SiteQ: Engineering High Performance QA system Using Lexico-Semantic Pattern Matching and Shallow NLP

Gary Geunbae Lee, Jungyun Seo*, Seungwoo Lee, Hanmin Jung, Bong-Hyun Cho, Changki Lee, Byung-Kwan Kwak, Jeongwon Cha, Dongseok Kim, JooHui An, Harksoo Kim**, Kyungsun Kim**

Dept. of Computer Science & Engineering, POSTECH
Dept. of Computer Science, Sogang University*
DiQuest.com, Inc.**

Abstracts

In TREC-10, we participated in the web track (only ad-hoc task) and the QA track (only main task).

In the QA track, our QA system (SiteQ) has general architecture with three processing steps: question processing, passage selection and answer processing. The key technique is LSP's (Lexico-Semantic Patterns) that are composed of linguistic entries and semantic types. LSP grammars constructed from various resources are used for answer type determination and answer matching. We also adapt AAD (Abbreviation-Appositive-Definition) processing for the queries that answer type cannot be determined or expected, encyclopedia search for increasing the matching coverage between query terms and passages, and pivot detection for the distance calculation with answer candidates.

We used two-level answer types consisted of 18 upper-level types and 47 lower-level types. Semantic category dictionary, WordNet, POS combined with lexicography and a stemmer were all applied to construct the LSP knowledge base. CSMT (Category Sense-code Mapping Table) tried to find answer types using the matching between semantic categories and sense-codes from WordNet. Evaluation shows that MRR for 492 questions is 0.320 (strict), which is considerably higher than the average MRR of other 67 runs.

In the Web track, we focused on the effectiveness of both noun phrase extraction and our new PRF (Pseudo Relevance Feedback). We confirmed that our query expansion using PRF with TSV function adapting TF factor contributed to better performance, but noun phrases did not contribute much. It needs more observations for us to make elaborate rules of tag patterns for the construction of better noun phrases.

1. Introduction

The goal of the QA track is to foster research on systems that retrieve answers rather than documents in response to a question [11][12]. The focus is on systems that can function in unrestricted open domains [11].

The web track features ad hoc search tasks on a document collection that is a snapshot of the World Wide Web. The main focus of this track is to form a Web test collection using pooled relevance judgments. We will describe our systems and experiences for both QA and Web tracks in this paper.

2. QA track: Systems and Experiences

In TREC-10, the QA track consisted of three separate tasks: the main task, the list task and the context task. We participated in only the main task.

The main task is similar to the task in previous QA tracks (TREC-8, TREC-9). NIST provided 500 questions that seek short, fact-based answers. Some questions may not have a known answer in the document collection. In that case, the response string "NIL" is judged correct. This differs from the previous QA tracks and makes the task somewhat more difficult. The answer-string should contain no more than 50 bytes; 250-byte runs were abandoned this year. Participants must return at least one and no more than five responses per question ranked by preferences.

The document collection consists of the following six data sets: AP newswire, Wall Street Journal, San Jose Mercury News, Financial Times, Los Angeles Times, and Foreign Broadcast Information Service. The documents are SGML tagged, and each document in this collection has a unique identifier in the field.

Distinguished from an information retrieval, a QA system must retrieve answers rather than documents as responses to a question. As an ordinary course of step, we focused on what can be a possible answer, how our system can determine the answer type of a question, and how our system can detect instances of each answer type in a document. We classified possible answers and designed a method for determining the answer type of each question and detecting instances of it in a document. We have not constructed the index of document collection this time and instead used the ranked document list provided by NIST for each question.

Our QA system, SiteQ, consists of three important steps; question processing, passage selection and answer processing, which will be explained in detail.

2.1 Question Processing

In general, a question answering system analyzes an input question at first step. It is important to understand what a user wants to find; whether it is person's name, location, organization, or any other types. To do so, we first classified the types of possible answers [1][2][3][6] and used Lexico-Semantic Patterns (LSP) to determine the answer type of a question.

