

Singapore Management University Institutional Knowledge at Singapore Management University

Research Collection School Of Information Systems

School of Information Systems

11-2007

SLOQUE: Slot-based Query Expansion for complex questions

Maggy Anastasia SURYANTO

Ee Peng LIM

Singapore Management University, eplim@smu.edu.sg

Aixin SUN


Nanyang Technological University

Roger Hsiang-Li CHIANG

University of Cincinnati

DOI: <https://doi.org/10.1145/1317353.1317364>

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

 Part of the [Databases and Information Systems Commons](#), and the [Numerical Analysis and Scientific Computing Commons](#)

Citation

SURYANTO, Maggy Anastasia; LIM, Ee Peng; SUN, Aixin; and CHIANG, Roger Hsiang-Li. SLOQUE: Slot-based Query Expansion for complex questions. (2007). *Workshop on CyberInfrastructure: Information Management in eScience (CIMS2007)*, In conjunction with *ACM 9th International Workshop on Web Information and Data Management (WIDM 2007)*. Research Collection School Of Information Systems.

Available at: https://ink.library.smu.edu.sg/sis_research/1263

This Conference Proceeding Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email libIR@smu.edu.sg.

SLOQUE: Slot-based Query Expansion for Complex Questions

Maggy Anastasia Suryanto,
Ee-Peng Lim, Aixin Sun
Nanyang Technological University
{magg0002, aseplim, axsun}@ntu.edu.sg

Roger H.L. Chiang
University of Cincinnati
chianghl@ucmail.uc.edu

ABSTRACT

Searching answers to complex questions is a challenging IR task. In this paper, we examine the use of *query templates* with semantic slots to formulate *slot-based queries*. These queries have query terms assigned to entity and relationship slots. We develop several query expansion methods for slot-based queries so as to improve their retrieval effectiveness on a document collection. Each method consists of a combination of term scoring scheme, term scoring formula, and term assignment scheme. Our preliminary experiments evaluate these different slot-based query expansion methods on a collection of news documents, and conclude that: (1) slot-based queries yield better retrieval accuracy compared to keyword-based queries in the complex question problems; and (2) directly applying traditional query expansion on the query terms of each slot does not always work well.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval - Query formulations, Retrieval models

General Terms

Design, Algorithms, Experimentation

Keywords

Query expansion, slot based query, query template, relationship query, complex query

1. INTRODUCTION

1.1 Motivation

Search engines today are good at handling simple queries each represented by a bag of query terms. They however are not designed to evaluate queries derived from complex questions that involve entities and relationships. For such complex questions, users often need to convey more specific and complex information needs using either a natural

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIMS'07, November 9, 2007, Lisboa, Portugal.

Copyright 2007 ACM 978-1-59593-831-2/07/0011 ...\$5.00.

Question: What evidence is there for [transport] of [goods] from [source] to [destination]?

Slots: evidence of transportation :

goods :

source :

destination :

Figure 1: Query Template Example

language query statement or a query consisting of semantic slots.

The following shows an example complex query taken from the TREC's Complex and Interactive Question Answering (CiQA) 2006 track [9]. The query seeks to find the extent of illegal immigration from Albania to Italy and the steps taken to curb it.

Query Example: What evidence is there for [transport] of [illegal immigrants] from [Albania] to [Italy]?

This complex query, among with other similar queries that involve transportation of objects from one location to another can be grouped under the same query template as shown in Figure 1. The query template consists of query slots assigned with entity and relationship semantics. The entity slots are *goods*, *source* and *destination*. The relationship slot in this template is the *evidence of transportation*.

In this paper, we focus on complex queries formulated using the query templates containing both entity and relationship slots. Such complex queries are challenging to search engines as it is difficult to achieve both good precision and recall in their results.

Hearst [7] and Mitra et al [11] observe that a query may consist of multiple aspects (similar to slots). They showed that good precision in query results can be achieved by assigning query terms to the different aspects, and retrieving documents containing at least one query term from each aspect. Their work however does not help to improve recall for several reasons:

- *Term mismatch problem:* The relevant documents may contain terms similar but not identical to terms given in the query. As a result, they are not retrieved. This could happen when users could not provide query terms that are both relevant and complete.
- *Missing slot problem:* The relevant documents may not contain information that cover all aspects (or slots) of the query. Using the retrieval strategies proposed

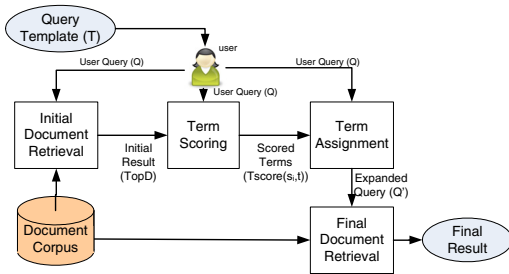


Figure 2: Slot-based Query Expansion framework.

by Hearst and Mitra, these documents will not be included in the query results.

To improve recall, we therefore study the slot-based query expansion (SLOQUE) task. A simple way to address the SLOQUE task is to apply traditional query expansion techniques on the slot query terms independently. However, query slots are not independent of one another and they may not carry equal importance. It is therefore important to study how query expansion can be extended to handle queries with multiple slots, and to evaluate the retrieval effectiveness of slot-based query expansion methods.

