Mining Term Association Patterns from Search Logs for Effective Query Reformulation

Xuanhui Wang
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
xwang20@cs.uiuc.edu

ChengXiang Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
czhai@cs.uiuc.edu

ABSTRACT

Search engine logs are an emerging new type of data that offers interesting opportunities for data mining. Existing work on mining such data has mostly attempted to discover knowledge at the level of *queries* (e.g., query clusters). In this paper, we propose to mine search engine logs for patterns at the level of *terms* through analyzing the relations of terms *inside* a query. We define two novel term association patterns (i.e., context-sensitive term substitutions and term additions) and propose new methods for mining such patterns from search engine logs. These two patterns can be used to address the mis-specification and under-specification problems of ineffective queries. Experiment results on real search engine logs show that the mined context-sensitive term substitutions can be used to effectively reword queries and improve their accuracy, while the mined context-sensitive term addition patterns can be used to support query refinement in a more effective way.

Categories and Subject Descriptors: H.3.1 [Content Analysis and Indexing]: Linguistic processing; H.3.3 [Information Search and Retrieval]: Query formulation, Search process

General Terms: Algorithms

Keywords: Term association patterns, search log mining, query reformulation

1. INTRODUCTION

As search engines are being used, they naturally accumulate a lot of log data, including submitted queries, viewed search results, and clicked URLs. Such search engine logs contain a lot of valuable information such as patterns of query reformulation. In general, a Web search engine answers millions of queries every day. Thus the huge amount of search engine log data offers excellent opportunities for data mining. Indeed, mining search engine logs has recently attracted much attention [22, 16, 11, 1, 8, 25, 21]. All these studies have shown the promise of improving search accuracy through mining search engine logs. However, virtually all the previous work has treated *a whole query* as a unit for analysis; as a result, the discovered knowledge is mostly at the level of queries. For example, clustering search queries is studied in [26, 4]. The similarity of queries can be measured by the clicked documents [26] or their temporal correlations [6, 23]. Existing query suggestion works such as [19] and [12] also consider a whole query as a unit and they further rely on other resources such as Web snippets [19] or human-labeled training data [12] to generate related queries. Furthermore, most of the work only suggests "related" queries and does not consider the effectiveness of the suggested queries, which is very crucial for successful query suggestions.

In this paper, we look into patterns at the level of *terms* through analyzing the relations of terms *inside* a query and use the discovered term association patterns for *effective* query reformulation. Our work is motivated from the following observations about what types of knowledge are useful to help a user formulate an effective query. A query is ineffective due to multiple reasons, but two of them are common: *mis-specification* and *under-specification*.

(1) The mis-specification problem is caused by the fact that there may be multiple ways of expressing the same idea or describing the same thing, and a user may not know what exact terms have been used by the authors of the documents to be searched. This is also called "vocabulary mismatch." For example, if a user wants to find a place to wash his/her vehicle, a good query would be "car wash". If the user uses a query such as "auto wash" or "vehicle wash", the search results are generally not as good as those from using the query "car wash" even though all these queries have roughly the same meaning. This is because in most relevant web pages, the authors used "car wash" rather than "vehicle wash" or "auto wash." In order to help a user in such a case, we need knowledge of the form "auto→car | _ wash" (i.e., in the context "_ wash", it is better to replace "auto" with "car"). This is an example of what we refer to as a context-sensitive term substitution pattern.

(2) The under-specification problem in a query may be because the user does not know much about the content to be found or can not naturally think of additional specific terms. For example, a query such as "auto quotes" can return mixed results with some about automobile insurance quotes and some about automobile sale prices. In such a case, it would be useful to suggest terms such as "insurance" and "sale" for a user to choose so as to make the query more

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.

discriminative. In order to do this, we need knowledge of the form "+insurance | auto _ quotes" and "+sale | auto _ quotes" (i.e., in the context of "auto _ quotes", "insurance" and "sale" are possibly useful terms to refine the query at a specified position). This is an example of what we refer to as a context-sensitive term addition pattern.

In this paper, we first formally define the two novel term association patterns in search logs – context-sensitive term substitution and addition patterns. Then we propose new probabilistic methods to discover these patterns through analyzing term co-occurrences in query logs. Our basic idea is to analyze the co-occurrences of terms within multi-word queries in logs and obtain two kinds of term relations: (1) quasi-synonyms and (2) contextual terms. Quasi-synonyms are words that are synonyms (e.g., auto and car) or that are syntactically substitutable (e.g., yahoo and google) [10]. Such terms tend to co-occur with the same or similar terms; for example, both "auto" and "car" often occur together with "rental", "pricing", etc. We propose to use probabilistic translation models for capturing quasi-synonyms. Contextual terms are terms that appear together. For example, "car" and "insurance" often co-occur in the queries and they can help each other to refine a topic – "car insurance" can be used to refine both "car" and "insurance". We propose to use probabilistic contextual models for capturing contextual terms. Based on both translation models and contextual models, we cast our context-sensitive term association pattern mining as probability estimation problems. Patterns with high probabilities are with high confidence and then used for query reformulation. For example, "car" has a high probability in the translation model of "auto" and high probability to co-occur with "wash" in contextual models, then the pattern "auto→car|_ wash" will have a high probability and thus is a pattern with high confidence.