2.1.1 Answer Type

We classified the type of answers to fact-seeking questions [12]. Referring to the types used in FALCON [3], we analyzed the questions used in the previous QA tracks and their answers judged correct and constructed 2-level hierarchy of answer types. Hierarchical structure of answer types is useful since only YEAR is available for 'what year' question, but YEAR, MONTH, DAY, or TIME is available for 'when' question. Our answer type has 18 types at top level as shown in the box.

QUANTITY DATE TIME PROPERTY
LANGUAGE_UNIT LANGUAGE
SYMBOLIC_REP ACTION ACTIVITY
LIFE_FORM NATURAL_OBJECT
LOCATION SUBSTANCE ARTIFACT
GROUP PHENOMENON STATUS
BODY_PART

2.1.2 Lexico-Semantic Patterns

Usually an interrogative in a question is an important factor but it is not enough to determine the answer type. LASSO first determined the question class and the question focus, and then determined the answer type by using them [6]. The question class is defined as an interrogative and the question focus is defined as the main information required by the interrogation.

We used Lexico-Semantic Patterns (LSP) to determine the type of answer expected. Usually in addition to an interrogative in a question, its surrounding words or their senses are expressed in LSP, which substitutes the question class and focus word.

LSP grammar is composed of condition part and conclusion part. The conclusion part is the type of answer expected if the LSP in condition part is matched. LSP is composed of lexical entries, POS tag, semantic category and their sequence, and is expressed in regular expression. For example, a grammar *"(%who)(%be)(@person) → PERSON"* can be constructed from a question *"Who was President Cleveland's wife?"*. '*%who*' and '*%be*' is lexical entries and '*@person*' is a semantic category for representing the position of a person. We have manually constructed LSP grammar from the questions used in the previous QA tracks and the questions gathered from the Web by ourselves. Among them 361 entry LSP grammar was used for this year's QA track.

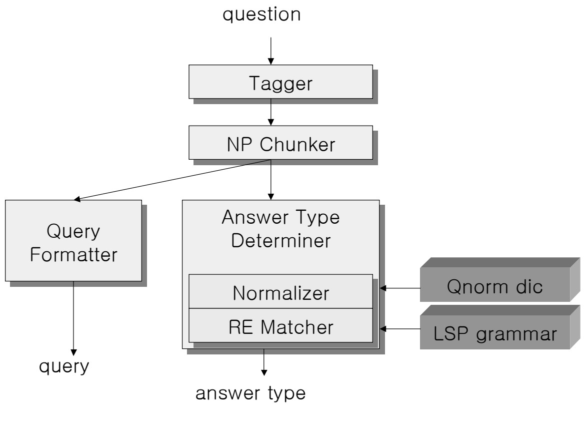


Figure 1 Question processing

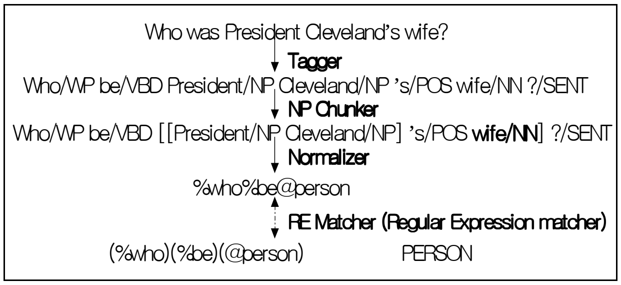
2.1.3 Determining The Answer Type

Figure 1 explains the procedures to determine the expected answer type of an input question. At first, an input question is POS-tagged using POSTAG/E English tagger and, at the second step, noun phrases are detected by NP chunker. Scanning the tagged question from right to left, this module detects the boundary of noun phrase and its head noun. To do this, we collected the POS patterns for noun phrases from the questions. A noun phrase almost always ends with a noun, usually starts with a pre-determiner, a determiner, an adjective, a possessive pronoun, or a noun. The rightmost noun in a noun phrase is selected as a head noun. Two noun phrases can be combined into a larger noun phrase by connecting them using a preposition '*of*' or a possessive ending. In case of a preposition '*of*', the head of its left-side noun phrase is selected as a head of the combined noun phrase, but in case of a possessive ending the head of its right-side noun phrase is selected.