1.2 Overview of Slot-based Query Expansion

In this section, we introduce the general solution framework for SLOQUE task as shown in Figure 2. We assume that *query templates* each with one or more slots are available for users to select at the time a query is formulated. An *initial slot-based query* (Q) consists of query terms assigned to each slot.

Given a user query, the **initial document retrieval subtask** is to construct an initial query to the information retrieval (IR) system to obtain the top ranked documents, $TopD$, which represent a small set of relevant results from which expansion terms can be derived. The choice of the number of documents to be returned, $|TopD|$, is an interesting research problem[12]. It may be determined based on several criteria including the number of expected relevant documents, query difficulty level, number of slots, and others. In this paper, we will adopt some empirical $|TopD|$ values and leave the choice of optimum $|TopD|$ to our future research.

The goal of the **term scoring subtask** is to score and to rank all the potential expansion terms with respect to the slot-based query. Ideally, the expansion terms that are relevant should be given higher scores. The term scoring method could draw knowledge from the initial results $TopD$, the document corpus D , and other external knowledge bases.

The **term assignment subtask** aims to assign expansion terms to query slots. The number of expansion terms assigned to each slot has to be determined. The output of this subtask is an **expanded query** (Q'), which is to be used in the **final document retrieval subtask**, to retrieve the final document answer set.

1.3 Objectives and Contribution

The main objective of this paper is to develop different query expansion methods for slot-based queries based on the earlier framework. We seek to understand the effectiveness of slot-based queries and their query expansion for complex

questions such as those in CiQA task. To the best of our knowledge, this is the first time query expansion research is conducted on complex queries involving slots. The experience gain will help search engine developers in designing future search strategies for complex queries.

We now summarize our research contribution as follows:

1. We divide slot-based query expansion into several components including term scoring scheme, term scoring formula, term assignment scheme and final query format.
2. We then propose three term scoring schemes, known as *Non Slot-based Term Scoring (NTS)*, *Slot-based Term Scoring (STS)*, and *All Slot-based Term Scoring (ATS)*. Each scheme can be coupled with different term scoring formulas.
3. We introduce a new term scoring formula known as *Lexical, Local and Global Analysis (LLG)* that uses linear combination of lexical, local and global term knowledge to derive a term score. LLG outperforms *Local Context Analysis (LCA)*, a term scoring formula known to do well in traditional query expansion[16].
4. We develop three term assignment methods based on different final query formats. The query formats consider different (re-)groupings of query terms and expansion term into query slots before the final document retrieval steps.

In this work, we use Lucene search engine as the underlying retrieval system. Lucene [1] uses a simple extension to vector space model to support boolean queries. Lucene allows query terms or clauses to be declared *required* or *optional* as shown in the following example:

Example: $(+(s_{11}, s_{12}, s_{13}), +(s_{21}, s_{22}, s_{23}), (s_{31}, s_{32}, s_{33}))$

The example shows a Lucene query with 3 clauses. The '+' sign preceding the first and the second clauses declares that the two clauses are required. The third clause is optional. Given this query, Lucene will retrieve documents that contain at least one term from each required clause (the first and the second clauses) only. All query terms in the required and optional clauses affect the ranking of a document.

1.4 Paper Organization

This paper is organized as follows. The next section briefly describes some related work. Section 3 presents our slot-based query representation. Sections 4 and 5 describe in detail our term scoring and term assignment approaches respectively. Sections 6 and 7 present our experiment setup and the results. Section 8 concludes the work and provides some future directions.

2. RELATED WORK

In slot-based document retrieval, Kumaran and Allan [10] proposed various retrieval strategies for different relationship types asked by slot-based queries. However, their work cannot be generalized for all relationship types in slot-based queries and does not address the slot-based query expansion.

Jaana and Kalervo [8] applied query expansion on queries with structures. Their query structure refers to different search engine specific operators (i.e. SYN, SUM, WSUM,

etc.) and is different from the structure found in slot-based queries. Furthermore, they did not propose any new query expansion method.

Query expansion for non slot-based queries has been well studied. The term scoring techniques can be categorized under *lexical analysis*, *global analysis* and *local analysis*. Lexical analysis uses pre-constructed term relationship knowledge, such as thesauri, ontologies and dictionaries to determine expansion terms. For example, Voorhees explored different term relationships in WordNet for query expansion[15].

Global analysis uses corpus-wide statistics, say term co-occurrences, to expand a query [13, 5, 6]. Local analysis uses only some initial retrieved documents to expand a query. Some of the widely known works on local analysis include [14, 16, 3].

Yang et al used both local scores and lexicon scores to select expansion terms[18] without considering global scores. Thompson and Callan proposed to combine local, global co-occurrence score and lexicon score in different stages of random walk model[4]. Our term scoring formula, LLG (see Section 4.2), is the first that linearly combines local, global co-occurrence score and lexicon score.

3. QUERY REPRESENTATION AND INITIAL DOCUMENT RETRIEVAL

3.1 Query Template (T)

A query template provides the slot structure that is used for query formulation. A query template T is a tuple $T = \langle S, R \rangle$ where

- $S = (S_1, \dots, S_m)$ is a set of entity slots, $S_i = (S_i.label, S_i.term)$ and
- $R = (R_1, \dots, R_n)$ is a set of relationship slots, $R_j = (R_j.label, R_j.term)$

Entity slot S_i has a label $S_i.label$ and a set of initialized terms, $S_i.term$. These terms will be inherited by all instantiations of this slot. Similarly, a relationship slot R_j has a label $R_j.label$ and relationship term set $R_j.term$. Each relationship term carries semantics that link between entity slots.