To test the effectiveness of our proposed algorithms, we conduct experiments on a sample of search logs. Experimental results on the real search engine logs show that our proposed methods can efficiently and effectively mine term association patterns and all these patterns can be used for effective query reformulation. These show that our proposed methods can discover useful knowledge based on the term relations inside queries. Our methods are totally orthogonal to, and thus can be enhanced by, other techniques which use other information such as click-through and user session data for query suggestions.

The rest of the paper is organized as follows. We first review the related work in Section 2. Then we formally define our mining problem in Section 3 and propose our models to discover term association patterns in Section 4. Our search log data collection is described in Section 5 and the experiments are presented in Section 6. Finally we conclude this paper and discuss future work in Section 7.

2. RELATED WORK

Our work is highly related to query suggestion works such as [19] and [12]. In [19], the similarity between two queries are measured by their retrieved snippets from a search engine. In [12], adjacent query pairs from the same user sessions are used as candidates and machine learning algorithms are used to categorize query pairs into 4 classes, which reflect levels of relevance between two queries. The main difference of our work is that we primarily discover patterns in *term* level and use the discovered pattern to recommend more *effective* queries, while previous work does not consider the effectiveness of a query and only focuses on finding the generally related queries in the level of *queries*. Furthermore, they always rely on external resources such as a Web corpus or training data, while our methods only need search logs.

Queries	Clicked URLs	Time
hotel taxes in las vegas	http://xxx.xxx.xxx/	xxxx
las airport	NO CLICKS	xxxx
las vegas airport	http://xxx.xxx.xxx/	xxxx
	http://xxx.xxx.xxx/	xxxx
...	...	...

Table 1: An example of user sessions

Our methods to mining term association patterns are related to translation models for natural languages. Traditional translation models [5] are designed to learn translating *word* pairs of different languages (e.g., English and French) based on the training data which, in general, consists of translating *sentence* pairs. In this paper, our translation model between different words is based on the bridge of their contexts. Based on similar ideas, there are several related works, such as [13] and [10], which identify synonyms or near-synonyms in text corpus. Our two types of term association patterns are closely related to the syntagmatic and paradigmatic word relations [17, 9]. The difference is that our work is based on search logs and we further use them for query reformulation.

Our work is also related to query modification work in information retrieval community [20]. The study of query modification can be traced back to the earliest relevance feedback techniques such as the Rocchio method [18], in which queries are modified based on the documents which are judged to be relevant and irrelevant. When all the top documents are irrelevant, negative feedback can be employed [24]. Pseudo-relevance feedback is to simulate relevance feedback by assuming top ranked documents of an initial retrieval as relevant ones [27]. In [2], a system Prisma is studied and it can recommend related terms and users can narrow the search results by selecting appropriate related ones to refine their queries. All these approaches depend only on the original queries and their initial retrieved documents for query refinement. Our query reformulation algorithms are based on the term association patterns mined from many queries accumulated by search engines, thus in a *collective* and *collaborative* way. Our methods rely on users' past activities recorded in search logs to discover term association patterns and thus can reflect users' preferences more appropriately.

Our context-sensitive query rewording is also related to the spelling correction [7] and context sensitive stemming [15]. But our method is to recommend a more *appropriate* word to replace the original one, which is not necessarily a misspelling or a stem. In this sense, our work can be regarded as a "semantic" extension of previous works which rely on "morphological" forms.

3. PROBLEM FORMULATION

Search engine logs record the activities of web users, which reflect the actual general users' need or interests when conducting a search. Generally, search engine logs have the following information: text queries that users submitted, the

time when they searched, and the URLs that they clicked after the queries. Search engine logs are separated by user sessions. A user session includes several queries from the same user for a coherent information need and the clicked URLs for each query in the session. An example of user sessions is shown in Table 1. In this paper, we focus on the pattern inside *queries* in search logs and we formally define our problem of mining term association patterns in this section.

DEFINITION 1 (QUERY). *A query q of length n with vocabulary V is an ordered sequence of terms $[w_1 w_2 ... w_n]$, where $w_i \in V$ for all $1 \leq i \leq n$. We use $w \in q$ if term w is contained in q.*

DEFINITION 2 (QUERY COLLECTION). *A query collection Q consists of a bag of N queries: $Q = \{q_1, q_2, ..., q_N\}$. The queries are not necessarily distinct from each other.*

For example, all the queries submitted to a search engine in a certain period of time form a query collection. A query collection provides us data for mining term association patterns. We now define two interesting patterns in search logs.

DEFINITION 3 (CONTEXT-SENSITIVE TERM SUBSTITUTION). *A context-sensitive term substitution pattern is in the form of $[w \rightarrow w' | c_L \_ c_R]$. c_L and c_R are left and right context words and this pattern means that term w should be substituted by term w' given a specific context.*

DEFINITION 4 (CONTEXT-SENSITIVE TERM ADDITION). *A context-sensitive term addition pattern is in the form of $[+w | c_L \_ c_R]$. This pattern means that term w can be added into the context $c_L \_ c_R$ and thus forms a new sequence $c_L w c_R$.*

The defined term association patterns can be easily extended for query reformulation. We define two types of query reformulation, query rewording and query refinement, in the following and they are to address the mis-specification and under-specification problems of an ineffective query respectively.