It is important that detecting a head in a noun phrase since the sense of the head noun plays an important role in determining the expected answer type but its modifiers are useful for justifying final answers. In the above question, "*President Cleveland's wife*" is detected as a noun phrase, and '*wife*' is its head and clarifies the answer type of the question is PERSON. In contrast to this question, the expected answer type of a question "*Who is Cleveland?*" will be POSITION, which means the position of Cleveland (i.e., president) will be an answer.

At the third step, based on normalization dictionary (Qnorm dic) and WordNet, each word in a question is converted into LSP code to be matched with the condition part of LSP grammar by regular expression. "*President Cleveland's wife*" is converted into '*@person*' since it is a noun phrase and its head is '*wife*', of which semantic category is '*@person*'.

The following box shows how the answer type of a question "*Who was President Cleveland's wife?*" is determined as PERSON.


Who was President Cleveland's wife?
Tagger
Who/WP be/VBD President/NP Cleveland/NP 's/POS wife/NN ?/SENT
NP Chunker
Who/WP be/VBD [[President/NP Cleveland/NP] 's/POS wife/NN] ?/SENT
Normalizer
%who%be@person
RE Matcher (Regular Expression matcher)
(%who)(%be)(@person)
PERSON

2.2 Passage Selection

We have not constructed an index database from the document collection since we had no enough time and computing resources this year. Therefore we couldn't help using only the document list provided by NIST and selecting relevant passages from them by scanning the whole documents and matching the keywords. The documents were ranked by document similarity because they were retrieved by the PRISE [7], a document retrieval system rather than a passage retrieval system. Generally, however, a document does not fit for detecting candidate answers within itself since it is too large and contains too much extra information. By analyzing the previous questions and their answers, we can assume that answers to a question usually occur comparatively near to the matched keywords in a document. This means that the answer can occur in any ranked documents and we had better select passages from each document and rank them by passage similarity. Then we can use top passages to find candidate answers. To do so, we first must define passage and keywords to be used in selecting relevant passages.

2.2.1 Keywords

We define keywords to be used in selecting passages from the retrieved documents. We first remove useless words in a question and then use the remained words as three types of keywords considering lexical normalization and semantic similarity. Finally we assign weights to each keyword.

- Removing stop words

The useless words in a question are removed first by POS tag and stop word list, which has 568 entries. Then the following five heuristics are applied to the remaining words.

a. When a word like 'kind', 'sort', 'one', 'most', etc. occurs in the left side of a preposition 'of', it is removed; eg) What *kind* of dog ...? Name *one* of the Seven Wonders ...?
b. When a word like 'name', 'nickname', etc. occurs in the right side of a possessive ending, it is removed; eg) What was the man's *name* who was killed ...? What is Shakespeare's *nickname*?
c. When a question is expressed in imperative sentence, the imperative verb is removed; eg) *Tell* me what city ...?
d. When a verb needs a to-infinitive, the verb is removed; eg) Where do lobsters

like to live?

e. When an adjective or an adverb follows an interrogative 'how', the adjective or adverb is removed; eg) How *wide* is the Atlantic Ocean?

- The type of keyword

After removing all stop words, the remaining words are considered as question keywords. We define following three types of keyword to solve the mismatching problem of keywords caused by lexical variants and synonyms.

a. Lemma form

The lemma form of a word is used as a keyword except the superlative adjective or adverb, in which case the word itself is used as a keyword; eg) invented → *invent*, inventers → *inventer*, smallest → *smallest*

b. Stemmed form

Though the lemma form solves somewhat of the mismatching problem, it is not enough to solve the mismatch between 'inventer' and 'invented'. This can be resolved by using a stemmer like the Porter's stemmer [8].

c. WordNet sense (in case of noun or noun phrase)

To match a word 'ship' in a question with a word 'steamship' in a document, we must compute semantic similarity between a question keyword and a document word. Using the WordNet [5], the synonym or hyponym of a question keyword occurring in documents is matched with the question keyword.