Example 1: A query template example is shown in Figure 1. The formal definition is $T = \langle (S_1, S_2, S_3), (R_1) \rangle$ where

- $S_1 = (goods, \{ \});$
- $S_2 = (source\ location, \{ \});$
- $S_3 = (destination\ location, \{ \});$ and
- $R_1 = (evidence\ of\ transport, \{freighter, ship, boat, \dots, transport, transportation, truck, plane, car, terminal\})$

The template can be used to construct queries on movement of goods from one location to another location.

3.2 User Query (Q)

A user query Q , an instantiation of a query template $T = \langle (S_1, \dots, S_m), (R_1, \dots, R_n) \rangle$, is $Q = \langle (s_1, \dots, s_m), (r_1, \dots, r_n) \rangle$ where

- $s_i = \{s_{i1}, \dots, s_{ig_i}\}$ is the original entity term set for the i th entity slot, $S_i.term \subset s_i$

- $r_j = \{r_{j1}, \dots, r_{jh_j}\}$ is the original relationship term set for the j th relationship slot, $R_j.term \subset r_j$

We provide an example of user query that is instantiated from a template shown in Figure 1. The query tries to collect the evidence of transport of illegal immigrants from Albania to Italy.

$Q = (s_1, s_2, s_3), (r_1)$ where

- $s_1 = \{\text{illegal immigrants, immigrant}\};$
- $s_2 = \{\text{albania}\}$
- $s_3 = \{\text{italy}\}$
- $r_1 = \{\text{freighter, ship, boat, \dots, transport, transportation, truck, plane, car, terminal}\}$

The above definitions for slot-based queries may involve multiple entities and multiple relationships. In this paper, we will confine our discussions to only one relationship slot for each template, i.e. $n = 1$. We see this type of slot-based queries is very common. In particular, all the TREC's CiQA queries use only one relationship each.

3.3 Initial Document Retrieval

As part of initial document retrieval task, we may define many initial query formats each representing a way to evaluate the user query. Among them, we would like to find one that can return a small set of relevant documents for query expansion. Based on earlier claim that query results will enjoy better performance when the returned documents cover many of the aspects of the query [7, 11], we represent the initial query with format $((+s_1, \dots, +s_m), +r_1)$ as opposed to the usually used bag of query terms. E.g. the initial query of earlier example in this format is $((+\{\text{illegal immigrants, immigrant}\}, +\{\text{albania}\}, +\{\text{italy}\}), +\{\text{freighter, ship, boat, \dots, transport, transportation, truck, plane, car, terminal}\})$.

4. TERM SCORING

The goal of the term scoring is to obtain relevance score for each expansion term to an initial query. Since an initial query is instantiated from a query template with multiple slots, an expansion term may have relevance score(s) for each given slot or for all given slots.

4.1 Term Scoring Schemes

To determine whether a term t should be included as an expansion term for entity slots, we consider the following time scoring schemes:

1. **Slot-based Term Scoring (STS)** For m entity slots, the expansion term, t , will have $TScore(t, s_i)$, $i = \{1, \dots, m\}$. that measures the relevance of term t with respect to s_i .
2. **Non-slot based Term Scoring (NTS)** This scheme combines the query terms from all entity slots into a bag of terms $s = s_1 \cup \dots \cup s_m$, and assign the $TScore(t, s)$.
3. **All Slot-based Term Scoring (ATS)** is done by scoring the expansion term, t , with respect to all entity slots. Each term, t , will have exactly one score, $TScore(t, s) = Avg(\{TScore(t, s_i)\}_{i=1}^m)$, that measures the average relevance of t to all the entity slots. (Average function can be replaced with other reasonable functions that aggregate the values of $TScore(t, s_i)$ into $TScore(t, s)$)

4.2 Term Scoring Functions

We study two expansion term scoring $TScore(t, \cdot)$ functions used in term scoring schemes, namely **local context analysis (LCA)** function and the **lexical, local and global analysis (LLG)** function. The former has been used in non slot-based query expansion while the latter is a new function we propose for the slot-based queries.

LCA term scoring function

$$TScore(t, s_i) = \prod_{s_{ij} \in s_i} \left(\delta + \frac{\log(af(t, s_{ij}))}{\log(|TopD|)} \cdot idf_t \right)^{idf_{s_{ij}}} \quad (1)$$

$$\text{where } af(t, s_{ij}) = \sum_{d_k \in TopD} TF_k(t) \cdot TF_k(s_{ij})$$

$$idf_t = \max \left(1.0, \frac{\log_{10} \left(\frac{|D|}{|D(t)|} \right)}{5} \right)$$

$$TF_k(t) = \text{number of times } t \text{ appears in } d_k$$

$$D(t) = \text{documents containing } t$$

$$\delta = 0.1 \text{ to avoid zero } TScore() \text{ value}$$

LCA favors terms with high term frequency ($TF_k(t)$) in documents within $TopD$ that contain original query terms with high $TF_k(s_{ij})$. This is captured by the $af(t, s_{ij})$. The term idf_t penalizes terms occurring too many times in the corpus, the term $idf_{s_{ij}}$ emphasizes infrequent query terms.

