

## A Monolithic Approach and a Type-by-Type Approach for Non-Factoid Question-answering

— Yokohama National University at NTCIR-6 QAC —

Tatsunori MORI<sup>†</sup> Mitsuru SATO<sup>†</sup> Madoka ISHIOROSHI<sup>†</sup>  
Yugo NISHIKAWA<sup>†</sup> Shigenori NAKANO<sup>†</sup> Kei KIMURA<sup>‡</sup>

<sup>†</sup> Graduate School of Environment and Information Sciences

<sup>‡</sup> Division of Electrical and Computer Engineering, Faculty of Engineering  
Yokohama National University

79-7 Tokiwadai, Hodogaya, Yokohama 240-8501, Japan

{mori,mitsuru,ishioroshi,yugo,s\_nakano,kei\_kimura}@forest.eis.ynu.ac.jp

### Abstract

*In order to process non-factoid questions in NTCIR-6 QAC, we introduced two types of approaches. First one has a monolithic architecture that retrieves answer passages related to a question using lexical chain. The other one has a type-by-type architecture and consists of four subsystems: i) the subsystem for definitional and other-type questions, ii) the subsystem for why-type questions, iii) the subsystem for how-type questions, and iv) the subsystem for factoid questions. Each of two approaches is based on a common hypothesis that the appropriateness of answer candidate for non-factoid questions can be measured by the combination of a) appropriateness of writing style and b) relevance to the question.*

**Keywords:** Non-factoid question-answering, lexical chain, lexico-syntactic patterns.

### 1 Introduction

In recent years, the *question answering* (QA) has gained attention as a way of information access to a large amount of text. QA is the technology that extracts answers for user's natural-language question from a knowledge resource, i.e., a large amount of text. It is expected that QA systems can answer to not only factoid questions, which are well studied, but also non-factoid questions like questions of definition type, why type, how type, and so on.

In order to process non-factoid questions in NTCIR-6 QAC, we introduced two types of approaches. First one has a monolithic architecture that retrieves answer passages related to a question using lexical chain. The other one has a type-by-type architecture and consists of four subsystems i) the subsystem for definitional and other-type questions, ii) the subsystem for why-type questions, iii) the subsystem for how-type questions, and iv) the subsystem for factoid questions. Each of two approaches is based on a common hypothesis that the appropriateness of answer candidate for non-factoid questions can be measured by the combination of a) appropriateness of writing style and b) relevance to the question.

This paper reports the architectures and the experimental results in NTCIR-6 QAC.

### 2 Related work

Han et al.[3] proposed a probabilistic model for definitional question-answering. The model is very suggestive for the methodology of answering non-factoid questions. It is the combination of the following three models: 1) general language model, 2) topic language model, and 3) definition language model. The last two model would be essential to answering definitional questions. The topic language model corresponds to relevance of answer candidates to the question. On the other hand, definition language model represents the appropriateness of writing style in terms of definition.

Most of previous studies follow the scheme. Some proposals utilize lexico-syntactic patterns to find answer candidates in terms of the appropriateness of writing style[2, 7]. To complement topic-related words, Kosseim et al.[4] proposes a method to extract from Wikipedia a list of interest-marking term and use them to score answer candidates. Summarization techniques are also used to reduce redundancy in answer candidates[1].

Our approach is basically a combination of major techniques in the previous studies including 1) lexico-syntactic patterns to find the key sentence, 2) retrieval of variable-length passage, 3) summarization based on clustering passages. On the other hand, the main contributions of this paper are summarized as follows: a) query expansion using the Web to improve the coverage of document retrieval, b) we introduced a passage retrieval method based on lexical chains to find appropriate extent of related description, c) to detect topic-related lexical chains only, we introduce a method to find interest-marking words and eliminate unrelated chains, d) we also propose a unified score for answer candidates calculated from document score, passage score, and score of writing style.

### 3 Overview of our systems for NTCIR-6 QAC

As described for the definitional question-answering by Han et al.[3], we also hypothesize that the appropriateness of answer candidate for non-factoid questions can be measured by the combination of the following two measures.

**Appropriateness of writing style:** how appropriate is the writing style of the candidate in terms of

the type of the given question?

**Relevance to the question:** how relevant is the candidate to the “topic” of the question?

By the first measure, each text segment like sentence is evaluated in terms of writing style of answers corresponding to the type of given question. For example, the Japanese sentence (1) describes the definition of “NPO-hou (NPO-law),” and it includes an appropriate answer candidate for the question (1). In the sentence (1), the combination of the case marker TO and the topic marker WA together with the copula DE-SU is one of writing styles to express a definition. In this paper, we call sentences including such key phrases *key sentences*.