DEFINITION 5 (QUERY REWORDING). *Given a query $q = w_1 w_2 ... w_n$, query rewording is to modify the query by replacing one term w_i in q by its semantically similar term s, thus form a new query $q' = w_1 ... w_{i-1}, s, w_{i+1} ... w_n$.*

DEFINITION 6 (QUERY REFINEMENT). *Given a query $q = w_1 w_2 ... w_n$, query refinement is to modify the query by adding one semantically related term r to q before a position i, thus form a new query $q' = w_1 ... w_{i-1} r w_i ... w_n$.*

It can be seen that query rewording and query refinement correspond to context-sensitive term substitution and term addition patterns respectively. In practice, query rewording and refinement involve multiple terms. We only consider single terms in the consideration of complexity.

4. TERM ASSOCIATION PATTERN MINING FROM SEARCH LOGS

In this section, we first define two basic types of relationship between a pair of terms: syntagmatic and paradigmatic relation [17], which correspond to our contextual and translation models respectively. We then describe our term association mining approaches based on these two models. In the following, we use $c(x, X)$ to represent the count of x in collection X.

4.1 Contextual and Translation Models

4.1.1 Contextual Models

Our contextual models are designed to capture syntagmatic relations between terms. The syntagmatic relation is for those terms which frequently co-occur together. For example "rental" has a stronger syntagmatic relation with "car" than the word "basketball" since "rental" co-occurs with "car" more frequently in queries. In general, semantically related terms have stronger syntagmatic relation. This type of knowledge is very useful for query refinement. We first define term contexts.

DEFINITION 7 (TERM CONTEXTS). *Given a query collection Q and a term w, we have several different types of contexts for w.*

General Context G is a bag of words that co-occur with w in Q. That is, $a \in G \Leftrightarrow \exists q \in Q$, s.t. $a \in q$ and $w \in q$.

The i-th Left Context L_i is a bag of words that occur at the i-th position away from w on its left side in any $q \in Q$.

The i-th Right Context R_i is a bag of words that occur at the i-th position away from w on its right side in any $q \in Q$.

For example, given that a query "national car rental" appears in the query collection, "national" and "rental" are in the general context G of "car"; only "national" is in the L_1 and only "rental" is in the R_1 of "car". L_i and R_i are more precise contexts for each term. In the following, given a type of context C, we use $C(w)$ to represents w's C context.

Our contextual models are to capture the syntagmatic relations probabilistically. Given a term w, different terms have different strength of syntagmatic relation with w. We thus model this relation probabilistically and adopt language model approaches here: Given a word w and its context $C(w)$, the contextual model is a uni-gram language model. The Maximum Likelihood estimation (ML) is

$$P_C(a|w) = \frac{c(a, C(w))}{\sum_i c(i, C(w))}.$$

Intuitively, a context model tells us what words have high probabilities to appear around a given word w (or at a specific position).

Smoothing techniques are usually used for language models due to the data sparseness problem. An effective approach is Dirichlet prior smoothing [28]:

$$\tilde{P}_C(a|w) = \frac{c(a, C(w)) + \mu P(a|\theta_B)}{\sum_i c(i, C(w)) + \mu}$$

where $P(a|\theta_B)$ is a predefined reference model (usually set as the whole collection language model) and μ is the Dirichlet prior parameter to be set empirically (3000 in our experiments). Note that we use $\tilde{P}_C(\cdot|w)$ and $P_C(\cdot|w)$ to represent the smoothed and non-smoothed contextual models of w respectively.

4.1.2 Translation Models

Our translation models are designed to capture paradigmatic relations between terms. The paradigmatic relations capture words which are quasi-synonyms (e.g., "car" and "auto"). Our translation models are built on contextual models. The basic idea is that two terms have stronger paradigmatic relation if they have similar contexts. For example, "car" and "auto" may share a lot of contextual

words such as "sales" and "insurance" and thus have strong paradigmatic relation. This type of knowledge could be very useful in helping a user replace a query term (with a potentially better term) in a certain context.

In our translation model, we use $t(s|w)$ to denote the probability of "translating" w to the word s. In the language modeling approach, we use the Kullback-Leibler divergence (KL) $D(\cdot||\cdot)$ between two contextual models to measure the similarity between two contexts. Given two language models p and q, their KL-divergence is defined as

$$D(p||q) = \sum_u p(u) \log \frac{p(u)}{q(u)}.$$

The KL-divergence value is smaller if p and q are similar. We use KL-divergence on contextual models to define $t_C(s|w)$ as follows:

$$t_C(s|w) = \frac{\exp(-D[P_C(\cdot|s)||\tilde{P}_C(\cdot|w)])}{\sum_s \exp(-D[P_C(\cdot|s)||\tilde{P}_C(\cdot|w)])}.$$

After a few transformations, it can be seen that

$$t_C(s|w) \propto \prod_u \tilde{P}_C(u|w)^{c(u,C(s))}$$

which is the likelihood of generating s's context $C(s)$ from w's smoothed contextual model. The above formula can be applied on any type of contextual models. For example, we can use contexts G, L_1, or R_1. In this paper, we use a combination of L_1 and R_1 contexts since these two are most indicative of the word in consideration:

$$t(s|w) = \frac{|L_1(w)| \times t_{L_1}(s|w) + |R_1(w)| \times t_{R_1}(s|w)}{|L_1(w| + |R_1(w)|} \quad (1)$$

where $|L_1(w)|$ ($|R_1(w)|$) is the total number of terms occurring in the L_1 (R_1) context of w.

