (6)$$
$$(j \neq l) \cdot freq(substr(L_i, j, j+1), L_q)$$

where s and e are the character positions of start and end of AC in L_i, and the constant C_b is a mixing factor in (1).

3.4 Matching in terms of Dependency Structure

Unfortunately, Japanese parsers are not precise enough to generate complete logical forms[5]. We therefore would like to use only necessary parts of a syntactic tree in some robust manner. Accordingly, we only use the information about position of an answer candidate relative to keywords in dependency structure. It is important to note that distance between two word in surface expression is not suitable for measuring distance between two words in Japanese, because the order of noun phrases are relatively flexible.

Here, we propose a new vector representation of a dependency structure in a parse tree. It represents the information about relative position in each dependency structure as a two-dimensional vector in the following way. Firstly, we extract a dependency relation between an answer candidate AC and each keyword K_i in a retrieved sentence. Then, relative position of AC to K_i is represented as a *dependency vector* $DV(AC, K_i)$ as shown in Figure 2. Similarly we can obtain a dependency vector $DV(Q, K_i)$ for a question, where Q is the interrogative in L_q.

[3]BUNSETSU segment is a kind of Japanese chunk, which consists of one content word and a few function words. BUNSETSU segment is considered as unit of dependency analysis.

[4]Japanese has explicit case markers as post-noun particles, because the order of noun phrases are relatively flexible.

[5]The accuracy of the state-of-the-art Japanese parser is about 90% in dependency analysis, and this means that one error will occur in ten dependencies.

Using the dependency vector, we can define a measures of degree of matching between the dependency structures of AC in L_i and one of Q in L_q as Formula (7):

$$\begin{aligned} Sd(AC, L_i, L_q) &= C_d \sum_{k \in SKW(AC, L_i, L_q)} Sd1(AC, Q, k) \qquad (7) \\ Sd1(AC, Q, k) &= (Sd_{sum}(AC, Q, k) + \\ &\quad Sd_{diff}(AC, Q, k)) \cdot w(k) \\ Sd_{sum}(AC, Q, k) &= \frac{1}{1 + \frac{1}{2}(|DV(AC, k)| + |DV(Q, k)|)} \\ Sd_{diff}(AC, Q, k) &= \frac{1}{2 + |DV(AC, k) - DV(Q, k)|} \end{aligned}$$

where the score of 'sum' Sd_{sum} represents the nearness between a keyword and an interrogative/answer candidate, while the score of 'difference' $Sd_{diff}(AC)$ expresses some kind of structural similarity between two dependencies.

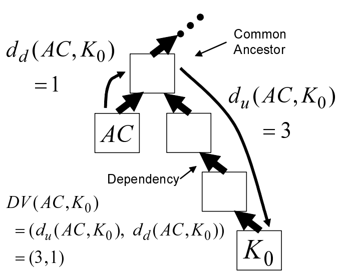

Figure 2. Dependency Vector — A Vector Representation of a Dependency Structure in a Parse Tree

The vector representation also makes *transformation rules* of dependency structure very concise. Since an interrogative sentences may have a different dependency structure from a declarative sentence, some transformation of questions is necessary to improve recall of answers. Such transformation can be represented as a set of simple rewriting rules of dependency vectors shown in Figure 3.

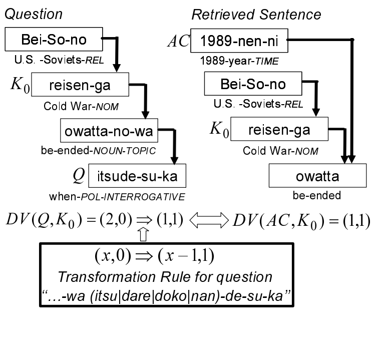

Figure 3. Transformation Rule of Sentence in terms of Dependency Vector

3.5 Matching in terms of Question Type

The matching score $St(AC, L_i, L_q)$ in terms of question type is calculated as follows. Firstly, a question type is estimated by the type of an interrogative and other clue expressions. According to the question type, one of three semantic type analyzers is performed. If the question type is in the entity class like PERSON, all of retrieved sentences are passed to an NE recognizer to find named entities. The matching score is calculated according to the consistency between the NE type and the question type.