- (1) a. NPO-hou-to-wa tokutei-hieiri-katsudou-  
NPO-law-TO-TOP specified-nonprofit-activity-  
sokushin-hou-no tsuushou-de-su  
promotion-law-REL popular name-BE-POL  
The NPO law is the popular name of the law to promote specified nonprofit activities.
- b. NPO-hou-to-wa nan-de-su-ka  
NPO-law-TO-TOP what-BE-POL-INTERROG  
What is the NPO-law?

Since the description surrounding the key sentence may have some relation to the key sentence, it has to be extracted along with the key sentence. The extent of the description may vary according to the given question and the key sentence, which may have topic-related information more than the question. We expect that the extent can be determined by the second measure described above.

### 3.1 Overview of the system with a monolithic architecture

According to the hypothesis described above, we developed a system shown in Figure 1 (a), which corresponds to the system ID *forst1*. The system has a monolithic architecture, and processes all types of questions including factoid questions uniformly<sup>1</sup>.

With regard to the appropriateness of writing style, we manually constructed a set of lexico-syntactic patterns that match with key sentences and assign score of appropriateness to matched sentences. With regard to the relevance to the question, we propose a method to retrieve passages related to topic using lexical chain. It is a modification of the method proposed by Mochizuki et al.[5]. The retrieval method also assigns relevance score to each passage. A set of answer candidates consists of not only the extracted passages but also the key sentences. Each answer candidate is assigned *score of answer*, which is a combination of the score of appropriateness and the score of relevance as well as the score of document retrieval, in order to select “good” answer candidates.

The process is summarized as follows.

**Step 1:** Classify the input question into a type of question using a set of manually constructed lexico-syntactic patterns for question classification. Each of pattern detects an interrogative and a question focus in a question and classifies the question into one of types shown in Table 1 when the pattern is matched. In the table, *other* represents the type of non-factoid questions other

than definition, why and how. It should be noted that the questions of the type *factoid* are treated as the type of *other* in the system with a monolithic architecture.

**Step 2:** Analyse the question to extract keywords that are content-bearing words or compound nouns, which are series of noun phrases and undefined words. When the question type is *definition*, also extract a *question target*, which is a topic noun phrase, and the *head* of the target, which is the last Bunsetsu segment of the topic noun phrase. With regard to *why*, *how*, and *other*, extract a *clue predicate*, which is a last verb or a last adjective.

**Step 3:** Retrieve documents with keywords and additional keywords. The additional keywords are obtained from the result of Web search. (See Section 4.)

**Step 4:** According to the type of the given question, Apply a set of manually constructed lexico-syntactic patterns of writing styles to each sentence in retrieved documents. Each sentence is given a score of appropriateness that is associated with a matched pattern. (See Section 5.1.)

**Step 5:** Filter out irrelevant documents according to the document score and the sum of appropriateness score of sentences in the document. (See Section 5.2)

**Step 6:** From remaining documents, retrieve passages related to the topic of question using the information about lexical chains related to the topic, and assign a relevance score to each retrieved passage. (See Section 5.3)

**Step 7:** Calculate a unified score for each answer candidate by combining the score of appropriateness, the score of relevance, and the score of document retrieval. Then, select a set of final answer candidates by using clustering-based summarization technique. (See Section 5.4)

Table 1. Types of question

Name of type	Examples
definition	“...-to-wa nan-desu-ka”, “...-te nani” (what is ...)
why	“naze ...-desu-ka” (why ...) “... riyuu-wa nan-desu-ka” (what is the reason for ...)
how	“... dou su-reba yoi-desu-ka” (how do I do ...) “... yari-kata-wa nan-desu-ka” (what is the way to do ...)
factoid	“...-wa dare-desu-ka” (who is ...) “...-wa doko-desu-ka” (where is ...)
other (other type)	“...-wa dou-desu-ka” (how about ...)

*factoid* is treated as *other* in the system with a monolithic architecture.

### 3.2 Overview of the system with a type-by-type architecture

It is possible that the two measures, namely appropriateness of writing style and relevance to the ques-

<sup>1</sup>Lexico-syntactic patterns to find key sentences are dependent on question types.

