SiteQ/J: A Question Answering System for Japanese

Seungwoo Lee Gary Geunbae Lee
Dept. of Computer Science & Engineering, POSTECH
San 31, Hyoja-Dong, Pohang, South Korea, 790-784
{pinesnow, gblee}@postech.ac.kr

Abstract

This paper describes our Question Answering system participated in QAC Task1 of NTCIR3 and reports the results with some observations. Through analyzing the previous TREC QA data, we defined passage and developed passage selection method suitable for Question Answering. Using Lexico-Semantic Patterns (LSP), we identify answer type of a question and detect answer candidates without any deep linguistic analysis of the texts. Answer candidates are ranked by passage scores and distances between answer candidates and matched terms. As a result of better engineering, our system showed excellent performance when evaluated by mean reciprocal rank (MRR) in NTCIR 3.

Keywords: question answering, passage selection, lexico-semantic patterns

1. Introduction

Question Answering (QA) is a task to retrieve answers rather than documents in response to a question and an effort to come nearer to true information retrieval. Since the test collection was prepared through TREC-8 [16], many researches were progressed vigorously. We also jumped into this task, participated in the last TREC QA task (TREC-10) and achieved relatively good performance [10]. Encouraged by the result, we decided to participate in the QAC task of NTCIR 3 (http://research.nii.ac.jp/ntcir/workshop/qac/cfp-en.html) and apply the similar but more advanced techniques to Japanese Question Answering.

Almost all QA systems use document retrieval systems and even passage retrieval techniques to reduce search space to small document snippets. Passage retrieval techniques were originally introduced to improve the precision of document retrieval systems. However, in the view of question answering, passage retrieval may show different characteristics. Usually the goal of question answering task is to find out an exact answer (or answer string) to a question, whereas the goal of document retrieval task is to find out as many relevant documents as possible to a requested topic. Relevance information to a topic tends to occur across several sentences in a document, but query terms of question and its answer tend to occur within one sentence or two. Thus we need to define new passage and develop a method suitable for question answering systems

One of main differences between the previous TREC QA tasks and NTCIR QAC task is exact answer constraint on returned answers. Returned answers containing any dummy word will be judged as incorrect. This requires methods that can determine the answer type of a question and detect exact answers belonging to the answer type more accurately. These are achieved by using Lexico-Semantic Patterns in our system.

The remainder of this paper is organized as follows. In section 2, we look into some previous researches on Question Answering. We explain how passages are defined and selected for further processing in section 3. In section 4, we describe what Lexico-Semantic Patterns are, how answer type of a question is determined, and how answer candidates are detected using these Lexico-Semantic Patterns and some scores.

2. Previous Work

Many question-answering systems have been developed since TREC QA test collection was constructed. They employed various techniques to answer a question from the test collection. We will review some of them in this section.

Identification of answer type of a question was used to constraint semantic type of potential answers and can narrow search space considerably. Answer type was defined based on ontology such as WordNet [7] or Named Entity labels used in MUC-7 [8]. [7] identified answer type of a question through parsing, while [8] identified through learning several features based on Maximum Entropy model.

Another technique to narrow search space for potential answers is passage retrieval (or selection). Passage retrieval techniques were initially developed for high precision of information retrieval systems [2][6][13][19][21] and also used for selecting passage that might contain potential answers [3][7][11]. Many QA systems defined their own passage (sentence, paragraph, topical segment, etc) and developed various ranking measures. However, it was not yet evaluated which passage definition and ranking method is most effective for Question

Answering systems.

To detect answer candidates to a question, many systems utilized Named Entity recognition techniques. This was achieved by pattern matching such as regular expression [3] or external NE taggers [7]. To justify answer candidates, [5] employed abductive reasoning with deep analyses of texts, while [7][14] used only surface patterns and showed good performance in the latest TREC QA evaluation.

3. Query-based Answer Passage Selection

Most question answering systems utilize existing document retrieval systems to reduce search space to several documents. However a document is usually not suitable for detecting answer candidates within itself because it is too long and contains too much extra information and various topics. By analyzing questions and their answers used in the past TREC QA tasks, we could find that an answer occurs comparatively near to keywords matched to a query in a document. This means that we can focus on only short part of a document rather than the whole to extract answer candidates and thereby considerably reduce the computational load. Considering this fact we employ passage selection method after retrieving documents through a probabilistic document retrieval system, POSNIR[10], like several previous researches ([3][7][11]).