The term scoring function for the relationship slot $TScore(t, r_1)$, using LCA can be defined similarly by replacing s_i with r_1 .

LLG term scoring function

LLG draws knowledge from lexicon, local and global analysis as shown in Equation 2.

$$TScore(t, s_i) = \alpha \cdot LexScore(t, s_i) + (1 - \alpha)[\beta \cdot LScore(t, s_i) + (1 - \beta) \cdot GCScore(t, s_i)] \quad (2)$$

LLG adopts a linear combination of lexicon score, local analysis score and global analysis score. The three component scores are normalized to the range of $[0, 1]$ by their maximum scores over all potential expansion terms over all slots.

The lexicon score $LexScore$ is derived from synonymy relationships among terms in WordNet[2]. $LexScore(t, s_i)$, is defined as follows:

$$LexScore(t, s_i) = \frac{\sum_{s_{ij} \in s_i} isSynonym(t, s_{ij})}{|s_i|} \quad (3)$$

where

$$isSynonym(t, s_{ij}) = \begin{cases} 1 & \text{if } t \text{ is a synonym of } s_{ij}; \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

The more terms in s_i that are synonymous to t , the higher the $LexScore(t, s_i)$.

The local closeness score, $LScore(t, s_i)$, is defined as follow:

Table 1: LLG with different α, β combinations.

α	β	Components used
0	0	GCScore
0.5	0	LexScore+ GCScore (equally weighted)
1	N/A	LexScore
0	1	LScore
0	0.5	LScore+ GCScore (equal weightage)
0.5	1	LexScore+ LScore (equal weightage)
0.33	0.5	All (equally weighted)

$$LScore(t, s_i) = IDF(TopD, t) \cdot \quad (5)$$

$$\sum_{d_k \in TopD} \sum_{s_{ij} \in s_i} cntSent_k(t, s_{ij})$$

$$\text{where } IDF(TopD, t) = \log \left(1 + \frac{|D| - |TopD|}{|D(t)| - |TopD(t)|} \right)$$

$cntSent_k(t, s_{ij})$ returns the number of sentences in d_k that contain t and s_{ij} , $TopD(t)$ are the documents in $TopD$ containing t . A term is assigned high $LScore(t, s_i)$ if it collocates frequently with many members of s_i within $TopD$. LScore favors terms that appear closely (within a sentence) with many query slot terms.

The global closeness score, $GCScore()$, measures the probability of co-occurrence between a term and query terms of a slot across the entire document corpus as shown below. This scoring function assumes that the correlation between a term t and any term $s_{ij} \in s_i$ is independent of one another.

$$\begin{aligned} GCScore(t, s_i) &= 1 - \overline{GCScore(t, s_i)} \\ &= 1 - \prod_{s_{ij} \in s_i} \overline{GCScore(t, s_{ij})} \\ &= 1 - \prod_{s_{ij} \in s_i} \left(1 - \frac{cn(t, s_{ij})}{cn(t) + cn(s_{ij}) - cn(t, s_{ij})} \right) \end{aligned} \quad (6)$$

where $cn(t, s_{ij})$ is the number of times terms t and s_{ij} co-occur within a sliding window of SW terms (we set $SW = 20$ in all our experiments) and $cn(t)$ is the total co-occurrence counts of term t with the other terms in the whole corpus. The distance between terms can be easily derived from the index structure.

An expansion term t receives high $GCScore(t, s_i)$ if it collocates frequently with many members of s_i in the global corpus (global closeness).

$LexScore(t, r_1)$, $LScore(t, r_1)$ and $GCScore(t, r_1)$ are defined similarly as above with s_i replaced by r_1 .

Table 1 shows interesting combinations of α and β that result in different LLG score functions.

$TScore(t, r_1)$, which is the relevance score of term t to r_1 , is defined similarly as above with s_i replaced by r_1 .

5. TERM ASSIGNMENT

Term assignment has three important roles: 1) to decide the expanded query format; 2) to decide the number of expansion terms for each slot; 3) to decide which slot each expansion term is to be assigned to. An *expanded query* Q' is obtained after expanding an user query Q .

There can be different formats of expanded queries in the SLOQUE task. The choice of query format affects the decision of number of expansion terms as well as the assignment of an expansion term to a particular slot. The following are 3 expanded (or final) query formats studied.

1. **Final Query 1 (QF1):** $QF1 = \langle (+s', +r'_1) \rangle$, where

- $s = s_1 \cup \dots \cup s_m \subseteq s'$ and $r_1 \subseteq r'_1$
- $s' - s$ are the expansion terms for all entity slots, and $r'_1 - r_1$ are expansion terms for r_1

QF1 enforces the presence of at least one term from s' and at least one term from r'_1 for a document to be returned by the IR system.

2. **Final Query 2 (QF2):** $QF2 = \langle (+s'_1, \dots, +s'_m), +(r'_1) \rangle$ where

- $s_i \subseteq s'_i$ and $r_1 \subseteq r'_1$
- $(s'_i - s_i)$ are new expansion terms for s_i and $(r'_1 - r_1)$ are new expansion terms for r_1

QF2 enforces at least one term from each of s'_i , $i = \{1, 2, \dots, m\}$ and at least one term from r'_1 for a document to be retrieved.