4.2 Mining Term Substitution Patterns

The context-sensitive term substitution patterns give us knowledge about query rewording. Recall that query rewording is to substitute a term w_i in q to be s and thus we get another query $q' = w_1...w_{i-1}sw_{i+1}...w_n$. The new query q' should have similar/related meaning to q. A good substitution should require that q' is better than q to retrieve more relevant documents. In other words, s is more appropriate than w_i to capture the information need given the context words in the query. For example, "car wash" is generally better than "auto wash" in the context "_wash".

4.2.1 Basic Approaches

In order to discover term substitution patterns, the probability we are interested in is: substituting the i-th position word w_i by s given the context $w_1...w_{i-1}\_w_{i+1}...w_n$: $P(s|w_i; w_1...w_{i-1}\_w_{i+1}...w_n)$. We use a shorthand $t(w_i \rightarrow s|q)$ to represent this probability. Then

$$\begin{array}{rcl} t(w_i \rightarrow s|q) & = & P(s|w_i; w_1...w_{i-1}\_w_{i+1}...w_n) \\ & \propto & P(w_i; w_1...w_{i-1}\_w_{i+1}...w_n|s)P(s) \\ & \propto & t(w_i|s)P(s)P(w_1...w_{i-1}\_w_{i+1}...w_n|s) \\ & = & t(s|w_i)P(w_1...w_{i-1}\_w_{i+1}...w_n|s) \end{array} \quad (2)$$

In Equation 2, each substitution candidate s is scored based on two factors: The first factor $t(s|w_i)$ reflects the similarity between the word w_i and the candidate s. This is a *global* factor which tells us how globally similar these two words are. The second factor $P(w_1...w_{i-1}\_w_{i+1}...w_n|s)$ is the *local* factor based on other context words in the query q. It tells us how likely s appears in such a context defined by q. These two factors are combined together to score each candidate in our method. To estimate the second factor, a simple approach is assume the context words are independent from each other given s. Using the general context G, we have

$$P(w_1...w_{i-1}\_w_{i+1}...w_n|s) = \prod_{j=1,j\neq i}^{n} \tilde{P}_G(w_j|s)$$

The general context ignores the position information and also considers all the words in context. In general, a word far away for the position in consideration should have lower impact. We thus use the more precise contextual models L_i and R_i and ignore those words which are far away:

$$\prod_{j=1,i-j>0}^{k} \tilde{P}_{L_{i-j}}(w_{i-j}|s) \times \prod_{j=1,i+j\leq n}^{k} \tilde{P}_{L_{i+j}}(w_{i+j}|s) \quad (3)$$

where k is the number of adjacent terms to consider. For example, if we set $k = 2$, we have $P(w_1...w_{i-1}\_w_{i+1}...w_n|s)$ as

$$\tilde{P}_{L_2}(w_{i-2}|s)\tilde{P}_{L_1}(w_{i-1}|s)\tilde{P}_{R_1}(w_{i+1}|s)\tilde{P}_{R_2}(w_{i+2}|s). \quad (4)$$

Note that we always use smoothed contextual models in the above formulas. To make the distributions at different position i comparable, we use n-th root of the value in Equation 3 and n is the total number of factors in the product.

4.2.2 Estimation Enhanced by User Sessions

For term substitutions, we need a reliable translation model $t(s|w)$. However, estimating $t(s|w)$ based only on contextual models need to be limited to ensure that s and w are semantically similar. For example, "American idol" and "American express" are two popular queries. Thus the same word "American" can show up in the L_1 contexts of "idol" and "express" frequently; as a result, we will have a high translation probability of t(express|idol), which is not desirable. To improve the translation models, we rely on the user sessions in our search logs (see an example in Table 1). Since queries in a user session are usually coherent, we would expect "idol" and "express" would not appear in the same sessions very often.

We use Mutual Information (MI) of the two words s and t over sessions to measure their correlation. MI is widely used to measure the mutual independency of two random variables in information theory, which intuitively measures how much information a random variable tells about the other. In our case, MI can be computed as follows:

$$I(s,w) = \sum_{X_s,X_w\in\{0,1\}} P(X_s, X_w) \log \frac{P(X_s, X_w)}{P(X_s)P(X_u)}. \quad (5)$$

where X_s and X_w are two binary random variables corresponding to the presence/absence of term s and term w in each user session. For example, $P(X_s = 1, X_w = 1)$ can be calculated as the proportion of the user sessions in which s and w are both present.

To make MI comparable across different pairs of words, we use a normalized version of MI in our paper, which is defined as

$$NMI(s,w) = \frac{I(s,w)}{I(w,w)} \quad (6)$$

It is easy to verify that $NMI(w,w) = 1$ and $0 \leq NMI(s,w) \leq 1$.

For our term substitution pattern mining, we combine $t(s|w_i)$ and $NMI(s,w_i)$ as follows:

(1) Given q and w_i, we use $t(s|w_i)$ to find the top N words which have the highest probabilities in $t(s|w_i)$.

(2) For each of these N words, we calculate its NMI with w_i using session information.