3.1 Definition of a Passage

Passage selection or passage retrieval techniques were originally used to improve the precision, especially top-level precision, of information retrieval systems [9]. These techniques can be divided into two different approaches: static and dynamic passage retrieval. In static approach, each document is segmented into several passages in indexing time and each passage, not document, is indexed. Therefore, a retrieved passage is always definite regardless of a request [6][13][19][21].

However, this approach may often degrade the performance to some queries since it is difficult to segment a document into passages equally suitable to all the queries. In contrast to the static approach, the dynamic approach determines the passages with fixed or variable length in response to requests [2][9]. This method can get passages more suitable to each query but needs an algorithm to efficiently determine them since all possible passages must be computed in retrieval time. According to Kaszkiel [9], we know that the dynamic approach gives substantial improvements in effectiveness than the static and so can be more suitable for question answering. Passage was defined in several ways in previous works: sections (or paragraphs) [19][13], pages (which are adjacent paragraphs within limited bytes)[21], tiles (which are adjacent sentences divided by topic shift)[6], fixed-length windows[2] and variable-length windows[9]. Sections, pages and tiles are static passages, whereas fixed-length and variable-length windows are dynamic. Kaszkiel compared these various passages through experiments and showed that passages based on windows were more effective and especially 150 to 350-word passages had good performance in retrieval precision.

However, in the view of question answering, passage retrieval may show different characteristics. We analyzed 492 questions and evaluation results of the runs submitted by each participant group of TREC-10 [18] to investigate the effect of passage retrieval when it was applied to question answering. Using answer patterns and judgment information provided by NIST, we gathered instances of each answer passage which consists of 11 adjacent sentences: one sentence containing an answer string and the previous and the next 10 adjacent sentences. And then we investigate the distribution of query terms in each answer passage.[1] Throughout this analysis, we found that about 80% of query terms of each question occurred on average within 50-word window including answer string in its center and within 3-sentence window including answer string in the middle sentence. This is very small relatively to the window size suggested by Kaszkiel[9] and might be due to the difference between the two tasks. Usually the goal of document retrieval task is to find out as many relevant documents as possible to a requested topic, but the goal of question answering task is to find out an exact answer (or answer string) to a question. Relevance information to a topic tends to occur across several sentences in a document, but query terms of question and its answer tend to occur within one sentence or two.

According to this preliminary examination, we defined a passage as consecutive three sentences for QAC task. We prefer sentence window to word window to prevent a possible answer from being cut off by window boundary even though some query terms and the possible answer co-occur in the same sentence.

3.2 Passage Scoring

We developed a scoring measure to rank each passage and an algorithm to compute efficiently scores of all possible passages. We can first think of the following assumptions for scoring measure of the passages:

- The more query terms a passage contains, the more probably the passage also contains an answer to the question.
- Duplication of terms in a passage or a question is not important and can be ignored.
- Document-specific terms are more important than general terms.

[1] Stop words and optional terms in a query were ignored and lemmatization was applied to each query term except the superlative. For example, in a question, “What is the largest city in the U.S.?”, ‘city’ is optional and therefore was ignored.

The first and second assumptions can be measured by the number of unique query terms occurred in a passage divided by the number of unique query terms and the third can be expressed by using inverse document frequency. Let *qtuc* be count of unique query terms in a question, *ptuc* be count of unique passage terms matched with the query terms, ptf_i be frequency of *i*th query term in a passage (we call this passage term frequency) and idf_i be inverse document frequency of *i*th query term. Then the score of a passage can be calculated using expression (1).

$$PScore = \alpha \times \frac{ptuc}{qtuc} + (1-\alpha) \times \frac{2000}{qtuc} \times \sum_{i=1}^{qtuc} ptf_i \times idf_i \quad (1)$$

, where α is a constant.