When the question type is supported by our numerical expression extractor like LENGTH, a set of patterns are use to filter out answer candidates that are not numerics or do not have suitable unit expressions. Then, retrieved sentences with suitable numerics are passed to the extractor to obtain triplets (*object*, *attribute*, *numeric+unit*). The matching score is calculated according to the consistency with respect to triplets.

If the question type is one of other simple numeric expressions like DATE, a set of patterns are applied to retrieved sentences to filter out answer candidates that are not numerics or do not have suitable unit expressions.

4 A* Search Algorithm for Sentential Matching

It is very inefficient that all of processes are uniformly performed for every answer candidate. Since one sub-process in sentential matching has dependency on some 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 the dependency. We therefore proposed a sentential matching method with a controlled search, which processes the most promising candidate firstly. The control also works effectively in finding n-best candidates.

All sub-processes to be applied to one candidate can be represented as a path of the search tree like Figure 4.

In the A* search, the estimated score $\hat{S}(AC, n)$ of a answer candidate AC at the processing step n is represented as the following summation of the exact score $S_1(AC, n)$ obtained until the step n and the score $\hat{S}_2(AC, n)$, which is an estimated score for the rest of sub-processes:

$$\hat{S}(AC, n) = S_1(AC, n) + \hat{S}_2(AC, n) \qquad (8)$$

where we omit L_i and L_q for conciseness. 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:

$$\hat{S}_2(AC, n) \geq S_2(AC, n) \qquad (9)$$

One of the simplest ways to estimate the score of a candidate is to adopt the maximum value of possible scores. By the estimation, we can always obtain the best answer because the condition (9) 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 revise

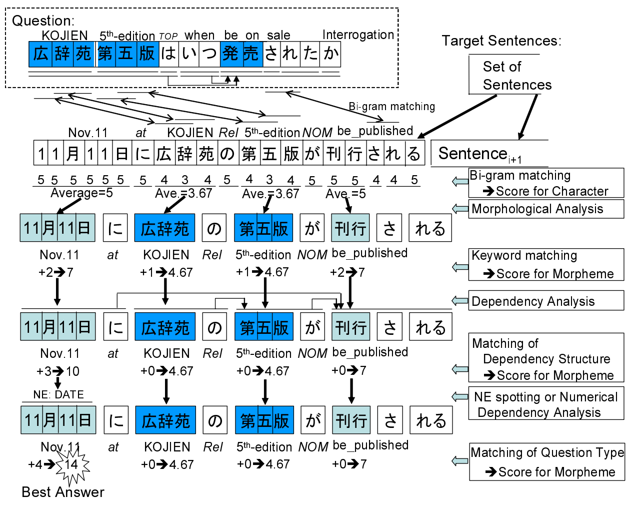
Question:
KOJIEN 5th-edition TOP when be on sale Interrogation
広辞苑第五版はいつ発売されたか
Target Sentences:
Set of Sentences
Bi-gram matching
Nov.11 at KOJIEN Rel 5th-edition NOM be_published
11月11日に広辞苑の第五版が刊行される
Sentence i+1
5 5 5 5 5 5 5 4 3 4 5 4 3 4 5 5 5 4 4 5
Average=5 Ave.=3.67 Ave.=3.67 Ave.=5
Bi-gram matching ➔Score for Character
Morphological Analysis
11月11日 に 広辞苑 の 第五版 が 刊行 さ れる
Nov.11 at KOJIEN Rel 5th-edition NOM be_published
+2➔7 +1➔4.67 +1➔4.67 +2➔7
Keyword matching ➔Score for Morpheme
Dependency Analysis
+3➔10 +0➔4.67 +0➔4.67 +0➔7
Matching of Dependency Structure ➔Score for Morpheme
NE: DATE
NE spotting or Numerical Dependency Analysis
+4➔14 +0➔4.67 +0➔4.67 +0➔7
Best Answer
Matching of Question Type ➔Score for Morpheme

Figure 4. Search Tree of Sentential Matching

the estimated score $\hat{S}(AC, n)$ as follows:

$$\hat{S}(AC, n) = S_1(AC, n) + \hat{S}_2^a(AC, n) + \hat{S}_2^b(AC, n) \quad (10)$$

where $\hat{S}_2^a(AC, n)$ is a more precise approximation of score for the next step, and $\hat{S}_2^b(AC, n)$ is the sum of maximum scores of steps in the rest of the path.