- The weight of the keyword

The lemma form is weighted by its part of speech. A proper noun, a common noun starting with a capital letter, and a superlative has higher weight than a verb, an adjective and an adverb. The stemmed form has some of the weights its lemma form has. The keyword (noun or noun phrase) matched by WordNet sense has the lowest weight relative to the number of its component words.

2.2.2 Passages

A passage is composed of more than one sentence segmented by punctuation. We make adjacent two sentences into a passage if they have a lexical chain, which indicates that a sentence has a noun and the other sentence has its anaphora. We however limited a passage to maximum three sentences since the more sentences have the more extra information, which may increase incorrect candidate answers.

Each sentence from a document gets scored by matching its terms with query terms (Score$_1$) and by considering the distance and number of the matched terms (Socre$_2$). Score$_1$ consists of sum of the weights of matched terms. Each query term is tried to be matched with document terms in the order of lemma form, WordNet sense and stemmed form, and gets assigned the weight of the first matched term type. Passages are ranked by sum of their sentence scores.

$$Score_{sent} = Score_1 + Score_2 \qquad (1)$$

$$Score_1 = \sum_i wgt(qw_i) \qquad (2)$$

$$\text{if } qw_i \text{ appears in a stentence}$$

$$Score_2 = \frac{\sum_{j=1}^{k-1} \frac{wgt(dw_j) + wgt(dw_{j+1})}{\alpha \times dist(j, j+1)^2}}{k-1} \times matched\_cnt \qquad (3)$$

$wgt(qw_i)$: weight of query word i

$wgt(dw_j)$: weight of query word i,
with which document w ord j was matched

$dist(j, j+1)$: distance between
document w ord j and $j+1$

$matched\_cnt$: number of query word s
matched in a sentence

α : constant

Our system selected 1000 passages from 1000 retrieved documents per question.

2.3 Answer Processing

Answer processing selects answer candidates matching the answer type from each passage and ranks them. It uses stemmer[8], thesaurus (WordNet) [5], encyclopedia for its performance elevation. Answer processing is composed of four steps: Answer Matching, Pivot Detection, AAD Processing and Answer Ranking.

2.3.1 System Architecture

Figure 2 shows components of answer processing system. Answer matching (detection) finds answer candidates in POS-tagged passages selected by passage selection using the answer type determined by question processing. A query term, which shows up in various forms in the passage, is called “pivot”. Answer ranking uses these pivots in scoring answer candidates. When the answer type of a question is “LANGUAGE_UNIT”, AAD processing finds context-based answer candidates that are in abbreviated, appositive and definitive relation with the pivots. Answer ranking calculates the score of each answer candidate with various parameters, filters them according to the range and the type of answer, and finally sorts them.

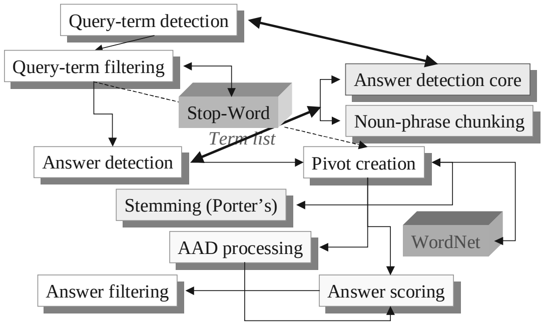


Figure 2 Answer processing

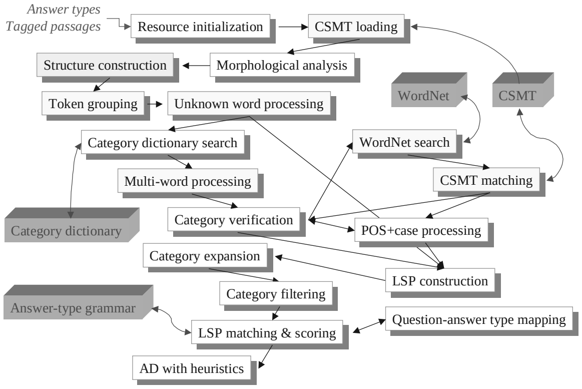


Figure 3 Answer matching

2.3.2 Answer Matching (detection)

Figure 3 shows the procedures of answer matching. Answer matching assigns semantic categories to each answer candidate by matching between LSP grammar and the normalized answer form from the following procedure. The procedure first searches semantic category dictionary. In case of its failure, it tries thesaurus matching between the sense-code from WordNet and the semantic categories in the CSMT (Category to Sense code Mapping Table), and then uses POS combined with lexicography.