3.3 Efficient Passage Selection

Passage selection algorithm must cover all possible passages from each retrieved document to select most probable passages. If a document consists of five sentences (S_1, S_2, S_3, S_4, S_5), there are three possible passages ($P_1=\{S_1, S_2, S_3\}$, $P_2=\{S_2, S_3, S_4\}$, $P_3=\{S_3, S_4, S_5\}$). Then the algorithm must calculate the score of each passage P_1, P_2 and P_3 respectively by expression (1). Since *qtuc* and idf_i are constant regardless of passages, it only needs to count *ptuc* and ptf_i. These can be easily obtained from term position information (TPI) which is a database that contains the position (i.e. *j*th token of *i*th sentence) of each index term occurred in each document and which is constructed in IR indexing time. When a document is retrieved, we can count the occurrences of each query term in each sentence and then count the occurrences of each query term in each passage and calculate the score of each passage.

We rank the passages according to the scores of them and select top *N* passages for further processing.

It is allowed to select more than one passage from a document, but only one passage is allowed to be selected among three consecutive and overlapped passages to prevent one sentence from being processed more than once in the next module. In the above example, if query terms occur in only S_3, the three passages, P_1, P_2 and P_3 have same scores. In this case, we prefer P_2 to others since answers tend to occur near the query terms.

After selecting top passages, answer candidates are extracted from them through Lexico-Semantic Pattern matching and most probable answers are returned as a response to the input question.

4. Question Answering using Lexico-Semantic Patterns

Various Question Answering systems and techniques have been developed and tested through recent TREC QA tracks on the English test collection [15][17][18]. Some systems like [5] showed good performance through complicated analyses of texts such as parsing and theorem proving while some systems like [14] obtained surprising results using only simple surface patterns extensively and showed the remarkable power of such lexical patterns by exploiting redundancy of the corpus.

The first Question Answering Challenge (QAC) task in NTCIR3 has a goal similar to that of the previous TREC QA tasks (that is, uses fact-based questions that require short answers) but requires the exact answer to each question rather than fixed-byte answer strings (TREC2002 QA track also adopts this policy). This exact answer constraint may degrade the performance of QA systems since a returned answer is judged as incorrect if it contains any dummy words. For example, a returned answer "DDI・KDD・IDO" to a question "2000年10月1日に合併することが決まった通信三社はどこですか。(*What are three communication companies which were decided to merge in Oct. 1, 2000?*)" is incorrect though each 'DDI', 'KDD' and 'IDO' is judged as correct. Therefore we need a technique to exactly extract the only entities that can be answers and well-engineered Lexico-Semantic Patterns (LSP) are developed for such purposes.

Instance LSP
…名作/NC は/EH 何/NRC です/U か/EM (*What is the masterpiece …*)
(%work)(は)(何)(です)(か)
会長/NC は/EH 誰/NRC です/U か/EM (*Who is the president …*)
(@position)(は)(誰)(です)(か)
30/NN 歳/NUN (*30 year-old*)
(@number)(@unit_age)
夏目/NPPS 漱石/NPPG の/EY 「/SO こころ/NC 」/SL (*Souseki Natsume's 'Kokoro'*)
(@person)(の)(@bracket)
バイ/NC スフ/NC ロク/NC さん/NUP (*Mr. Baisuhuroku*)
(@np)(NUP)

Table 1. Example of Lexico-Semantic Patterns[2]

4.1 Lexico-Semantic Patterns

In this subsection, we describe Lexico-Semantic Patterns (LSP), which is used to determine the answer type of an input question and also to extract answer candidates from selected passages.

LSP is a pattern that is expressed by lexical entries, part-of-speeches (POS), syntactic categories and semantic categories. Unlike surface patterns, which are expressed literally using only lexical entries, LSP has more flexibility, can reduce the number of necessary patterns and gives the expression power to handle the complex syntactic

[2] Each instance was attached with POS tag, and the parentheses in LSP were inserted only to separate each component of the LSP.

and semantic phenomena in human language since each lexical can be generalized by POS, syntactic or semantic category.

Table 1 shows some examples of LSP's that can be constructed from instances. '*%work*', '*@position*', '*@unit_age*' and '*@person*' are semantic categories of '名作', '会長', '歳' and '夏目漱石', respectively. Of course each number expression is generalized to '*@number*'. '*@bracket*' and '*@np*' are syntactically generalized from '「こころ」' and 'バイスフロク', respectively. '*NUP*' is a POS tag of 'さん' and represents a suffix following person's name.