If the approximation score $\hat{S}_2^a(AC, n)$ for the next step can be calculated with some small additional cost, we can prune candidates with low possibility in earlier stage.

5 Approximated Scores of Sentential Matching

In this section, we discuss methods to calculate approximated scores for matching sub-scores in Section 3.

5.1 Approximated Score of Keyword Matching

While keywords and their post-positional particles in a question have been already extracted by the question analyzer, we need a morphological analysis to judge whether a retrieved sentence includes each keyword and its post-positional or not[6]. The calculation of approximated score is required to be performed with lower cost than morphological analysis. We therefore adopt the simple string match to judge whether the string of each keyword occurs in the string of a retrieved sentence or not. This approximation satisfies the condition (9).

[6]Since Japanese does not have a marker of word boundaries, one of main purposes of morphological analysis for Japanese sentences is the identification of the boundaries.

5.2 Approximated Score of Matching of Dependency Structure

It is difficult to estimate dependency structures precisely without parsing. Thus, we use the information of 1) occurrence of keywords, 2) boundaries of BUNSETSU segments and 3) the cases of BUNSETSU segments which can be derived from the result of POS tagging. The approximation consists of the following two stages.

5.2.1 First Approximation

The first approximation is performed before morphological analysis. The approximated score is obtained by calculating Sd in Formula (7) under the assumption that all of keyword strings in a retrieved sentence are in *complete-matching* with respect to a question with an interrogative Q. We call a keyword K_i is in *complete-matching*, when the keyword satisfies the following relationship:

$$DV(Q, K_i) = DV(AC, K_i) \quad (11)$$

The score estimated by this method satisfies the condition (9).

5.2.2 Second Approximation

The second approximation is based on estimation of boundaries of BUNSETSU segment after morphological analysis. Using boundaries, we can also estimate the number of BUNSETSU segments between a answer candidate AC and a keyword K_i, and the case marker of each BUNSETSU segment. Those kinds of information can be used to refine the first approximation as follows.

Relative Position of AC and K_i

If the following relationship between the relative positions of AC and K_i is satisfied, it is obvious that the keyword K_i is not in complete-matching:

$$(DV(Q, K_i) = (a, 0) \wedge (a > 0) \wedge pos_r(AC) < pos_r(K_i)) \quad \vee$$
$$(DV(Q, K_i) = (0, b) \wedge (b > 0) \wedge pos_r(K_i) < pos_r(AC)) \quad (12)$$

where $pos_r(X)$ is the position of the morpheme X in a retrieved sentence. In that case, the first approximation is an over-estimation, and we can revise the score by finding another estimation of the dependency relation with the second best score. The estimated score also satisfies the condition (9).

Number of BUNSETSU segments n_B between AC and K_i

If the following relationship is satisfied, it is obvious that the keyword K_i is not in complete-matching.

$$DV(Q, K_i) = (a, 0) \wedge a > 1 \wedge a > n_B \quad (13)$$

In that case, we can also revise the score by finding another estimation of the dependency relation with the second best score. The estimated score does *not* satisfy the condition (9), because of possible errors in BUNSETSU segmentation.

The cases of BUNSETSU segments

Matching score for case markers can be estimated according to the case marker candidate assumed in the BUNSETSU segmentation. The estimated score does *not* satisfy the condition (9), because of possible errors in the BUNSETSU segmentation and the estimation of cases.

5.3 Approximated Score of Matching of Question Type

5.3.1 Named Entity

We give score to each of the following answer candidates based on the result of morphological analysis.