- Searching semantic category dictionary

Semantic category dictionary has about 80,000 entries including single word and compound one. Each entry is assigned a semantic category among 65 ones which are components of LSP abstraction.

- Trying thesaurus matching

Sense code retrieved from WordNet [5] is mapped to each category among 65 semantic categories if it has a similarity greater than a threshold value.

- POS combined with lexicography

In case of failure of searching semantic category dictionary, POS combined with lexicography is used to build normalized form. If “Newton” has “np” (proper noun) POS tag, “Np” is used for normalization. It is because capitalization is important for detecting candidate answers, especially named entities.

When a normalized form matched with a LSP of the answer type, its terms are chosen as an answer candidate. The followings show some examples of LSP and its actual instances.

cd@unit_lengthcd@unit_length   *length|1|4|4*
10 feet 5 inches
cd@unit_length%per@unit_time   *speed|1|4|4*
3 km per hour

2.3.3 Pivot Detection

Pivots corresponding with query terms emerge in the passage in various way: full matching terms, partial matching ones for multi-words, stem matching ones for inflections and semantic synonyms using WordNet. When answer ranking scores answer candidates, pivots are weighted according to these normalized representations of query terms in a passage. When an answer candidate itself is a pivot, it is excluded from answer candidate set.

2.3.4 AAD Processing

In the case that no answer type can be

determined in question processing due to short of information ("LANGUAGE_UNIT" answer type), AAD processing finds context-based answer candidates that are in abbreviated, appositive and definitive relation with the pivots. It uses lexicographic patterns for abbreviation, and noun phrase chunking and clue words such as "so-called" and "stand for" for apposition and definition. The followings are examples of questions, of which answer type is LANGUAGE_UNIT.

Why does the moon turn orange?
What is epilepsy?
What imaginary line is halfway between the North and South Poles?
What is done with worn or outdated flags?

For more improvement of performance, AAD processing uses encyclopedia information extracted from WordNet glossary [5]. We gathered descriptions of about 110,000 words from WordNet glossary and removed stop words from the descriptions. Answer ranking reweighs each answer candidate through its semantic similarity with remaining terms in the descriptions.

2.3.5 Answer Ranking

Score of each answer candidate is mainly calculated by distance between pivots within some window in each selected passage. In addition to basic distance measure, the type and ratio matching each pivot with query terms, mean distance between pivots, and semantic type of answer candidate (especially in case of AAD processing) are all used for scoring each answer candidate:

$$Score_i = R_{pivot} \cdot (1 - \frac{dist_{avg.pivot}}{dist_{\max.pivot}}) \cdot \frac{S_i}{AADfactor} \cdot \frac{1}{N_p} \cdot \sum_{j=1}^{N_P} r_j \cdot (1 - \frac{dist_j}{dist_{\max}}) \qquad (4)$$

R_{pivot}: ratio of matched pivots
$dist_{avg.pivot}$: average distance between pivots
$dist_{max.pivot}$: maximum of distance between pivots
S_i: intermediate score of *i*th Answer Candidate
AADfactor:
if question type is language-unit,
if NE type is AAD, 1
otherwise 4
otherwise 1
$Score_i$: final score of *i*th Answer Candidate
N_p: number of Pivots
r_j: weight factor of match type of *j*th Pivot
$dist_j$: distance between *j*th Pivot and *i*th Answer Candidate
$dist_{max}$: max value of $dist_j$

This formula (Eq. 4) reflects some of the following assumptions: (1) Reliable answer candidates would appear near query terms, so called pivots, in a passage. (2) Reliable answer candidates would show up around pivots which matched with query terms more exactly. (3) In the case of "LANGUAGE_UNIT" answer type, answer candidates extracted from AAD processing are more reliable than the others. (4) The smaller mean distance between pivots is, the more reliable an answer candidate around them would be. (5) If most of query terms appear in a passage, an answer candidate around their pivots is more reliable. (6) Finally, reliable answer candidates show up in some limited distance between pivots. After scoring all answer candidates, answer ranking filters less reliable answer candidates according to the range and type of the answer, sorts remaining answer candidates by their scores and presents *N* most reliable answer candidates.