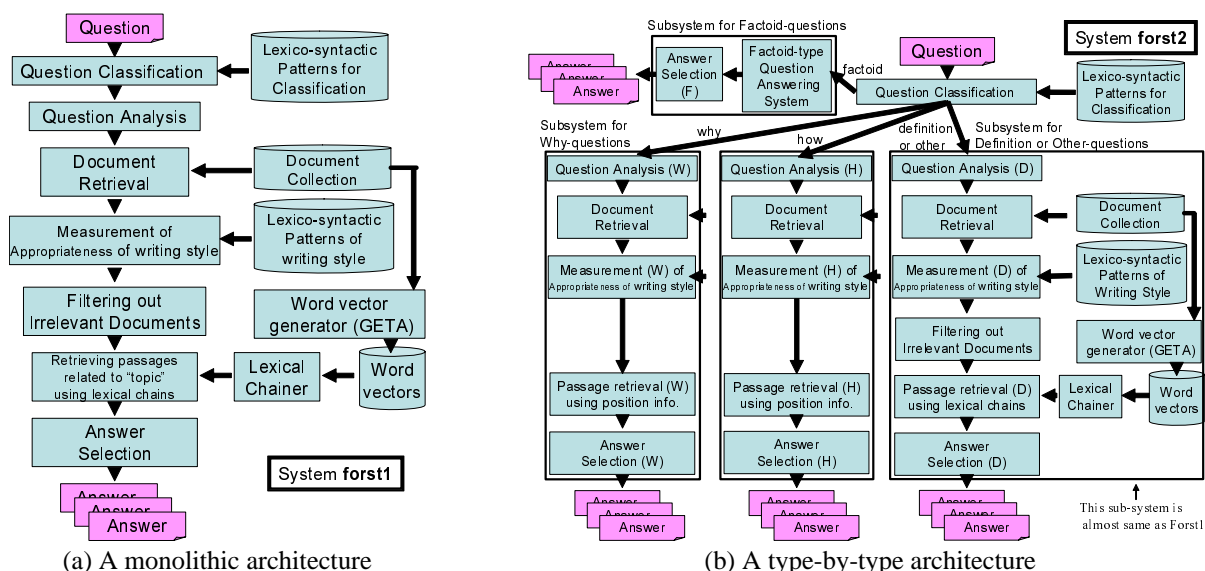


Figure 1. Overview of systems

tion, heavily depend on the question types. Therefore, we also developed another system shown in Figure 1 (b), which corresponds to the system ID *forst2*. It has a type-by-type architecture and consists of four subsystems i) the subsystem for definitional and other-type questions, ii) the subsystem for why-type questions, iii) the subsystem for how-type questions, and iv) the subsystem for factoid questions. Given questions are dispatched to one of these subsystems by the module of question classification.

The subsystem for definitional and other-type questions is almost same as the system with the monolithic architecture. On the other hand, the subsystems for how-type questions and why-type questions have different components from the the system with the monolithic architecture. Especially they do not utilize the information of lexical chain. In stead of it, they use the information of relative position between a key sentence and an answer passage. Here, we expect that the relative position depends on the writing style of key sentences. With regard to factoid questions, we adopt an existing factoid-type question-answering system[6].

#### 4 Document retrieval with query expansion using Web — A Common method used for both of architectures

The search engine used for document retrieval is a ordinary system, which is based on the  $tf \cdot idf$  method for term weighting and the vector space model for calculating similarity between keywords and a document.

With regard to definitional questions, the number of keywords is too small to retrieve relevant documents. We also take account of the difference of vocabularies of questions and document collection. In order to cope with the problem and improve the recall of documents, we introduce a way of query expansion using the result of Web search engine. The procedure is as follows.

**Step 1:** Retrieve the top- $N_{snippet}^{expand}$  snippets by submitting keywords to a Web search engine to perform

the boolean “AND” search. ( $N_{snippet}^{expand} = 50$  and the search engine “goo” is used in the experiment)

**Step 2:** Extract words from snippets using a morphological analyser, then calculate the following “word score” of each word and select top- $N_w^{expand}$  words as additional keywords:

$$score_{word}(w) = freq_{snippet}(w) \cdot IDF_{DB}(w) \quad (1)$$

where  $freq_{snippet}(w)$  is the frequency of the word  $w$  in the snippets, and  $IDF_{DB}(w)$  is the IDF value of  $w$  in terms of the document collection. ( $N_w^{expand} = 10$  in the experiment)

**Step 3:** Retrieve documents by the following operations: a) retrieve the top- $N_{doc}^{expand}$  documents by using the query that consists of original keywords ( $N_{doc}^{expand} = 250$  in the experiment), b) retrieve the top- $N_{doc}^{expand}$  documents by performing “AND search” with original keywords, c) retrieve the top- $N_{doc}^{expand}$  documents by using the query that consists of original keywords with the weight 1.0 and additional keywords with the weight 0.5.