3. **Final Query 3 (QF3):** $QF3 = \langle (+s_1, \dots, +s_m), (s' - s), +(r'_1) \rangle$.

In QF2, the expansion term set $s' - s$ is assigned a new slot termed as the auxiliary entity slot. QF3 enforces at least one original query term from each of s_i , $i = \{1, 2, \dots, m\}$ and at least one term from r'_1 for a document to be retrieved. The auxiliary term set, $s' - s$, is designed to enhance the ranking of those documents satisfying the slot-based query needs.

Another important parameter of the step is the number of expansion terms to be assigned. In this paper, we assign equal number of expansion terms for all the slots. We use M to denote the number of expansion terms for each entity and relationship slot.

If STS is the term scoring scheme, we have m values of $TScore(t, s_i)$ for each expansion term t . Term assignment is done differently for QF2 compared to QF1 and QF3 as shown below:

$$\mathbf{QF2} : s'_i - s_i = \{t | t \notin s_i, rank(t, s_i) \leq M\}$$

$$\mathbf{QF1 \ or \ QF3} : s' - s = \bigcup_{s_i \in s} \{t | t \notin s, rank(t, s_i) \leq M\}$$

If NTS or ATS is the term scoring scheme, we have a $TScore(t, s)$ value for each expansion term t . Term assignment is done for QF1, QF2 and QF3 as shown below:

$$\mathbf{QF2} : s'_i - s_i = \{t | t \notin s_i, rank(t, s) \leq M\}$$

$$\mathbf{QF1 \ or \ QF3} : s' - s = \{t | t \notin s, rank(t, s) \leq M \times m\},$$

All expansion terms in $s' - s$ are unique.

To give more importance to the original query terms, similar to query expansion framework adopted by [16, 17], we adopt a query weighting scheme for the final expanded queries as follows:

- Each original query term in s_i receives a weight of two.
- Each expansion term t in $s'_i - s_i$ is assigned a weight of $1 - 0.9(rank(t, s_i)/M)$

Lucene allows a boost factor to be assigned to each query term. We therefore use query term weights as boost factors in our experiments.

id	T (query template)	R _i label	R _i term
1	What evidence is there for [transport] of [goods] from [source location] to [destination location]?	evidence of transportation	ship, cargo, vessel, boat, vehicle, container, port, train, transport, pipeline, link, passenger, freight, plane, shipment, terminal
2	What [financial relationships] exist between [entity1] and [entity2]?	financial relationship	dollar, money, fund, budget, fee, expense, income, market, price, business, profit, grant, tax, revenue, transfer, financial, bank, trade
3	What [familial ties] exist between [entity1] and [entity2]?	familial ties	relation, mother, father, cousin, brother, sister, descendant, ancestor, related to, family, link, link, relationship
4	What [common interests] exist between [entity1] and [entity2]?	common interests	common, interest, collaborate, cooperate, cooperation, share, joint
5	What [influence/effect] do(es) [subject entity] have on/in [object entity]?	influence/effect	because, cause, lead to, increase, lower, decrease, affect, effect, influence, help, risk
6	What is the [position] of [entity] with respect to [issue]?	position	against, for, agree, agreement, oppose, opposition
7	Is there [evidence to support the involvement] of [subject entity] in [object event/entity]?	evidence of involvement	involve, involved, engage, engaged, support

Figure 3: TREC CiQA’s 7 templates

Table 2: Distribution of the number of relevant documents.

average	minimum	maximum
17.2	8	41

6. EXPERIMENTS

The experiments were conducted to compare the different term scoring schemes, term scoring formulas and term assignment schemes when applied to SLOQUE task. We would like to know how effective are they in retrieving the documents carrying relevant information for complex query answering tasks. We also evaluate how the methods using LLG behave when different α and β values are used.

6.1 Dataset and Queries

We use TREC CiQA 2006’s query topics (topic 26-55) and the AQUAINT news corpus in our experiments [9]. For each query topic, CiQA task requires each participant to submit their answers in the form of sentences. From these sentences, an NIST assessor then looks for relevant snippets. We construct a set of relevant documents that are the ones containing sentences which cover the relevant snippets.

We exclude 3 query topics with too few number of relevant documents. The remaining 27 topics can be classified into 7 groups each using the same query template. As shown in Figure 3, the 7 query templates cover different types of relationships. In our experiments, we carefully selected the query terms for the relationship slot and used them in all the queries sharing the same query template.

The relevant set is a partial ground truth set as it was derived from CiQA 2006 participants’ submissions only. Table 2 presents the distribution of the number of relevant documents over all evaluated topics.

6.2 Evaluation Metrics

We use *Recall* and *MRR-normalized* (MRR-norm) metrics to evaluate retrieval performance. We measure Recall and MRR-norm at top 20 and 100 retrieved documents.

$$Recall@P = \frac{|A_P \cap R|}{|R|} \quad (7)$$

$$MRR - norm@P = \frac{\sum_{d \in R \cap A_P} \frac{1}{rank(d)}}{\sum_{j=1}^{|R|} \frac{1}{j}} \quad (8)$$

where A_P is the top P retrieved documents using the evaluated method, R is the relevant set, and $rank(d)$ is the rank of relevant document d in A_P .

The same numbers of retrieved documents (i.e., 20 and 100) have also been commonly used by the TREC participants to select documents for extracting answer sentences.