(3) We use a threshold τ to remove a word s from the N words if $NMI(s,w_i) \leq \tau$.

In our experiments, we set $N = 20$ and $\tau = 0.001$. Since all the remaining words have high translation probabilities and frequently co-occur with w in user sessions, they are more reliable and we thus set all $t(s|w_i) = 1$ for those remaining terms and compute $t(w_i \rightarrow s|q)$ only based on Equation 3. To decide when we need to replace w_i by s, we use $\frac{t(w_i \rightarrow s|q)}{t(w_i \rightarrow w_i|q)}$ as an indicator. For example, if this value is larger than 1, we would recommend to replace w_i by s.

A query may contain multiple words and any one of them can be potentially replaced/reworded. In an interactive manner, a user can tell the system which term he/she wants to replace. In an automatic manner, our general strategy is to iterate all the words and try to replace each of them. Then we get a set of candidates which differ from the original query by one term. Each of these candidates has a probability computed by Equation 3. We finally sort all these candidates by their corresponding probabilities and recommend the top ranked ones as substitutions.

4.3 Mining Term Addition Patterns

A term addition pattern $[+w|c_L\_c_R]$ is to add a word w given the context $c_L\_c_R$. Formally, given a query $q = w_1 w_2 ... w_n$ which contains n words and a position i, our task is to recommend a term r which can be added to the original query to form a new query $q' = w_1 ... w_{i-1} r w_i ... w_n$. We formalize this problem in a probabilistic way and we use $\gamma_i(r|q)$ to denote the probability of a pattern $[+r|w_1 ... w_{i-1}\_w_i ... w_n]$.

$$\begin{aligned} \gamma_i(r|q) &= P(r|w_1 ... w_{i-1}\_w_i ... w_n) \\ &\propto P(w_1 ... w_{i-1}\_w_i ... w_n|r)P(r) \end{aligned}$$

$P(w_1 ... w_{i-1}\_w_i ... w_n|r)$ can be estimated similarly to the estimation of the local factor in Equation 2. Here we estimate it similarly to Equation 3 as follows:

$$\prod_{j=1,i-j>0}^{k} \tilde{P}_{L_{i-j}}(w_{i-j}|r) \cdot \prod_{j=0,i+j\leq n}^{k-1} \tilde{P}_{L_{i+j}}(w_{i+j}|r) \qquad (7)$$

$P(r)$ is the prior probability of the appearance of the term r. In the simplest case, we can assume $P(r)$ to be uniform thus it will not affect the ranking of different terms.

Intuitively, the terms which have higher probabilities to be added to q are those which co-occur frequently in the query collection together with the words in q. Each word will be assigned a probability based on Equation 7 and we then rank all the terms.

Given query $q = w_1 ... w_n$, there are $n + 1$ positions in which we can add a term. In our experiments, we iterate over all these positions and get a list of new query candidates for position i. Each query has a corresponding probability estimated from $\gamma_i(r|q)$. A final list of the recommended queries is the ranked list merged from queries for all the positions.

5. DATA COLLECTION

We construct our data set based on the MSN search log data set released by the Microsoft Live Labs in 2006 [14]. Our log data spans 31 days from 05/01/2006 to 05/31/2006. In total, there are 8,144K queries, 3,441K distinct queries, 4,649K distinct URLs, and 7,470K user sessions in the raw data.

We separate the whole data set into two parts according to the time: the first 2/3 data is used to simulate the history data and it is used as a query collection to mine the term association patterns. The queries in the last 1/3 data are retained to test our methods. In the history collection, we clean the data by only keeping those well-formatted English queries (queries only containing characters from 'a' to 'z' and space). We also use a predefined stopword list to remove those common words such as "a" and "the" from our query collection. After cleaning, we get 4,431,152 queries in our query collection in total and 1,577,424 of them are distinct. The total number of unique words contained by the queries in this collection is 199,629 and the media length of the queries is 2. This data set is used in our experiments to compute the contextual models and translation models. For the user sessions, we obtain 3,540K in total and 1,320K of them have at least two queries in our training data. We use these 1,320K user sessions to compute the normalized mutual information between two terms.

Based the queries in our query collection, we build G, L_2, L_1, R_1, and R_2 contexts for the 76,693 most frequent words in the collection. All the contexts provide us the statistics of the necessary probabilities needed in our contextual models. Furthermore, we also compute the words which have high translation probabilities to each of the 76,693 words and thus build their translation models. All the contextual models and translation models are stored for online query rewording and query refinement.

6. EXPERIMENTS

In this section, we describe our experiments on mining term association patterns from the search engine logs. In all the following experiments, we set smoothing parameter $\mu = 3000$ and $k = 2$ in Equation 3.

6.1 Contextual and Translation Models

In Table 2, we show the G, L_1 and R_1 contextual models of two words: "car" and "yahoo". From this table, we can see that all the contextual models in this table appear to be meaningful. We can also see that G contexts mix L_1 and R_1 contexts and that L_1 and R_1 contexts are much different. This shows it is better to model these precise contexts for a given term in our query collection. Furthermore, these contextual words may cover different aspects. In Table 3, We show the results of the discovered aspects of "car" and "yahoo". The aspects are obtained by applying the star clustering [3] on the top words in the G contextual models (see [3] for more details) and we show the top 5 clusters. Clearly, for the word "car", people usually care about "rental" and "pricing". In the example of "yahoo", people are interested in "search", or "games". All these aspects correspond to different information needs of end users and thus can be potentially used for search result organization and query refinement.