**Step 4:** Merge retrieved documents according to the document score  $score_{doc}(D_i)$  of the document  $D_i$  that is calculated with the similarity  $sim(Q, D_i)$  between the vector of the question  $Q$  and the vector of the document  $D_i$  as follows:

$$score_{doc}(D_i) = \frac{w(D_i) \cdot sim(Q, D_i)}{\max_d \{w(d) \cdot sim(Q, d)\}} \quad (2)$$

$$w(D_i) = \begin{cases} 1.0 & \text{obtained by (a)} \\ 2.0 & \text{by (b) or (c)} \end{cases} \quad (3)$$

### 5 The monolithic architecture

#### 5.1 Measurement of appropriateness of writing style

In this step, each sentence of retrieved documents is assigned a score of appropriateness in terms of writ-

ing style corresponding to the type of the given question. It is performed by using a set of manually constructed lexico-syntactic patterns that match with key sentences. Each pattern has a score of appropriateness, which is a real number between 0 and 1. If a pattern is matched with a sentence  $S_i$ , then the sentence receives the pattern's score  $score_s^{style}(S_i)$  as its estimated appropriateness. When multiple patterns are applicable to one sentence, the score of sentence is the maximum score of patterns.

With regard to definition-type questions, each pattern is generated from one of *templates* that has a *slot* which will be filled by a question target. If there are no sentences that match with any of the filled patterns, then the templates are combined with the head of the target instead of the target itself to make patterns more relaxed.

With regard to how type or other type questions, instead of the application of patterns, it is examined whether compound nouns or clue predicates in the question appear in the sentences. If a sentence  $S_i$  has a compound noun  $CN_j = w_1^j w_2^j \dots w_n^j$ , then the sentence will receive the following score:

$$score_{cn_s}^{style}(CN_j) = \sum_{w_i^j \in CN_j} \frac{IDF(w_i^j)}{100.0} \quad (4)$$

On the other hand, when the sentence has a clue predicate  $CP_k$  in the question, the score

$$score_{cp_s}^{style}(CP_k) = 0.2 \quad (5)$$

is given to the sentence. Finally, the score will be as follows:

$$score_s^{style}(S_i) = \sum_{CN_j \in S_i} score_{cn_s}^{style}(CN_j) + \sum_{CP_k \in S_i} score_{cp_s}^{style}(CP_k) \quad (6)$$

## 5.2 Filtering out irrelevant documents

Since the calculation of lexical chains is a time-consuming process and it depends on the amount of documents to be processed, it is preferable that irrelevant documents are filtered out before the calculation. Therefore, we define the following another document score  $score_{doc}^{style}(D_i)$  that takes account of a score of appropriateness in terms of writing style. Documents are re-ranked with the score, and the top- $N_{doc}^{style}$  documents is selected. ( $N_{doc}^{style} = 50$  in the experiment)

$$score_{doc}^{style}(D_i) = score_{doc}(D_i) + \sum_{S_j \in D_i} score_s^{style}(S_j) \quad (7)$$

## 5.3 Passage retrieval using lexical chains

A lexical chain is a sequence of semantically related words in a text. It is used for capturing a portion of the cohesive structure of the text. In this paper, we utilize lexical chains in order to retrieve passages related to the topic of question.

### 5.3.1 Detection of lexical chains

To detect lexical chains, we make use of the program "lexical chainers."<sup>2</sup> The program reads a text in the sentence-by-sentence manner. When a sentence is inputted to the program, it finds word clusters in a sentence according to the "similarity" between words. Then, it merge the newly found clusters into a set of clusters that are found for the portion of text so far. The process will iterated until the end of text. Each of generated word clusters  $c_i$  is regarded as a lexical chain. As the similarity between two words, we select the cosine similarity between "word vectors," which are rows of a word-by-document frequency matrix. With regard to a clustering algorithm, the single-link algorithm is adopted and the similarity threshold is  $Th_{word}^{sim}$  ( $Th_{word}^{sim} = 0.25$  in the experiment).

### 5.3.2 Filtering out lexical chains unrelated to questions

A set of lexical chains obtained by the method described above contains not only chains related to question but also unrelated chains, because the method finds lexical chains independent of the question. Therefore, we introduce the following procedure to eliminate unrelated chains and preserve lexical chains that contain interest-marking words.

**Step 1:** Delete lexical chains that do not contain any keywords of the question or additional keywords.

**Step 2:** Delete words other than "words associated with the question" from lexical chains. (See below)

**Step 3:** If two adjacent words in a lexical chain do not locate within two adjacent sentence, divide the chain at that point.

In Step 2, the words associated to the question are gathered according to the similarity relation with a dumping factor. As shown in Figure 2, the process starts with the set of keywords as the set of associated words. If a word in lexical chains has an association relation with one of associated words, that is, if the word has a association score greater than a threshold  $Th_{assoc}$ , the word is added to the set of associated words. The expansion of a set of association words will be performed until no more words are added to the set.