6.3 Experiment Settings

We evaluate several methods to solve SLOQUE task by combining different scoring schemes (STS, NTS, ATS), term scoring formulas (LCA, LLG), and final query formats (QF1 to QF3). These methods are named following the convention (term scoring scheme)((term scoring formula)-(final query format)).

Our first set of experiments evaluates the methods using LLG for different α and β values shown in Table 1. Our second set of experiments evaluates the performance of the methods using LCA and LLG score formulas separately, we then pick the best two methods from each and compare them. In the following, we summarize the evaluated methods:

1. STS(LCA)-QF2 and STS(LLG)-QF2. These two methods score the relevance of each expansion term with respect to each entity slot (STS scheme). For each entity slot, we assign the top M expansion terms and construct the final query using QF2.
2. NTS(LCA)-QF1 and NTS(LLG)-QF1. These two methods score the relevance of each expansion term to the bag of entity terms from all entity slots (NTS scheme). The top $M \times m$ expansion terms are assigned to the bag of entity terms in the final query using QF1.
3. NTS(LCA)-QF2 and NTS(LLG)-QF2. These two methods use NTS scheme to score expansion terms. The top M expansion terms are assigned to each entity slot of the final query using QF2.
4. NTS(LCA)-QF3 and NTS(LLG)-QF3. These two methods use NTS scheme to score expansion terms. The top $M \times m$ expansion terms are assigned to the auxiliary slot of the final query in QF3.
5. ATS(LCA)-QF3 and ATS(LLG)-QF3. These two methods score the relevance of each expansion term with respect to all entity slots (ATS-scheme) using *average* function. The top $M \times m$ expansion terms are assigned to the auxiliary slot of the final query in QF3.

Recall that $|TopD|$ refers to the size of initial query result used for query expansion. We empirically set $|TopD| = 10$ as the minimum number of relevant documents is 8 (see Table 2). To vary the number of expansion terms per slot, we try different M values from 1 to 5. We use the run that does not involve query expansion as our baseline, **B1**.

7. EXPERIMENT RESULTS

7.1 Determining Appropriate α and β Values for Methods using LLG

We tried different combinations of α and β values for 5 evaluated LLG methods at all the evaluated metrics. As the best results are consistent among these methods across different metrics, we only show the Recall and MRR-norm at 100 results for NTS(LLG)-QF3 using different α and β values as shown in Figure 4. The following describes the behavior of different combination on NTS(LLG)-QF3. All

other methods show similar tendency though not exactly the same ordering of performance among different combinations.

As shown in the figure, the combination ($\alpha=0.33, \beta=0.5$) yielded the best results in Recall and MRR-norm. This suggests equal weightage to LexScore, LCScore and GC-Score is a reasonable choice. GC-Score-only ($\alpha=0, \beta=0$) was the worst performing combination. Figure 4 also shows that combining different scoring components can improve the performance of expanded queries. For example, the combined GC-Score and LCScore scoring formula using $\alpha=0, \beta=0.5$ was shown to improve over GC-Score-only and LCScore-only methods. For the remaining experiments, we will therefore use $\alpha=0.33$ and $\beta=0.5$ for all methods using LLG.

7.2 Comparing Methods using LLG and LCA

The final set of experiments aims to compare the methods using LLG and LCA separately and to compare the best methods of LLG and LCA.

Comparison of Methods using LLG

Figure 5 shows performance of various methods using LLG.

The following is the summary of the general behavior of the methods obtained from the figure:

- NTS(LLG)-QF3 yielded the best MRR-norm (the figure only shows MRR-norm at 20 since the performance is consistent with that of at 100).
- NTS(LLG)-QF1 yielded the best Recall@100 but only the second best Recall@20.
- NTS(LLG)-QF2 was the worst among all the evaluated methods in both Recall and MRR-norm.
- STS(LLG)-QF2 was worse than NTS(LLG)-QF1 and NTS(LLG)-QF3 in both Recall and MRR-norm.
- ATS(LLG)-QF3 yielded the second best MRR-norm but not on Recall.

Generally, when LLG is used, NTS scheme is better than STS scheme. This seems to confirm the claim made by Xu and Croft[17] as well as Qiu and Frei [13] that a good expansion term must be “close” to “all” aspects of a query instead of only one of them.

Using STS, relevant terms may not be ranked highly since they are relevant only to the combined slot, s , instead of to any single slot. On the other hand, NTS manages to score these terms highly.

Even ATS using *average* score over all slots is not as good as NTS. $TScore(t, s)$ using ATS scheme is derived from $TScore(t, s_i)$ using STS scheme which score relevance of t to different slots independently. $TScore(t, s)$ using ATS scheme receives high score if term t receives high score (using STS) to many slots or extremely high score (using STS) to a single slot. The latter is not a good criteria to choose relevant expansion terms. In contrast to independently scoring the relevance of t to each slot, NTS scheme pools the terms/slots together and scores each expansion term t to the pool of terms.

It is more difficult to make general comparison on the second factor, QF1, QF2 and QF3 since their performances depend on the use of STS, NTS or ATS. Thus, we compare QF1, QF2 and QF3 together with the term scoring scheme used.