We use POSTAG/J, our part-of-speech (POS) tagger for Japanese, to generalize each lexical entry to its POS. For syntactic category such as '*@np*' and '*@vp*', we implemented simple verb and noun phrase chunker. '*@bracket*' and '*@parenthesis*' are made by checking the boundaries surrounded by brackets and parentheses within small window and is very useful for extracting titles of books, movies and TV programs and acronyms of entities, respectively. To generalize a lexical entry to a semantic category, we defined 68 semantic categories by referring the previous TREC data and gathered about 250,000 instances belonging to each semantic category from lots of web sites and dictionaries. Semantic categories include person, location, school, city, company, bird, drug, etc.

4.2 Determining Answer Type using LSP

It is important for a QA system to predict what type of answer the question requires (i.e. answer type): person name, location, organization, or any others since it can fairly reduce the number of answer candidates. Referring to the previous TREC QA questions, we defined 62 answer types and developed a method that classifies questions into the answer types using the LSP's.

Usually an interrogative in a question is an important factor but it is not enough to determine the answer type of a question because it also can has sense ambiguities like other words. For example, 'どこ' indicates a '*location*' in a question, "東京ディズニーランドはどこにありますか。(*Where is Tokyo Disneyland?*)" while it indicates a '*company*' in another question, "日本サブウェイはどこの会社の子会社ですか。(*Nihon Subway is which company's subsidiary?*)". This means that LSP's for classifying questions must include its surrounding contexts as well as an interrogative itself. 'どこにあり' and 'どこの会社' are key phrases that can tell the answer type of two questions and expressed in LSP grammar for 'どこ' question as follows:

(どこ)(に)(ある) → 1|3|location
(どこ)(の)(%company) → 1|3|company

'*%company*' is a semantic class to represent synonyms of '会社'. The first number in the right-hand side of arrow (→), divided by vertical bar, indicates the location of an interrogative ('どこ' in this case) in the LSP of the left-hand side and the second number indicates number of components of the LSP. The third is an answer type. In other words, the above grammars represent that the LSP consists of three components and the first one is an interrogative and if some part of a question matches with the LSP, then the answer type of the question is '*location*' (or '*company*'). If more than one answer type is possible to each LSP then all possible answer types can be enumerated with a separating vertical bar. Table 2 shows some more examples of LSP grammars for determining answer types with sample questions.

Determined answer type can be extended to several subsumed answer types. For example, '*location*' is extended to "*location, state, city, country, etc*" before detecting answer candidates. If a question is not assigned to any predefined answer type, we assign '*language_unit*' to the question.

4.3 Detecting Answer Candidates using LSP

Unlike the previous TREC QA tasks, we have to return only exact answers in the QAC task. This makes answer detection module more important since an answer string involving any dummy words will be judged as incorrect.

LSP is used for extracting answer candidates from text as well as classifying questions. Once an answer type of a question is determined, what we

Interrogatives	Sample question	Grammar
何/なん/なに (*what/which*)	"夏目漱石の名作は何ですか。" (*What is the masterpiece of Souseki Natsume?*) "千葉県の県庁所在地は何市ですか。" (*Which city is the provincial office of Chiba located?*)	(%work)(は)(何)(です) → 3|4|movie|book|music (は)(何)(%city)(です) → 2|4|city
誰/だれ/どなた (*who*)	"大学審議会の会長は誰ですか。" (*Who is the president of university council?*)	(@position)(は)(誰)(です) → 3|4|person
どこ/何処/何所, どちら/どっち (*where/which direction*)	"タージ・マハールはどこにありますか。" (*Where is Taj Mahal?*)	(どこ)(に)(ある) → 2|3|location
いつ (*when*)	"米ソの冷戦が終わったのはいつですか。" (*When was the Cold War between USA and Russia terminated?*)	(は)(いつ) → 2|2|date