We now show several examples of our translation models using Equation 1. In Table 4, we give 6 different terms

w=car						w=yahoo					
a	$P_G(a\|w)$	a	$P_{L_1}(a\|w)$	a	$P_{R_1}(a\|w)$	a	$P_G(a\|w)$	a	$P_{L_1}(a\|w)$	a	$P_{R_1}(a\|w)$
rental	0.152	rent	0.1187	rental	0.1937	mail	0.4806	sbc	0.5853	mail	0.5316
rent	0.046	rental	0.0892	rentals	0.0517	games	0.0672	verizon	0.0366	games	0.073
rentals	0.0318	national	0.0656	seat	0.044	maps	0.0459	mail	0.0327	maps	0.0506
enterprise	0.0306	classic	0.0451	audio	0.0403	finance	0.0402	launch	0.0274	finance	0.0444
national	0.0301	enterprise	0.0396	dealers	0.0312	music	0.0354	sign	0.0228	music	0.0384
prices	0.0271	race	0.0257	insurance	0.031	sbc	0.0331	email	0.015	personals	0.0259
audio	0.0246	budget	0.0237	wash	0.0275	personals	0.0234	chat	0.0124	email	0.0213
budget	0.0197	alamo	0.0235	max	0.0251	email	0.0206	download	0.0124	messenger	0.0157
insurance	0.0192	electric	0.0182	sales	0.0246	messenger	0.0157	games	0.0111	sports	0.0138
dealers	0.0191	hertz	0.0169	loan	0.0203	sports	0.0136	weather	0.0111	chat	0.0133

Table 2: Examples of the contextual models for "car" and "yahoo".

	w=car		w=yahoo
1.	buy, prices, values	1.	search, people, address
2.	rental, rent, alamo	2.	news, sports, photos
3.	audio, stereo, speakers	3.	online, games, word
4.	accidents, crashes	4.	videos, music, video
5.	loans, calculator, payment	5.	messenger, instant, im

Table 3: Aspects for words "car" and "yahoo".

Translation Model		Mutual Information	
s	$t(s\|w)$	s	$NMI(s,w)$
idol	0.0149626	idol	1
express	0.00305314	idols	0.00270233
airlines	0.00207636	top	0.000339295
inventor	0.00195964	medical	0.000206774
haunting	0.00194115	west	1.70E-04

Table 5: Translation model and Normalized Mutual Information of w = "idol".

from different domains and their translation models. For each term example, the top 5 words with highest translation probabilities are shown. We can seen that our proposed translation models are very effective to identify semantically similar words. For example, "fox", "abc", and "cnn" are all related to broadcast companies; "bmw", "honda", and "suzuki" are motorcycle/car brands. It is also interesting to note that words about different languages and words about different minerals can be identified to be similar. All these show the effectiveness of our proposed translation models to identify 'related words". All these terms provide possibility for users to do exploratory search.

6.2 Term Substitution Patterns

In this section, we study the effectiveness of our term substitution patterns. We first show several examples. We then compare our methods with previous methods to show that our method can improve the effectiveness of a query.

In this section, we enhance our translation models using user session information. Table 5 show an example. It can be seen that the words reranked using Normalized Mutual Information can indeed reduce those non-related words.

6.2.1 Examples of Substitution Patterns

We show several substitution patterns on query rewording. We decide to reword a query if the ratio $\frac{t(w\to s|q)}{t(w\to w|q)} > 1$, which means that s is more likely than w given query q. Table 6 shows several examples of the patterns (1st column) and the reworded queries by our method (2nd column). From the table, we have the following observations: (1) Our method can recommend more effective queries. For example, "kids games" is usually more effective than "children games". (2) Term substitution patterns are context-sensitive. For example, we substitute "auto" by "car" in the context "_ wash", while we substitute "car" by "auto" in the context of "_ trade". (3) We can see that queries from our methods are related to the original ones, but their meanings are not exactly equivalent (e.g., "birthday cards" and "greeting cards"). This is because translation models tend to find words with similar concepts but not always having the exactly same meanings, in general.

Pattern	**Reworded query**
auto→car |_ wash	car wash
car→auto |_ trade	auto trade
children→kids |_ games	kids games
kids→children |_ clothing	children clothing
driving→maps | google_	google maps
military→army |_ acu	army acu
birthday→greeting |_ cards	greeting cards
lotto→lottery | florida _ results	florida lottery results
interpretation→meanings | _ of dreams	meanings of dreams
music→song |_ lyrics	song lyrics

Table 6: Examples of term substitution patterns.

6.2.2 Effectiveness Comparison I

In this section, we study the effectiveness of query rewording by comparing with a previous method proposed in [12].

Experiment Design. In [12], related queries are generated according to user sessions. For each query, they first find all its next queries in all user sessions and use Log-Likelihood Ratio (LLR) to identify those highly related queries. Then they rerank the queries based on a model learned from training data. In our paper, we use their linear regression model for reranking and use LLR to denote this method. The LLR method gives a ranked list of queries. Some of the resulting queries is query rewording, but they also contain other types of query reformulation such as query refinement. To compare fairly, we filter the ranked list and only retain those queries which are rewording of the original queries.