2.4 Experiments in TREC-10

We participated in the main task of QA track. 500 questions were given to each participant to evaluate their QA systems. After all evaluation, it was known that 49 questions among them have no known correct answers in the document collection. Eight questions were excluded from the evaluation due to various problems with those questions.

Table 2 shows that, unlike the questions used in the previous QA tracks, questions like "what is X?" were remarkably increased. So, the task became more difficult since the answer types of such questions are often not specified definitely.

For each question, SiteQ used the top 1000 documents provided by NIST (PRISE search engine [7]), selected top 1000 passages from those documents, detected top five candidate answers from those passages and picked out 50-byte string including the candidate answer as an answer string. When the score of a candidate answer was lower than a threshold value or less than five candidates were detected, we added "NIL" string in the appropriate rank, which means that there might be no answer.

We submitted only one run (posqa10a) and it was evaluated by mean reciprocal rank (MRR) like the previous QA tracks [13]. The unsupported answers were judged incorrect in strict judgment but correct in lenient judgment. Table 1 shows the number of questions judged correct in each judgment and the mean reciprocal rank of 492 questions. Comparing with the average MRR of the 67 other runs submitted this year, our system located correct answers at rank 1 for relatively many questions. The difference between the strict and the lenient MRR arises because a word of the same answer type was added to 50-byte string when we picked out the answer string including a candidate answer.

Rank	# of Qs (strict)	# of Qs (lenient)	Avg. of 67 runs
1	121	124	88.58
2	45	49	28.24
3	24	29	20.46
4	15	16	12.57
5	11	14	12.46
No	276	260	329.7
MRR	0.320	0.335	0.234

Table 1 The number of questions judged correct and MRR

Q-type	freq	MRR	MRR
		(strict)	(lenient)
how + adj/adv	31	0.316	0.332
how do	***2***	***0.250***	***0.250***
what do	***24***	***0.050***	***0.050***
what is	**242**	**0.308**	**0.320**
what/which noun	*88*	*0.289*	*0.331*
when	26	0.362	0.362
where	27	0.515	0.515
who	46	0.464	0.471
why	***4***	***0.125***	***0.125***
name a	2	0.500	0.500
Total	492		

Table 2 The frequency and MRR in each type of question

Table 2 shows the MRR for each type of question. For the questions like "What is X?", our system shows relatively good performance. This means that AAD processing was effective for those questions.

According to table 3, we know that the systems in TREC-10 show slightly higher performance than the systems in TREC-9. But this does not necessarily refer to the improvement of the systems.

	TREC-10 67runs	TREC-9 35runs
Avg. MRR	0.234	0.22
Median MRR	0.121	0.115
# of Qs with no answer(%)	67.01 %	68.54 %

Table 3 The comparison between TREC-10 and TREC-9

3. Web track: Systems and Experiences

This is our first participation in the Web track of TREC. Our system is based on POSNIR/K, Korean natural language information retrieval system [4]. For TREC-10, we focused on effectiveness in both noun phrase extraction and PRF (Pseudo Relevance Feedback). While query expansion using PRF turned out to contribute to the performance significantly, the noun phrases were used with single terms actually didn't contribute much.

3.1 Keyword Extraction

For keyword extraction, we tagged the document collection, wt10g, and queries using POSTAG/E, the English POS (Part-Of-Speech) tagger based on HMM. The output of POSTAG/E is composed of lexis, POS tag, and lemma. From the result of the tagger, we selected keywords using two-phase extraction. If the lemmas were registered in the dictionary, they were selected. On the other hand, lexes were stemmed by Porter's stemmer[8] and then the stemmed lexes were selected as keywords. Stop words were eliminated using two kinds of stop list: common stop list containing 569 words, and query-specific stop list containing 28 words which must be removed from the query.