The association score is calculated as follows:

$$score_{assoc}(w_i, w_j) = sim(w_i, w_j) \cdot dumping(level(w_i)) \quad (8)$$

$$dumping(l) = \frac{x}{\exp(x-1)^2} \quad (9)$$

where  $sim(w_i, w_j)$  is the similarity in terms of word vector and  $level(w_i)$  is the association level, or distance of the word from the keywords. The function  $dumping(l)$  represents a dumping factor according to the association level  $l$ .

<sup>2</sup><http://www.jaist.ac.jp/~motizuki/software/chainers/chainers-1.50.2.tar.gz>



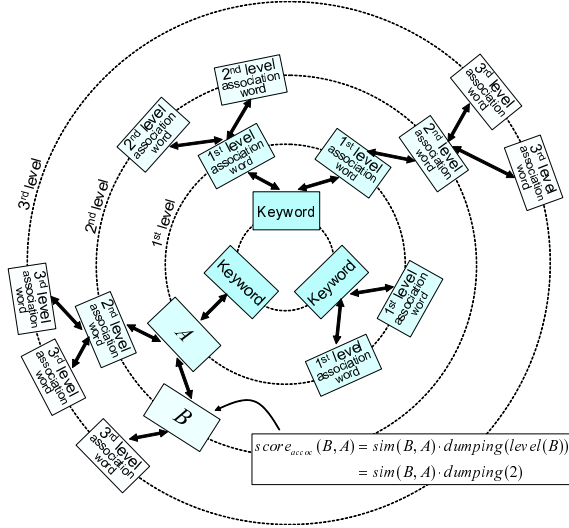


Figure 2. Gathering words associated with the keywords of question

### 5.3.3 Extraction of passage using lexical chains

Mochizuki et al.[5] proposed a method that detects a portion of text as a passage if all sentences of the portion are covered by , at least, one lexical chain. However, the method tends to extract longer passages than expected when the question has more (different) keywords. On the other hand, when we take the approach that extracts a portion of text which is covered by a maximal number of lexical chains, the length of passage tends to be too short. We need more moderate approach.

Therefore, we propose the following method to extract better passages.

**Step 1:** Calculate “density of lexical chain” for each sentence, and select each sentence whose density is maximal. We term the sentence *the center of passages*.

**Step 2:** Determine the extent of passage for each center of passage according to the density of sentences adjacent to the center.

The density of lexical chain for the sentence  $S_j$  is defined as follows:

$$density(S_j) = \frac{|K \cap \bigcup_{c_i \in C(S_j)} c_i|}{|K|} \sum_{c_i \in C(S_j)} \frac{w_{chain}(c_i)}{length(c_i)} \quad (10)$$

$$w_{chain}(c_i) = |c_i| \cdot \min_{w \in C_i} IDF(w) \cdot \sum_{k \in (C_i \cap K)} IDF(k) \quad (11)$$

where  $K$  is the set of keywords in the question,  $C(S_j)$  is a set of lexical chains that cover the sentence  $S_j$ ,  $length(c)$  is the number of sentences that are covered by the chain  $c$ .

Each sentence that has a maximal density will be one of centers of passages. The center and some sentences adjacent to the center form one passage. Basically, we also use the density to determine the extent

of passage, but passage detection using the raw value of density may generate very long passage. To cope with the problem, we introduce a window function to produce passages with a predefined length. The following equation represents the adjustment of density in terms of the center of passage  $S_c$ .

$$density_{win}(S_j, S_c) = density(S_j) \cdot \frac{L^2 - |j - c|^2 / 2}{L^2} \quad (12)$$

where  $L$  is a parameter that controls the window size ( $L = 4$  in the experiment).

We adopt the extent from the sentence  $S_s$  to the sentence  $S_e$  as a passage in terms of the center  $S_c$ . The extent is defined as follows:

$$s = \min\{j | j \leq \forall k \leq c \quad (density_{win}(S_k, S_c) \geq R_d \cdot density(S_c))\} \quad (13)$$

$$e = \max\{m | c \leq \forall k \leq m \quad (density_{win}(S_k, S_c) \geq R_d \cdot density(S_c))\} \quad (14)$$

where  $R_d$  is a parameter that controls the length of passages ( $R_d = 0.5$  in the experiment).

Using the density, we define the score  $score_p(P_{s,e})$  of a passage  $P_{s,e}$ , which consists of the sentences from  $S_s$  to  $S_e$ , as follows:

$$score_p(P_{s,e}) = \frac{\max_{S_i \in P_{s,e}} density(S_i)}{\max_{P \in P_{all}} \max_{S_j \in P} density(S_j)} \quad (15)$$

where  $P_{all}$  is the set of all retrieved passages.