To compare different methods, we construct our test cases from our hold-out logs as follows:

1) We merged all the sessions which have the same initial queries together as one test case.

2) For each test case, we use all the clicked URLs, *except* those of the initial query, in all the merged sessions

w=fox		w=bmw		w=computer		w=leg		w=chinese		w=calcium	
s	$t(s\|w)$	s	$t(s\|w)$	s	$t(s\|w)$	s	$t(s\|w)$	s	$t(s\|w)$	s	$t(s\|w)$
fox	0.0024	bmw	0.00195	computer	0.00155	leg	0.00571	chinese	0.0027	calcium	0.01034
cbs	0.00035	honda	0.00019	computers	0.00014	abdominal	0.00024	japanese	0.00011	sodium	0.00037
cnn	0.00034	suzuki	0.00017	laptop	0.00011	stomach	0.00024	korean	0.0001	potassium	0.00035
abc	0.00032	yamaha	0.00017	pc	0.00009	legs	0.00024	italian	0.00009	magnesium	0.0003
bbc	0.0003	triumph	0.00017	notebook	0.00009	muscle	0.00019	greek	0.00009	cholesterol	0.00023

Table 4: Examples of translation models of 6 different terms.

to approximate relevance documents. These documents are to approximate relevant documents which users obtained through query reformulation. Our purpose is to compare different methods with respect to fetching additional relevant documents.

Given the test cases constructed above, we compare 3 methods: The first method (denoted by Original query) is to use the initial/original queries to get a ranked list from a search engine. The second method is the LLR method in [12] and the third is ours. For either of these two methods, we first generate a list of recommended queries. We then fetch a ranked list of search results for each of the queries from the same search engine. Finally we use our relevance judgement to evaluate these different search results.

Our goal is to test which method can recommend more effective queries. Since both our and LLR methods can not generate recommended queries for every query, we thus filter the test cases and only retain those for which both LLR and our method can generate at least 5 queries and for which the first generated queries by our method satisfy $\frac{t(w\to s|q)}{t(w\to w|q)} > 1$. This is to simulate the scenario in which the first query is not very effective since it is precisely in such a scenario that a user would need help with reformulation. From all the remaining test cases after filtering, we randomly sample 50 test cases for our evaluation.

We use Precision@5 (P@5) as our evaluation metric. Given m recommended queries for each test case, we select the best one which has the largest number of relevant documents in its top 100 search results and use its P@5 as the accuracy of the corresponding test case.

Results. We vary the number of recommended queries m from 1 to 5 and the best P@5 values are shown in Figure 1. From this figure, we can see that both our method and LLR outperform original query and thus can recommend more meaningful queries. Compared with LLR method, our method is more effective. For example, when we only consider the first recommended query, our method can give P@5 = 0.08 while LLR method can only give P@5 = 0.05. This is because LLR method does not consider the effectiveness of a query while our method would recommend a more effective one based on the term substitution patterns mined from search logs. For example, our method can recommend "cheap tickets" for "cheap airfare", while LLR only suggests queries such as "discount airfare", which is not as effective as "cheap tickets".

6.2.3 *Effectiveness Comparison II*

In this section, we study the effectiveness of our method for queries with the same meanings. In order to ensure that the recommended queries have the same meanings as the original ones, we propose a lexical matching constraint for the translation pairs.

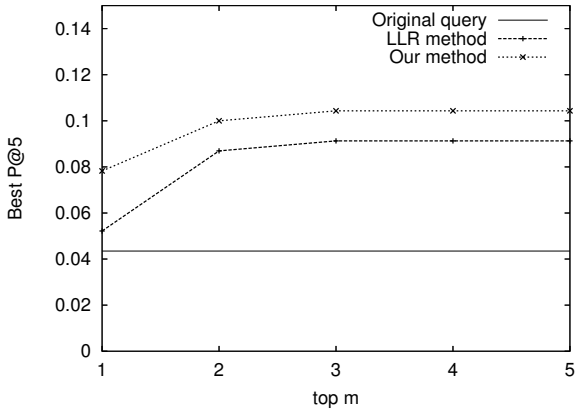

Figure 1: Comparison of term substitution patterns. We compare the best P@5 of the top m recommended queries by different methods.

translation pairs					
plural/singular		**abbreviation**		**others**	
map	maps	dept	department	hair	hairstyles
page	pages	tx	texas	space	myspace
code	codes	tv	television	pics	pictures
number	numbers	co	company	fish	fishing
loan	loans	st	saint	air	airlines

Table 7: The categories and examples of translation pair after applying lexical matching constraint.

Lexical Matching Constraint. Given a word w, we first get its top 3 words with the highest probabilities based on its translation model. This gives us 3 translation pairs. Our lexical matching constraint only retains those pairs such that every character in the short word of the pair must appear in the long word, in the same order. For example, "tx" and "texas" are a qualified pair since "t" and "x" both appear in "texas" and "t" is before "x" in both words. By applying this lexical matching constraint, we can hopefully get semantically equivalent pairs.

Table 7 shows several examples of the identified pairs after applying our lexical matching constraint. We found that the results can be classified into three categories: plural/singular, abbreviations, and others.