Table 2. Examples of LSP grammars for classifying questions into answer types

Instances	Grammars
30歳 (*30 year-old*) 夏目房之介さん(47) (*Mr. Husakorekai Natsume (47)*)	(@number)(@unit_age) → age|1|2|2|1.0 (@person)(NUP)(()(@number)()) → age|4|4|5|1.0
夏目漱石の「こころ」(*Natsume's 'Kokoro'*) …主演の「L. A. コンフィデンシャル」 (*'LA Confidential' starring …*)	(@person)(の)(@bracket) → book|3|3|3|0.85 (主演)(の)(@bracket) → movie|3|3|3|0.9
バイスフロクさん (*Mr. Baisuhuroku*) 村上雅則投手 (*A pitcher Masanori Murakami*)	(@np)(NUP) → person|1|1|2|0.85 (NPPS)(NPPG)(@position) → person|1|2|3|1.0
スカイマークエアラインズ(本社・東京) (*Sky Mark Airlines (head office, Tokyo)*)	(K)(()(本社)(・) → company|1|1|4|0.9

Table 3. Sample grammars for extracting answer candidates

have to do is to find out entities belonging to the answer type within passages selected in the previous step. Named Entity (NE) tagger was used for this purpose in many QA systems and is substituted by LSP grammars in our system.

From instances belonging to each answer type, gathered from texts, we can construct LSP grammars for extracting answer candidates. For example, in a text snippet, "村上雅則さん(53)", we can extract '村上雅則' and '53' as '*person name*' and '*age*', respectively, using following LSP grammars:

(NPPS)(NPPG)(さん) → person|1|2|3|1.0
(NPPS)(NPPG)(さん)(()(@number)()) → age|5|5|6|1.0

The first grammar means that the LSP is composed of three components and if it matches with a text snippet, then corresponding parts to between the first component and the second including boundaries is extracted as '*person name*' with weight 1.0. Weight value indicates the precision of the pattern and can be obtained by counting correct instances among ones extracted by applying the pattern to test collection. Table 3 shows some more examples of LSP grammars for detecting answer candidates.

If a question is not assigned to any predefined answer type and so assigned to '*language_unit*', any noun phrase can be considered as answer candidates, but we prefer proper noun or bracketed phrase to others.

Answer detection must be applied to a question itself since it also can contain entities belonging to its answer type. An answer candidate to a question may be filtered out if it occurs in the question itself.

4.4 Answer Filtering and Scoring

Before scoring each answer candidates, they are filtered out by the following assumptions:

- A word itself occurring in the question cannot be an answer.
- An answer candidate cannot be a correct answer if it also occurred in the question
- An answer candidate cannot be a correct answer if it came from the phrase presenting the origin of the news article. This heuristic is specific to the test collection.

Some heuristics specific to each answer type are also applied for filtering. For example, the number representing a year usually consists of 2 to 4 digits. One character person name is also filtered out.

If date constraint was specified in a question, e.g., "99年7月の完全失業率は何パーセントでしたか。(*What percent was the whole-unemployment rate in July, 99?*)", then answer passage should satisfy it too. That is, the date should occur in the answer passage or be same with the date of document from which the answer passage came.

Some questions contain a word that can become a clue for finding answers. For example, in a question, "チュニジアの人口は何人ですか。(*How many people are there in Chunizia?*)", a word, '人', is a clue for finding an answer, "909万人 (*909 ten thousand people*)". This clue word can be useful for filtering answer candidates especially for '*count*' type answers.

The score of each extracted answer candidate is based on the number of unique terms matched in the passage and distances from each matched term. This assumes that a correct answer to a question will occur probably near terms matched with the question. This assumption is not necessarily true but makes a good hit for many questions of the previous TREC QA tasks.

The score of an answer candidate is calculated using the following four features and linearly combined with the passage score (expression (2))

- *LSPwgt* : the weight of the LSP's used to extract the answer candidate. This reflects the confidence of LSP's applied to extract the answer candidate.
- *qtuc* : the count of unique query terms.
- *ptuc* : the count of unique terms matched with query terms in a selected passage. The ratio of *ptuc* to *qtuc* reflects the weight of a passage from which the answer candidate was extracted.
- *avgdist* : the average distance between the answer candidate and each matched term.

$$AScore = \beta \times PScore + (1-\beta) \times LSPwgt \times \frac{ptuc}{qtuc} \times \frac{ptuc}{(ptuc + avgdist)} \quad (2)$$

, where β is a constant, *PScore* is the score of a passage, from which the answer candidate was extracted.