1. Candidates whose detailed POS information consistent with the question type[7].

2. Candidates that consist of KATAKANA characters or alphabets (They are possibly loanwords).

Since there are named entities that do not satisfy the above condition, the estimated score does *not* satisfy the condition (9).

5.3.2 Numerical Expression

The numerical expression extractor is applied to each answer candidates that matches one of patterns corresponding to a question type. Thus, we give each of those candidates the score that is calculated on the assumption that the triplet of the candidate completely matches one of a question. This approximation satisfy the condition (9).

6 Pseudo Voting Method in Search Scheme

Many existing QA systems exploit global information about answer candidates. Especially redundancy is the most basic and important information. For example, there are previous studies that boost the score for answers that occur multiple times in documents[2, 18]. The method is known as 'voting method'. Web search engines are also utilized for obtaining redundancy information like the number of hits in Web[9].

On the other hand, we cannot exploit a voting method directly in our scheme of searching answers, because it quits search after n-best answers are found. We therefore introduce an approximation of voting method, called *pseudo voting*, as follows. Our method continues searching for answers untile scores of n *different* answer candidates are fixed, in case of finding n-best answers. The system therefore may find other answer candidates that have same surface expression as one of answer candidates whose score has been already fixed. Consequently we can partially use the frequency information of answer candidates by recording all of answer candidates whose scores are fixed in the process of search. Here, we define the pseudo voting score $S^v(AC, L_q)$ for an answer candidate AC as follows:

$$S^v(AC, L_q) = (\log_{10}(freq(AC, AnsList)) + 1) \cdot \max_{L_i} S(AC, L_i, L_q) \quad (14)$$

where $AnsList$ is the list of answer candidates whose scores are fixed.

7 Experimental Result

In this section, we describe experimental results in NTCIR4 QAC2 Subtask 1. We conducted the following experiments under the condition shown in Table 1. The question set for the evaluation consists of the 200 questions prepared by QAC2 task organizers. We also used 200 questions of QAC1 Formal Run in order to develop the system. In experiments, each passage consists of a series of three sentences.

Table 1. Details of System

Morphological analyzer	JUMAN 3.61
Dependency Analyzer	KNP 2.0b6
NE recognizer	SVM-based NE recognizer using SVM-Light
Numerical Expression Extractor	System by Fujihata et al.[3]
Document Database (Knowledge Resource)	Mainichi Shimbun Newspaper Articles in 1998 and 1999 Yomiuri Shimbun Newspaper Articles in 1998 and 1999
Computer	CPU: Xeon (2.2GHz) × 2, Main memory:4GByte (for QA server) CPU: UltraSPARC III Cu (900MHz) × 2, Main memory:8GByte (for search engine)
Language of Implementation	JPerl 5.005_03

7.1 Experiment 1: QAC2 Subtask1 Formal Run

We submitted the outputs of two runs, SYS-1212 and SYS-1112, to the task organizers. Their parameter settings are shown in Tables 2 and 3. The evaluation by the official scorer and the answer set is shown in Tables 4 and 5.

Table 2. Parameters of two formal runs

	a	d	kd	ppd	p
SYS-1212	10	250	30	5	50
SYS-1112	5	50	0	2	10

7.2 Experiment 2: Evaluation of Effectiveness of Proposed Matching Techniques

To evaluate the effectiveness of each matching techniques described in Section 3, we prepare systems in which one of those functions is suppressed and check the accuracy of those systems by MRR[8] and number

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

[8]MRR, or *Mean Reciprocal Rank*, is the average of reciprocal rank of the highest correct answer.

Table 3. Description of Parameters

a:	Number of answers to be searched for. (Note that we only use 5-best answers even if the number is greater than 5.)
d:	Number of documents to be retrieved.
kd:	Number of documents to be retrieved for weighting keywords (N_{kd} in Section 3.2). '0' means no keyword weighting.
ppd:	Maximum number of passages retrieved from one document.
p:	Number of passages to be concerned in retrieved document.