## 5.4 Answer selection

A set of answer candidates consists of not only the extracted passages but also the key sentences. The process of answer selection chooses “good” answer candidates by taking account of increasing importance and reducing redundancy, simultaneously.

### 5.4.1 Scoring answer candidates

The score of answer candidate is a combination of 1) document score, 2) score of appropriateness of writing style, and 3) passage score, as follows:

$$score_{AC}(AC) = \alpha \cdot score_{doc}(doc(AC)) + \beta \cdot score_p(AC) + \gamma \cdot score_{AC}^{style}(AC) \quad (16)$$

where  $doc(AC)$  is the document that  $AC$  belongs to, and the parameters  $\alpha$ ,  $\beta$ , and  $\gamma$  control the ratio of mixture of score.

$$(\alpha, \beta, \gamma) = \begin{cases} (0.5, 0.5, 1) & \text{definition type} \\ (1, 1, 1) & \text{otherwise} \end{cases} \quad (17)$$

The score  $score_{AC}^{style}(AC)$  is calculated for the definition or why types as follows:

$$score_{AC}^{style}(AC) = \max_{S_i \in AC} score_s^{style}(S_i) \quad (18)$$

and for other types as follows:

$$score_{AC}^{style}(AC) = \max\left\{ \max_{CN_j \in AC} score_{cn_s}^{style}(CN_j), \max_{CP_k \in AC} score_{cp_s}^{style}(CP_k) \right\} \quad (19)$$

where  $CN_j$  and  $CP_k$  are a compound noun and a clue predicate in the question.

### 5.4.2 Reducing redundancy

Since answer candidates come from different documents, they may have redundant information. This situation is the same as multi-document summarization. In order to reduce the redundancy, several researches like Stein et al.[9] and Radev et al.[8] utilize the document clustering to detect redundancy. We also introduce a similar type of redundancy control mechanism that is based on the document clustering as follows:

**Step 1:** Cluster answer candidates with the threshold  $Th_{sim}^c$  of similarity.

**Step 2:** Select a representative answer candidate for each cluster, and calculate the modified score for the representative by taking account of the size of cluster.

The adopted clustering algorithm is an ordinary agglomerative complete-link clustering. The similarity between answer candidates is defined as the cosine similarity between the word-frequency vectors of answer candidates. If the similarity between two clusters is larger than the threshold  $Th_{sim}^c$ , then these clusters are merged. ( $Th_{sim}^c = 0.7$  in the experiment.)

The representative answer candidate for a cluster is the candidate whose score is largest. The final answer score is the modified version of the answer score as follows.

$$score_{AC}^{weighted}(AC \in CL) = score_{AC}(AC) \cdot (1 + \frac{1}{4} \log_{10} |CL|) \quad (20)$$

## 6 A type-by-type architecture

### 6.1 The subsystem for definitional and other-type questions

The system is almost same as the system with the monolithic architecture. However, it employs a different strategy to determine the extent of a passage in the following way. First, each passage has to start with a key sentence. Second, the extent of passage  $P_{s,e}$  is calculated as follows:

$$\begin{aligned} s &= c \\ e &= \max\{m | c \leq \forall k \leq m \\ &\quad (density_{win}(S_k, S_c) \geq \\ &\quad R_d \cdot density(S_c))\} \end{aligned} \quad (21)$$

where  $c$  is the index of key sentence. The parameter  $L$  for the window function (12) is 5, and  $R_d = 0.5$  in the experiment.

### 6.2 The subsystem for why-type questions

This subsystem finds passages of fixed length that include a key sentence. The length of passages is three in the experiment. The relative position between the key sentence and the passages is determined according to the writing style of the key sentence, namely, the matched lexico-syntactic patterns of writing style.

The score of passage  $score_p^{why}(P_i)$  is calculated by the following equations:

$$score_p^{why}(P_i) = \alpha \cdot score_s^{style}(S_j)$$

$$\begin{aligned} &+ \beta \cdot score_{doc}^{why}(D_h) \\ &+ \gamma \cdot score_s^{why}(P_i) \end{aligned} \quad (22)$$

$$score_{doc}^{why}(D_h) = \frac{\sum_{k \in HLine(D_h) \cap K} IDF(w)}{\max_{k' \in K} IDF(k')}$$

$$score_s^{why}(P_i) = \frac{\sum_{k \in P_i \cap K} IDF(w)}{\max_{k' \in K} IDF(k')}$$

where  $S_j$  is a key sentence in the passage  $P_i$  of the document  $D_h$ ,  $K$  is a set of keywords,  $HLine(D_h)$  returns the headline of  $D_h$ , and  $(\alpha, \beta, \gamma) = (0.35, 0.13, 0.52)$  in the experiment.