For constructing noun phrases, we made lexico-syntactic rules based on the POS-tag patterns. Some of the rules are described below.

Term1/{*NN* | *NP*} Term2/{*NN* | *NP*}
→ Term1_Term2

Term1/{*NN* | *NP*} (*'s/POS* | *of/IN*) Term2/{*NN* | *NP*}
→ Term1_Term2
Term1/*JJ* Term2/{*NN* | *NP*} Term3/{*NN* | *NP*}
→ Term1_Term2_Term3

3.2 Initial Retrieval

Our retrieval system uses 2-poisson model based on the probabilistic term distribution. The system retrieves top-raked documents after giving scores to each document of a target data collection with each query term list made from the keyword extraction process. For scoring, a rank system uses Okapi BM25 formula [9] as shown below.

$$w^{(1)} = \log\left(\frac{N - n + 0.5}{n + 0.5}\right) \tag{5}$$

$$Score(d,q) = \sum_{t \in q}\left(\frac{(k_1+1)\times tf_t}{k_1 \times ((1-b) + b \times \frac{dl_d}{avdl}) + tf_t} \times w^{(1)} \times \frac{(k_3+1)tf_q(q,t)}{k_3 + tf_q(q,t)}\right) \tag{6}$$

, where N is the number of documents in the collection, n is the number of documents containing the term, tf_t is the term frequency of term t in a document d, dl_d is the document length, $avdl$ is the average document length, $tf_q(q,t)$ is the term frequency of query term t in the query q, and k_1, b, $k3$ are tunable constant parameters.

3.3 Query Expansion

Query expansion is achieved through PRF (Pseudo Relevance Feedback). In the process of PRF, top-ranked documents are regarded as relevant and TSV (Term Selection Value) is given to all single terms except stop words in them. Then, top-ranked single terms are expanded and added to the original query term list. In this process, the weights of both original and expanded query terms are reweighted by Eq.(7) reflecting relevance and non-relevance information [10].

$$w^{(1)} = \frac{k_5}{k_5+\sqrt{R}}(k_4 + \log\frac{N}{N-n}) + \frac{\sqrt{R}}{k_5+\sqrt{R}}(\log\frac{r+0.5}{R-r+0.5}) - (\frac{k_6}{k_6+\sqrt{S}}\log\frac{n}{N-n}) - (\frac{\sqrt{S}}{k_6+\sqrt{S}}\log\frac{s+0.5}{S-s+0.5}) \tag{7}$$

, where N, n is the same as in the Eq.(5), R is the number of documents known to be relevant to a specific topic, r is the number of relevant documents containing the term, S is the number of documents known to be non-relevant, s is the number of non-relevant documents containing the term, and k_5, k_6 are tunable constant parameters.

For TSV function, we developed and compared some TSV formulas adapting diverse TF (Term Frequency) factors.

$$TSV = \left(\sum_{d \in R} 0.5 + 0.5 \times \frac{tf_{t,d}}{dl_d}\right) \times w^{(1)} \tag{8}$$

$$TSV = \left(\sum_{d \in R} \log(tf_{t,d} \times \frac{avgdl}{dl_d}\right) \times w^{(1)} \tag{9}$$

$$TSV = \left(\sum_{d \in R} \frac{tf_{t,d}}{k_1((1-b) + b\frac{dl_d}{avgdl}) + tf_{t,d}}\right) \times w^{(1)} \tag{10}$$

, where $w^{(1)}$ is Eq. (7).

3.4 Final Retrieval

Final retrieval process is the same as the initial one except that, this time, each query term has the new weights given by Eq. (7) and the expanded query term list is used.

3.5 Experiments in TREC-10

Table 4 summarizes the TREC-10 results. The results indicate that when a query was expanded using PRF, the performance was better, but noun phrases didn't give much contribution to the performance. As for TSV function in using PRF, Eq. (10) which is adapting TF factor of the weight formula of Okapi was better than any others.