Table 4. Official result of QAC2 subtask 1 (1) (SYS-1212)

Question	Answer	Output	Correct
200	392	1000	147
Recall	Precision	F-value	MRR
0.375	0.147	0.211	0.489

of correct answers in the system output. We also prepare a baseline system that has the naive scoring function based on the distance between an answer candidate and a keyword. The result is shown in Figure 5.

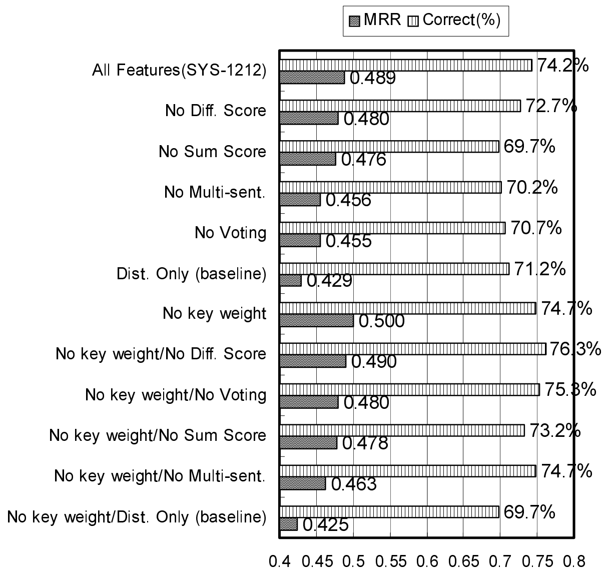

Figure 5. Evaluation of Effectiveness of Proposed Matching Techniques

7.3 Experiment 3: Evaluation of Performance with respect to System Parameters

Since Mori et al.[10] have already reported the effectiveness of the search mechanism controlled by A*, we would like to concentrate upon the relationship between parameter setting and performance including accuracy and turn-around time. In this experiment, we examine MRR and turn-around time with varying the values of parameters in Table 3. The result is shown in Figure 7.3. Note that the turn-around time includes processing time of a (external) search engine.

8 Discussion

In Figure 5 of Experiment 1, we can see that each of the proposed matching techniques except for the keyword weighting has an effect on improving accuracy. Especially, the sentence chaining method described in Section 3.1 is very effective. On the other hand, the keyword weighting method in (4), which has slightly improved the accuracy in QAC1 Subtask 1, makes the accuracy worse. As for the matching of dependency structures, the score based on the 'difference' between dependency vectors is also useful as well as the score based on the 'sum' of dependency vectors, while the latter score is superior to the former one. This means that the distance between a keyword and a answer candidate is not the unique measure to represent the goodness of matching. It is also supported by the low MRR of the system (DIST) in Figure 5, which adopts a naive scoring based on distance between keywords and an answer candidate. Moreover, the pseudo voting method also takes effect to some degree.

As for the parameter setting in Experiment 2, Figure 5 shows a correlation between the accuracy and the number of documents to be retrieved. Roughly speaking, the bigger the parameter 'd' is, the higher the accuracy is. We also have to pay attention to the fact that MRR at d=400 is slightly degraded in comparison with one at d=250. One of the reasons is that the probability that sentences in unrelated documents happen to have higher score becomes higher, as the number of documents increases. The parameter 'p', which is the number of passages to be concerned, has the same tendency as the parameter 'd'. As for the parameter 'a', which is the number of answers to be searched for, the setting of a=10 is almost same as one of a=5. It means that the pesade voting method we proposed works well even if the number of answers to be searched for is not much larger than the necessary number of answers. From the viewpoint of both accuracy and turn-around time, the setting a=5/d=250/kd=0/ppd=5/p=50 is the best one in our experiment, at least for QAC2 Formal Run. At the setting, MRR is 0.500 and the average turn-around time is 20.4 second.