### 6.3 The subsystem for how-type questions

Since the how-type question requires a way to do something, the predicate, or the verb phrase, in the question that corresponds to “do something” is expected to play an important role in the how-type question. We term the predicate *focused predicate*.

From the question, this subsystem extracts the relative clause that contains the predicate. Then, the system produces several types of paraphrases of the relative clause that are expected to appear with answer candidates in documents. These paraphrases are used as lexico-syntactic patterns of writing styles. Sentences that are “similar” to the paraphrases are detected as key sentences. The similarity is calculated based on the number of keywords in the sentence, whether the sentence contains the focused predicate or not, and the surface similarity between the sentence and the paraphrase.

The system finds passages of fixed length that include a key sentence. The length of passages is three in the experiment. Since the relative position of answer candidate to the paraphrase is usually fixed, the system may find the position of answer candidate according to the paraphrase that matched with the key sentence.

## 7 Experimental results and discussion

In NTCIR-6 QAC, we submitted two runs, *forst1* and *forst2*, as the formal run. These run IDs correspond to the system with the monolithic architecture and the system with the type-by-type architecture.

Before we examine the performance in terms of question answering, let us check a basic information about question answering. The average number of answers per question is shown in Table 2. It should be noted that the questions of the type *factoid* are treated as the type of *other* in the system *forst1*. According to the table, the system *forst1* seems to generate about six answers per question. On the other hand, *forst2* is three. However, it also should be noted that this is the result of cutting off the lower ranked answers requested by the task organizers. Originally a little bit more answers are generated. Since answer candidates includes a number of incorrect answers, we have to have a better mechanism to control the number of answers.

Table 3 and 4 show the number of answers with judgment generated by System *forst1* and System *forst2*, respectively. These tables show the performance of question answering in answer-by-answer

**Table 2. Average number of answers per question**

Detected Q. type	forst1	forst2
def.	4.57	4.05
why	6.44	3.09
how	7.00	5.00
other	6.12	6.12
factoid (other)	6.06	1.91
All	5.93	3.26

**Table 3. Number of answers with judgment generated by System forst1**

Detected Q. type	Judgment				Total
	A	B	C	D	
def.	19	21	6	50	96
why	10	35	11	150	206
how	0	5	9	28	42
other	3	9	3	34	49
other (factoid)	13	36	5	146	200
All	45	106	34	408	593

**Table 4. Number of answers with judgment generated by System forst2**

Detected Q. type	Judgment				Total
	A	B	C	D	
def.	20	16	7	42	85
why	5	24	5	65	99
how	0	4	2	24	30
other	3	9	3	34	49
factoid	2	0	4	57	63
All	30	53	21	222	326

manner. Both of systems have relatively better performance for definition-type questions than other types. Table 4 shows that the system forst2 finds few correct answers for the questions that are detected as factoid-type. As described below, it is caused by the failure of question classification. The many of non-factoid questions are incorrectly classified into the type of factoid.

Table 5 and 6 show the number of questions to which the systems, forst1 and forst2 respectively, reply at least one answer of the judgment.

**Table 5. Number of questions to which the system replies at least one answer at the judgment (forst1)**

Detected Q. type	Judgment					Total
	A	A +B	A +C	A +B +C	D only	
def.	10	18	10	18	3	21
why	7	20	9	22	10	32
how	0	3	2	5	1	6
other	3	7	3	7	1	8
other (factoid)	11	19	13	21	12	33
All	31	67	37	73	27	100

**Table 6. Number of questions to which the system replies at least one answer at the judgment (forst2)**

Detected Q. type	Judgment					Total
	A	A +B	A +C	A +B +C	D only	
def.	11	16	12	17	4	21
why	3	14	6	17	15	32
how	0	1	2	3	3	6
other	3	7	3	7	1	8
factoid	2	2	6	6	27	33
All	19	40	29	50	50	100

As shown in the tables, System forst1 finds at least one correct answer(A) or one partially correct answer (B,C) for 73% of questions. There is no big difference between forst1 and forst2 in definition-type, why-type, how-type, and other-type. On the other hand, we can see that the system forst2 fails to find correct answers for the questions that are detected as factoid-type.

In the evaluation using the last two tables, the rank of correct answer candidate are not taken account of. In order to examine the ability of ranking answer candidate, namely, the effectiveness of scoring method, the type-by-type mean reciprocal rank of each system is shown in Table 7. The table shows that the scoring method of the monolithic architecture produces generally better ranking than that of type-by-type architecture. Table 8 shows the type-by-type mean precision of each system.

According to the results described above, the system based on monolithic approach has better performance than that of the type-by-type approach. However, it is mainly due to the failure on the question-type classifier, which is used in both of approaches.