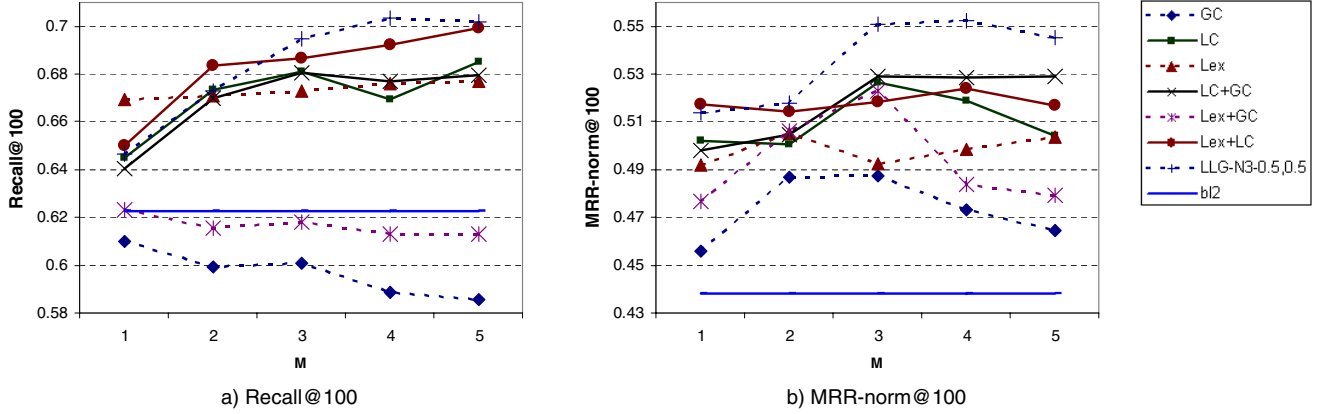


Figure 4: Performance of LLG with different combinations of α and β

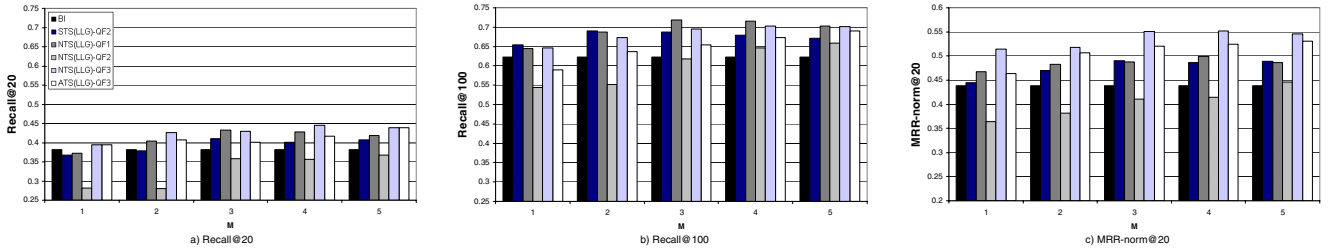


Figure 5: Performance of STS(LLG)-QF2, NTS(LLG)-{QF1/QF2/QF3} and ATS(LLG)-QF3 with $\alpha = 0.33$ and $\beta = 0.5$

As shown in Figure 5, NTS(LLG)-QF1 was the best recall for large number of retrieval (i.e. $P = 100$), but it was the second best for $P = 20$. QF1 combines all entity slots into one slot and relaxes the selection criteria of documents to be retrieved. The relaxation explains why its MRR-normalized performance cannot outperform some of the other methods.

Knowing that slot-based is better than bag of terms (see Section 3.3) and NTS is generally better than ATS and STS, it is natural to think that combining NTS and QF2 may yield good performance. However, the experiment results showed otherwise. Hence, combining NTS and QF2 is not a good idea.

As shown in Figure 5, NTS(LLG)-QF2 almost consistently performed below the baseline. This is because NTS(LLG)-QF2 gives top terms ranked by NTS to all the slots such that all slots share exactly the same expansion terms. This greatly relaxes the selection criteria and significantly boosts the expansion terms importance with respect to the original query terms. This will not only reduce precision (in terms of MRR-norm) but also recall. Similar behavior was shown in the performance of ATS(LLG)-QF2. This result is omitted for conserving space.

This leads us to the combination of NTS and QF3. It accommodates the assignment of expansion terms scored by NTS to a new slot instead of to existing slots, preserving the selection criteria of original query and the high importance of original query terms. Compared with baseline (BI), NTS(LLG)-QF3 with $M = 4$ improved Recall@20 and MRR-norm@20 by 17% and 27% respectively.

Comparison of Methods Using LCA

The performance of methods using LCA was consistent with the performance of methods using LLG. NTS(LCA)-QF3 gave the best performance. Generally NTS is better than STS and ATS (except when it is combined with QF2). Compared with baseline (BI), NTS(LCA)-QF3 improved Recall@20 with $M = 5$ and MRR-norm@20 with $M = 2$ by 10% and 17% respectively.

Comparing Best Methods using LCA and Best Methods using LLG

This section compares the performance of LCA and our new proposed score function, LLG. We choose to compare the two best methods for LCA and two best methods for LLG, i.e. NTS(LCA)-QF1, NTS(LCA)-QF3, NTS(LLG)-QF1 and NTS (LLG)-QF3.