When ranking answer candidates, we check any duplicates and remove them out. For person name, we also check partial duplicates between family name and full name.

5. Experiments in NTCIR3

We participated in the QAC Task1 of NTCIR 3. The test collection was composed of 230,000 documents (about 280MB) from 98-99 Mainichi Newspaper articles and relatively small compared to the TREC test collection (about 3GB).

To test each participating system, 200 factoid questions were provided by National Institute of Informatics (NII) and each participant returned top five exact answer candidates to each question. This exact answer constraint is one of the main differences from the TREC-8,9,10. Among the questions, five questions were excluded from evaluation since they had no answer in the test collection.

Returned answers were judged by NII assessors and evaluated by mean reciprocal rank. In this section we report our system's performance evaluated from NII and performances of each module in our system.

5.1 Experimental Environment

We indexed the test collection using our document retrieval system for Japanese, POSNIR/J, which maintains each document with POS tag provided by POSTAG/J, our POS tagger for Japanese, and term position information (TPI) of each indexed term.

We then manually constructed LSP grammars for answer type identification and answer candidate detection from the extra web data and the texts of the test collection.

Our system used top 1000 documents retrieved by POSNIR/J to select answer passages and top 500 passages to detect answer candidates and returned exact answer candidates in 8.74 seconds per question averagely on a PC server with P-III 650MHz dual CPUs.

5.2 Effect of Document Retrieval

Based on the answer file provided by NII, we evaluated result of our document retrieval system. We constructed relevant document list for each question from the answer file and ran the evaluation tool, *trec_eval*, on the retrieved document list. The evaluation result is shown in Table 4.

Unlike general document retrieval, just one relevant document is enough to answer to a question. So we inspected rank of the first relevant document to each question. All questions among 195 questions except only one (q#24) had at least one relevant document within top 500 retrieved documents and 93.3% had at least one within top 100.

Top *N* docs	Precision
5 docs	0.1672
10 docs	0.1267
15 docs	0.1002
20 docs	0.0874
30 docs	0.0697
100 docs	0.0328
200 docs	0.0206
500 docs	0.0096
1000 docs	0.0053
Retrieved	192667
Relevant	1123
Rel_ret	1037

Table 4. Performance of document retrieval

Top *N* passages	Rate of questions
5 psgs	0.7897
10 psgs	0.8513
15 psgs	0.8718
20 psgs	0.8923
30 psgs	0.8974
50 psgs	0.9123
100 psgs	0.9179
200 psgs	0.9385
500 psgs	0.9487

Table 5. Performance of passage selection

5.3 Effect of Passage Selection

We also evaluated the performance of passage selection module. Based on (answer instance, document id) pairs of the answer file, we judged a selected passage as correct if it contains an answer instance and its document id is same with document id of the answer instance. Table 5 shows the rate of questions whose answer was included within top N selected passages. Only 10 questions had no answers within top 500 selected passages and 89.2% of the questions had an answer within top 20 selected passages. For two questions (q#78 and q#82) that had no answer passages within top 20, our system returned correct answers at top rank. As evaluated by mean reciprocal rank (MRR) of the first relevant passage, our system got 0.6560.

5.4 Performance of Question Answering

Finally, returned answers of our system were evaluated. Table 6 shows the performance of our system by the scoring tool (version 1.40) provided by NII. Five among 200 questions were excluded from the computation of mean reciprocal rank (MRR) as they have no answers in the test collection.

Some questions have more than one answer from more than one document. There are 288 correct

answers to 200 questions in the test collection. Our system returned 994 answer candidates to 200 questions and, among them, 220 responses were judged as correct and 183 were distinct answers. Recall and precision were calculated from the numbers of answers, outputs and correct outputs. F-measure was calculated by F-measure = (2 * Recall * Precision) / (Recall + Precision).

In MRR, the standard evaluation criteria in QAC Task1, our system recorded 0.608, which is best performance among 15 participating systems (average 0.303). Our system returned correct answers within top five for 149 questions and at the first rank for 98 questions. Table 7 shows the number of questions according to the rank of the first correct answer and Table 8 shows the performance of our system according to each answer type. Our system was successful to return the correct answers to about half of the questions at top rank, and the deviation of performance according to answer type is not very large.