**Table 7. Mean reciprocal rank**

Detected Q. type	Judgment of correct answer			
	forst1		forst2	
	A	A+B+C	A	A+B+C
def.	0.383	0.742	0.406	0.698
why	0.109	0.405	0.078	0.484
how	0.000	0.597	0.000	0.417
other	0.175	0.452	0.175	0.452
factoid (other)	0.211	0.485	0.020	0.111
All	0.199	0.517	0.131	0.400

**Table 8. Mean of precision**

Detected Q. type	Judgment of correct answer			
	forst1		forst2	
	A	A+B+C	A	A+B+C
def.	0.226	0.582	0.222	0.521
why	0.051	0.281	0.039	0.348
how	0.000	0.354	0.000	0.200
other	0.064	0.298	0.064	0.298
factoid (other)	0.074	0.276	0.020	0.086
Total	0.093	0.348	0.071	0.285

The performance of the question-type classifier is described in Table 9. As it shows, the many of non-factoid questions are incorrectly classified into the type of factoid. The system *forst2* is affected by the failure. On the other hand, the system *forst1* is not affected because it treats all questions as non-factoid type even if the questions are detected being of the factoid type.

**Table 9. Performance of question classification (forst1 and forst2)**

Actual Q. type	Detected Q. Type				
	def.	why	how	other	factoid
def.	16	0	0	0	0
why	0	30	1	0	2
how	0	0	5	0	0
other	1	2	0	8	23
factoid	0	0	0	0	1
def. or other	3	0	0	0	4
def. or other or factoid	1	0	0	0	0
other or factoid	0	0	0	0	2
how or other	0	0	0	0	1
Total	21	32	6	8	33
Recall	1.00	0.91	1.00	0.24	1.00
Precision	$\frac{16}{21}$	$\frac{30}{32}$	$\frac{5}{6}$	$\frac{8}{8}$	$\frac{1}{33}$
	0.95	0.94	0.83	1.00	0.09
	$\frac{20}{21}$	$\frac{30}{32}$	$\frac{5}{6}$	$\frac{8}{8}$	$\frac{3}{33}$

## 8 Conclusion

In order to process non-factoid questions in NTCIR-6 QAC, we introduced two types of approaches. First one has a monolithic architecture that retrieves answer passages related to a question using

lexical chain. The other one has a type-by-type architecture and consists of four subsystems i) the subsystem for definitional and other-type questions, ii) the subsystem for why-type questions, iii) the subsystem for how-type questions, and iv) the subsystem for factoid questions.

The experimental results in NTCIR-6 QAC show that the system based on the former approach has better performance than that of the latter approach. However, it is mainly due to the failure on the question-type classifier, which is used in both of approaches. The many of non-factoid questions are incorrectly classified into the type of factoid. The latter system is affected by the failure.

In order to achieve better performance, we need a more precise question classification method that can classify questions into not only non-factoid types but also sub-types of factoid.

## Acknowledgment

The authors would like to express their thanks to Yoshiaki Asada, who developed the earlier version of our system. This study was partially supported by Grant-in-Aid for Scientific Research on Priority Areas (No.18049031 and No.19024033) from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

## References

- [1] S. Blair-Goldensohn, K. McKeown, and A. Schlaikjer. A hybrid approach for QA track definitional questions. In *Proceedings of TREC 2003*, 2003.
- [2] A. Fujii and T. Ishikawa. Organizing encyclopedic knowledge based on the Web and its application to question answering. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-2001)*, pages 425–432, 2002.
- [3] K.-S. Han, Y.-I. Song, and H.-C. Rim. Probabilistic model for definitional question answering. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 212–219, 2006.
- [4] L. Kosseim, A. Beaudoin, A. Keighbadi, and M. Razmara. Concordia university at the trec 15 qa track. In *Proceedings of TREC 2005*, 2005.
- [5] H. Mochizuki, M. Iwayama, and M. Okumura. Passage-level document retrieval using lexical chains. In *Proceedings of RIAO 2000*, pages 491–506, 2000.
- [6] T. Mori. Japanese question-answering system using A\* search and its improvement. *ACM Transactions on Asian Language Information Processing (TALIP)*, 4(3):280–304, 2005.
- [7] K. Morooka and J. Fukumoto. Answer extraction method for why-type question answering system. NLC NLC2005-107, IEICE, Feb. 2006.
- [8] D. R. Radev, H. Jing, and M. Budzikowska. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proceedings of the ANLP/NAACL Workshop on Automatic Summarization*, 2000.
- [9] G. C. Stein, T. Strazalkowski, and G. B. Wise. Summarizing Multiple Documents using Text Extraction and Interactive Clustering. In *Proceedings of the sixth Pacific Association for Computational Linguistics (PACLING 99)*, pages 200–208, 1999.