Table 5. Official result of QAC2 subtask 1 (2) (SYS-1112)

Question	Answer	Output	Correct
200	392	991	146
Recall	Precision	F-value	MRR
0.372	0.147	0.211	0.451

9 Conclusion

In this paper, in order to improve accuracy of answers, we introduce several techniques to a QA system based on A* search. According to the experiments in NTCIR4 QAC2 Subtask 1, we confirm that the systems using those techniques achieves a higher MRR

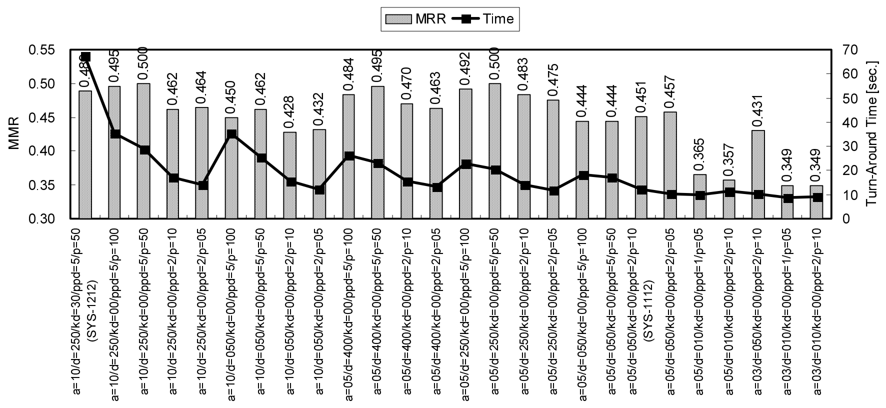

Figure 6. Evaluation of System Performance with respect to System Parameters

than the previous our system. MRR is 0.489 in the QAC3 formal run and 0.500 in the parameter-tuned run after the formal run.

In our future works, we would like to consider 1) introduction of caching mechanism to reuse the result of time-consuming sub-processes, 2) further improvement of accuracy including refinement of sentential matching, 3) optimize the weight of each score, and so on.

10 Acknowledgment

We are grateful to the task organizers of NTCIR4 QAC2 and people who manage NTCIR workshops.

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] C. L. Clarke, G. V. Cormack, and T. R. Lynam. Exploiting redundancy in question answering. In *Proceedings of SIGIR '01: the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 358–365, 2001.

[3] 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).

[4] S. M. Harabagiu, D. I. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. C. Bunescu, R. Girju, V. Rus, and P. Morarescu. Falcon: Boosting knowledge for answer engines. In *Proceedings of the ninth Text Retrieval Conferene (TREC 9)*, pages 479–488, 2000.

[5] 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.

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

[7] D. Kuwahara, N. Kaji, and S. Kurohashi. Question and answering system based on predicate-argument matching. In *Working Notes of the Third NTCIR Workshop meeting – Part IV: Question Answering Challenge (QAC1)*, pages 21–24, 2002.

[8] S. Lee and G. G. Lee. SiteQ/J: A question answering system for Japanese. In *Working Notes of the Third NTCIR Workshop meeting – Part IV: Question Answering Challenge (QAC1)*, pages 31–38, 2002.

[9] B. Magnini, M. Negri, and R. P. H. Tanev. Is it the right answer? exploiting web redundancy for answer validation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pages 425–432, 2002.

[10] T. Mori, T. Ohta, K. Fujihata, and R. Kumon. An A* search in sentential matching for question answering. *IEICE Transactions on Information and Systems*, E86-D(9):1658–1668, Sept. 2003. Special Issue on Text Processing for Information Access.

[11] M. Murata, M. Utiyama, and H. Isahara. Question answering system using similarity-guided reasoning. SIG Notes 2000-NL-135, Information Processing Society of Japan, Jan. 2000.

[12] 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.

[13] 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.

[14] QAC Organizing Committee. NTCIR workshop 3 QA task — Question Answering Challenge (QAC). http://www.nlp.cs.ritsumei.ac.jp/qac/index-e.html, 2002.

[15] 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.

[16] 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. (In Japaese).

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

[18] J. Xu, A. Licuanan, and R. Weischedel. TREC2003 QA at BBN: Answering definitional questions. In *Proceedings of the twelfth Text Retrieval Conferene (TREC 2003)*, 2003.