As shown in Figure 6, LLG performed better than LCA for almost all parameter settings

Even using LCscore alone (LLG with $\alpha = 0$ and $\beta = 1$) is already better than LCA. LCA favors terms with high term frequency from documents within $TopD$. On the other hand, LCscore imposes stricter correlation criteria. LCscore favors terms that appear closely (within a sentence) with many original query terms.

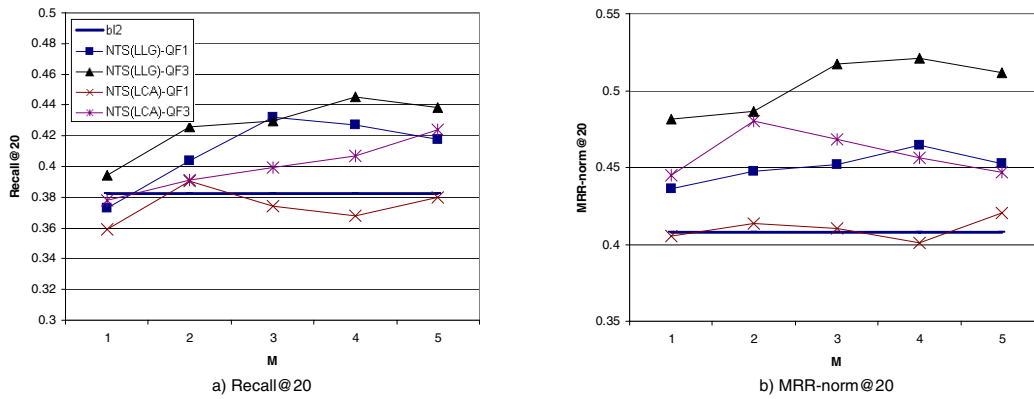


Figure 6: Performance of BI, best methods of LLG, and best methods of LCA.

8. CONCLUSION AND FUTURE WORK

Slot-based queries are designed to support complex questions. Slot-based queries can improve retrieval performance compared to the traditional bag-of-term queries. In this paper, we propose different query expansion methods for slot-based queries. Our best method can improve both recall and MRR-normalized by 17% and 27% compared with methods without using expansion.

Our key findings include: (a) the LLG term scoring formula performs well when it gives equal importance to lexical, local closeness and global closeness term knowledge; (b) the NTS, STS and ATS term scoring schemes using LLG with some final query formats can improve retrieval accuracy, outperforming the schemes using LCA; and (c) NTS using LLG or LCA produces good ranking of expansion terms compared to ATS and STS.

As part of our future work, a more comprehensive set of experiments will be conducted to compare the different query expansion methods. In particular, we would like to consider multiple relationship slots, the choice of $|TopD|$, and semantic constraints on the slots.

9. REFERENCES

- [1] Apache lucene - overview. <http://lucene.apache.org/java/docs/>.
- [2] Wordnet. <http://wordnet.princeton.edu/>.
- [3] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using smart:trec. In *TREC-3*, pages 4–11, NIST, 1994.
- [4] K. Collins-Thompson and J. Callan. Query expansion using random walk models. In *ACM CIKM*, pages 704–711, Bremen, Germany, 2005.
- [5] C. J. Crouch and B. Yang. Experiments in automatic statistical thesaurus construction. In *ACM SIGIR*, pages 77–88, Copenhagen, Denmark, 1992.
- [6] S. Gauch, J. Wang, and S. M. Rachakonda. A corpus analysis approach for automatic query expansion and its extension to multiple databases. *ACM TOIS*, 17(3):250–269, July 1999.
- [7] M. A. Hearst. Improving full-text precision on short queries using simple constraints. In *Proceedings of the Symposium of Document Analysis and Information Retrieval*, pages 1–16, Las Vegas, USA, 1996.
- [8] J. Kekalainen and K. Jarvelin. The impact of query structure and query expansion on retrieval performance. In *ACM SIGIR*, pages 130–137, Melbourne, Australia, 1998.
- [9] D. Kelly and J. Lin. Trec 2006 ciqa task guidelines. <http://www.umiacs.umd.edu/~jimmylin/ciqa/guidelines.html>, 2006.
- [10] G. Kumaran and J. Allan. Information retrieval techniques for templated queries. In *RIAO*, 2007.
- [11] M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *ACM SIGIR*, pages 1–15, Melbourne, Australia, 1998.
- [12] J. Montgomery, L. Si, J. Callan, and D. A. Evans. Effect of varying number of documents in blind feedback: analysis of the 2003 nrcc ria workshop bf-numdocs experiment suite. In *ACM SIGIR*, pages 476–477, 2004.
- [13] Y. Qiu and H. Frei. Concept based query expansion. In *ACM SIGIR*, pages 160–169, Pittsburgh, PA, USA, 1993.
- [14] J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323, Englewood Cliffs, NJ, 1971. Prentice-Hall Inc.
- [15] E. M. Voorhees. Query expansion using lexical-semantic relations. In *ACM SIGIR*, Dublin, Ireland, 1994.
- [16] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *ACM SIGIR*, pages 4–11, New York, NY, 1996.
- [17] J. Xu and B. W. Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM TOIS*, 18(1):79–112, January 2000.
- [18] H. Yang, T.-S. Chua, S. Wang, and C.-K. Koh. Structured use of external knowledge for event-based open domain question answering. In *ACM SIGIR*, pages 33–40, Toronto, Canada, 2003.