Compared to the performance (MRR=0.320) we made at TREC-10, this performance is very high even though the exact answer constraint was given. This may be because the test collection is very small and the 'no answer' questions were excluded in NTCIR 3. Another reason can be that we used our own document retrieval system this time. In TREC-10, we used document list provided by NIST since we couldn't finish indexing of the test collection within the given amount of time. However, after the evaluation, we finished indexing and found that the results of our own document retrieval system are much better than the document lists given by NIST.

Question	Answer	Output	Correct
200	288	994	183
Recall	**Precision**	**F-measure**	**MRR**
0.635	0.184	0.285	0.608

Table 6. Performance of our system

Rank	# of questions
1	98
2	27
3	14
4	6
5	4

Table 7. Number of questions according to the rank of the first correct answer

Answer type	# of questions	MRR
Artifact	21	0.579
Date	14	0.649
Living thing	6	0.389
Location	31	0.610
Number	31	0.661
Organization	19	0.543
Person	42	0.614
Substance	6	0.750
Title	10	0.683
Others	15	0.536
Total	195	0.608

Table 8. Performance according to each answer type[3]

6. Discussions

In this section we analyze our system's results with some examples, especially in the case of failures.

Answer types we used were not enough to cover all the questions. Q#36 and q#132 require a name of prize and legislative bill, respectively, but they are not defined in our system. In questions like q#7 and q#72, *artifact* is too rough to detect correct answers. We must define more general answer types using general or domain-specific thesaurus so as to cover various questions.

For some questions like q#90, q#169, q#173, q#186, etc., our system failed to determine answer types and thus failed to return correct answers because manually constructed LSP's and semantic class dictionaries were not complete yet.

Failure to determining answer type does not always mean failure to detecting correct answers. For q#21, q#152, q#187, q#195, etc., our system could be successful to return correct answers by selecting bracketed entities.

Some questions can be assigned to multiple answer types. In this case, answer candidates belonging to each answer type are detected and ranked by the scoring measure. For example, q#29 was assigned '*location*', '*company*', etc. and q#136 was assigned '*year*' and '*count*', this increased the ambiguity of answer candidates though correct answers to some questions could be ranked within top five by the scoring measure.

We did not divide '*count*' type in detail except age, volume, weight, area, speed, size, temperature, duration, rate, length, power and money. That is, we did not distinguish number of people from number of animals. Instead, we used unit word as a clue for detecting correct answer candidates. Unit word, however, is not always available with all '*count*' type questions (e.g., q#49, q#52, q#194, etc). This also increased the ambiguity of answer candidates and made it difficult to rank correct answers high.

Our LSP grammar for detecting answer candidates was not enough and complete, and thus failed to detect correct answers or extracted incorrect-type candidates to some questions.

Success in determining answer type and detecting correct answer candidates does not always mean correct answers. Many of other failures were caused that our system ranked answer candidates only depending on the probabilistic measures without any

[3] Living thing includes 'bird' and 'plant', location includes 'city', 'country', 'mountain', 'planet', 'river' and 'state', number includes 'age', 'area', 'count', 'duration', 'length', 'money', 'rate', 'speed' and 'temperature', organization includes 'company' and 'team', and title includes 'book', 'magazine', 'movie' and 'music'.

semantic answer justification.

7. Conclusions

We developed our Question Answering system for Japanese, SiteQ/J, to participate in QAC Task1 of NTCIR3.

Our system narrows a search space using dynamic answer passage selection, and determines answer type of a question and extracts answer candidates using Lexico-Semantic Pattern matching without deep linguistic analysis of the texts. Detected answer candidates are ranked by passage scores and the measure of distance between answer candidates and matched terms. As a result of LSP engineering, our system showed the best performance (MRR=0.608) among 15 participating systems.

In the future, we will try to reduce manual efforts to construct LSP's and semantic category dictionaries by automatically learning them from extra web data. We will develop a method and construct knowledge to justify answers for higher precision. Automatic contextual taxonomy [20] learning can be one of the alternatives to go.