In order to further validate the results, the t-test was performed on the data (Table 5). The table shows the mean difference, the standard deviation difference, the t-statistics and the probability of average precision and recall-precision for no-PRF (baseline) versus PRF (using Eq. (10)) case. Though there are no significant differences for average precision in TREC-9 topics, the table shows the rest of the performance are all significantly improved when PRF was used.

	No query expansion			Query expansion			
	title only		title+desc	title only			
	no phrases	phrases	phrases	phrases			
		baseline		Eq. (7)	Eq. (8)	Eq. (9)	Eq. (10)
TREC-9							
Precision	0.1740	0.1747	0.2188	0.1758	0.1740	0.1781	0.1837
R-Precision	0.1962	0.1967	0.2399	0.1954	0.1940	0.2049	0.2082
TREC-10							
Precision	0.1535*	0.1521**	0.1877***	-	-	-	0.1771****
R-Precision	0.1853	0.1760	0.2240	-	-	-	0.2081

Table 4 Average precision & R-Precision for TREC topics (* : posnire01st **: posnire01pt *: posnire01ptd ****: posnire01rpt)**

		Mean difference	STD difference	T	Prob > |T|
TREC-9	Precision	0.0091	0.0393	1.6279	0.1100
	R-Precision	0.0116	0.0430	1.9071	0.0624
TREC-10	Precision	0.0356	0.0939	2.6841	0.0099
	R-Precision	0.0480	0.0836	4.0577	0.0002

Table 5 T-test: Avg. Precision & R-Precision - no-PRF (baseline) vs. PRF (Eq. (10))

4. Conclusion

In TREC-10, we participated in the QA track and the Web track.

We submitted a run for the main task of the QA track and it was judged and evaluated by the reciprocal rank. The MRR for 492 questions is 0.320 (strict), which is considerably higher than the average MRR of other 67 runs.

In the Web track, we confirmed that our new query expansion using PRF with TSV function adapting TF factor contributed to better performance.

5. References

[1] Lehnert, W. (1978) The process of question answering: A computer simulation of cognition. Lawrence Erlbaum Associates.

[2] Graesser, A. C., Lang, K., & Horgan, D. (1988) A taxonomy of question generation. Questioning Exchange, 2, 3-16.

[3] Harabagiu, S., Moldovan, D., at al. (2000) FALCON: Boosting knowledge for answer engines. In Proceedings of the 9th Text REtrieval Conference (TREC-9).

[4] Lee, S. and Cho, B. (2000) Statistical Natural Language Query System THE IR based on P-Norm Model; Korean TREC-1, In *Proceeding of The 5th Korea Science & Technology Infrastructure Worksho*p, 189-202. (*in Korean*)

[5] Miller, G.A. (1995) WordNet: A lexical database, Communication of the ACM, vol 38: No11, pp 39-41.

[6] Moldovan, D., Harabagiu, S., at al. (1999) LASSO: A tool for surfing the answer net. In Proceedings of the 8th Text REtrieval Conference (TREC-8).

[7] NIST PRISE Search Engine: http://www.itl.nist.gov/div894/894.02/works/papers/zp2/main.html

[8] Porter, M.F. (1980) An algorithm for suffix stripping. Program, 14(3), pp 130-137.

[9] Robertson, S.E. et al. (1995) Okapi at TREC-3. In Overview of the Third Text Retrieval Conference (TREC-3). 109-126.

[10] Robertson, S.E and Walker, S. (1997) On relevance weights with little relevance information, In *Proceeding of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 16-24.

[11] TREC 2001 Question Answering Track Guidelines,

http://trec.nist.gov/act_part/guidelines/qa_track_spec.html

[12] Voorhees, Ellen M. and Dawn M. Tice, (2000) Building a question answering test collection. In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval

[13] Voorhees, Ellen M. and Dawn M. Tice, (1999) The TREC-8 question answering track evaluation. In Proceedings of the 8th Text REtrieval Conference (TREC-8).