References

[1] E. Brick, J. Burger, D. House, M. Light, and L. Mani. Question answering from large document collections. In *The 1999 AAAI Fall Symposium on Question Answering Systems*, pages 26-31. North Falmouth, Massachusetts, 1999. AAAI.

[2] J. P. Callan. Passage-level evidence in document retrieval. In *The 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 302-309, Dublin, Ireland, 1994. ACM.

[3] C. L. A. Clarke, G.V. Cormack, D. I. E. Kisman, and T. R. Lynam. Question Answering by passage selection (MultiText Experiments for TREC-9). In *The 9th Text Retrieval Conference (TREC-9)*, pages 673-683, Maryland, 2000. NIST.

[4] C. L. A. Clarke, G. V. Cormack, and T. R. Lynam. Exploiting redundancy in question answering. In *The 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 358-365, Louisiana, 2001. ACM.

[5] S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, and P. Morarescu. Falcon: Boosting knowledge for answer engines. In *The 9th Text Retrieval Conference (TREC-9)*, pages 479-488, Maryland, 2000. NIST.

[6] M. A. Hearst and C. Plaunt. Subtopic structuring for full-length document access. In *The 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 59-68, Pittsburg, 1993. ACM.

[7] E. Hovy, L. Gerber, U. Hermjakob, M. Junk, and C.-Y. Lin. Question answering in webclopedia. In *The 9th Text Retrieval Conference (TREC-9)*, pages 655-664, Maryland, 2000. NIST.

[8] A. Ittycheriah, M. Franz, W.-J. Zhu, and A. Ratnaparkhi, IBM's Statistical Question Answering System, In *The 9th Text Retrieval Conference (TREC-9)*, pages 229-234, Maryland, 2000. NIST.

[9] M. Kaszkiel and J. Zobel. Passage retrieval revisited. In *The 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 178-185, Philadelphia, 1997. ACM.

[10] G. G. Lee, J. Seo, S. Lee, H. Jung, B.-H. Cho, C. Lee, B.-K. Kwak, J. Cha. D. Kim, J. An, H. Kim, and K. Kim. SiteQ: Engineering high performance QA system using lexico-semantic pattern matching and shallow NLP. In *The 10th Text Retrieval Conference (TREC-10)*, pages 437-446, Maryland, 2001. NIST.

[11] D. Moldovan and S. M. Harabagiu. Lasso: A tool for surfing the answer net. In *The 8th Text Retrieval Conference (TREC-8)*, pages 175-184, Maryland, 1999. NIST.

[12] J. Prager. Question-answering by predictive annotation. In *The 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 184-191, Athens, 2000. ACM.

[13] G. Salton, J. Allan, and C. Buckley. Approaches to passage retrieval in full text information systems. In *The 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 49-58, Pittsburg, 1993. ACM.

[14] M. M. Soubbotin. Patterns of potential answer Expressions as clues to the right answers. In *The 10th Text Retrieval Conference (TREC-10)*, pages 293-302, Maryland, 2001. NIST.

[15] E. M. Voorhees and D. Tice. The TREC-8 question answering track evaluation. In *The 8th Text Retrieval Conference (TREC-8)*, pages 83-105, Maryland, 1999. NIST.

[16] E. M. Voorhees and D. Tice. Building a question answering test collection. In *The 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 200-207, Athens, 2000. ACM.

[17] E. M. Voorhees. Overview of the TREC-9 Question Answering Track. In *The 9th Text Retrieval Conference (TREC-9)*, pages 71-79, Maryland, 2000. NIST.

[18] E. M. Voorhees. Overview of the TREC 2001 Question Answering Track. In *The 10th Text Retrieval Conference (TREC-10)*, pages 42-51, Maryland, 2001. NIST.

[19] R. Wilkinson. Effective retrieval of structured documents. In *The 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 311-317, Dublin, Ireland, 1994. ACM.

[20] W. A. Woods. Conceptual indexing: A better way to organize knowledge. In Technical Report SMLI TR97-61, Sun Microsystems Laboratories, Mountain View, CA., 1997.

[21] J. Zobel, A. Moffat, R. Wilkinson, and R. Sacks-Davis. Efficient retrieval of partial documents. Information Processing Management, 31(3):361-377, 1995.