Using the translation pairs filtered by our lexical matching constraint as candidate pairs, we apply our query rewording algorithm on a set of queries sampled from our test data. Each of these queries q contains at least a word w from our candidate pairs and our rewording algorithm tries to replace w with its paired word s. If the ratio $\frac{t(w\to s|q)}{t(w\to w|q)} > 1$, we reword q by replacing w by s. Finally, we rewrite 1,437 queries.

Original query	New query	Ratio
maps quest	map quest	358.571
sams clubs	sams club	264.500
white page	white pages	149.027
six flag	six flags	39.2353
aol email	aol mail	31.7024
continental air	continental airlines	21.2667
hair pics	hair pictures	20.0000
lotto tx	lotto texas	16.4815
window media	windows media	14.4026
yahoo map	yahoo maps	7.96296

Table 8: Examples of context-sensitive query rewording using the pairs filtered by our lexical matching constraint.

Table 8 shows several examples of our query rewording, ordered by their ratios. In this table, we can see that some queries are changed from singular to plural form, while some queries are changed from plural to singular form. All the changes are context sensitive. For example, our algorithm changes "yahoo map" to "yahoo maps" (from singular to plural), but changes "maps quest" to "map quest" (from plural to singular). Intuitively, our reworded queries are more effective since the domain names of these two queries are maps.yahoo.com and mapquest.com. A recent work [15] has reach similar conclusion that context-sensitive stemming can improve click-through rate. Our results are consistent with this conclusion.

Effectiveness Experiment Design. To study the effectiveness of query rewording, we use the clicked web pages in our search engine log data to evaluate. Given a query, we collect all the positions of its clicked documents in our test log data and aggregate all these clicks together. We treat all the clicked positions as the positions of relevant documents in a ranking list and evaluate and compare the accuracy of the original queries and our recommended queries. Since we use our lexical constraint to force all the pairs to share equivalent meaning, comparing their results to show their effectiveness is reasonable. Intuitively, a better query would retrieve more relevant documents on the top of the ranked list that users tend to click. In our experiments, for each query, we calculate the precision at 1, 5, 10, 15, and 20 documents.

Effectiveness Results. Figure 2 shows the comparison between the original queries and the reworded queries. The precisions are averaged over all the queries that our algorithm decides to rewrite. Clearly, we can see that, at every level of precision, our recommended queries can retrieve more relevant documents and thus outperform the original queries. For example, the P@10 of our recommended queries is 0.42, while the P@10 of the original queries is about 0.28. We achieve 41.3% relative improvement.

The ratio $\frac{t(w \to s|q)}{t(w \to w|q)}$ can be regarded as a confidence score of our query rewording. A high ratio means that the reworded query is more appropriate than the original query. To test this, we ordered the query pairs by the ratio in decreasing order and we then evaluate the accuracy of top m pairs by varying m from 100 to 600. Figure 3 shows the influence of ratio and the difference is measured by the quotient of P@10 of recommended queries over original ones. From this figure, we can see that when the ratio is higher, the performance difference is larger. This means that the ratio is a good indicator of the confidence of query rewording.

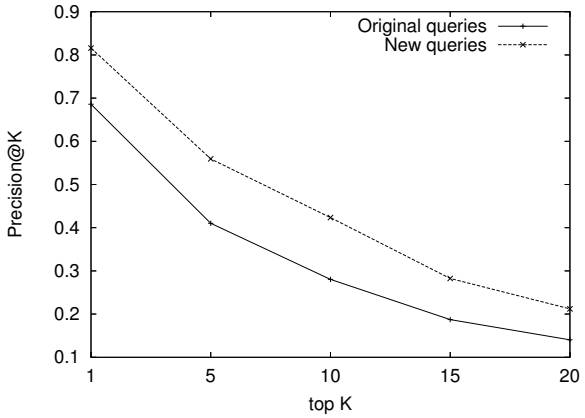

Figure 2: The overall performance comparison of the original and rewritten queries. We compare their results by Precision@K.

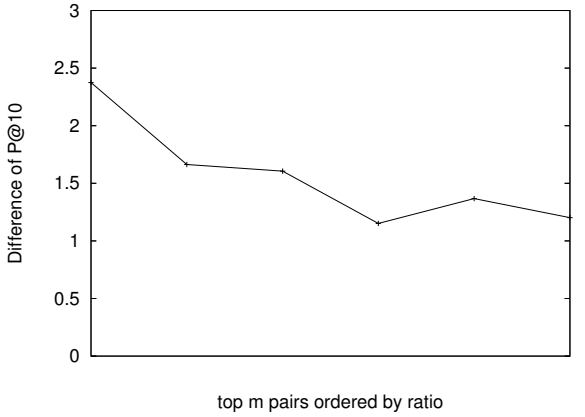

Figure 3: The impact of the ratio on the performance. The Difference of P@10 is measured as the quotient of P@10 of recommended queries over original ones.

6.3 Term Addition Patterns

In this section, we study our context-sensitive term addition patterns and use them for query refinement.

6.3.1 Examples of Addition Patterns

Table 9 shows several examples of the mined term addition patterns and the refined queries based on Equation (7). For each query, all the patterns are ordered by their probabilities in decreasing order. These examples show that our method can recommend very meaningful terms to refine an original query. For example, for the query "wedding", we can recommend meaningful terms related to different aspects of "wedding", such as "dresses" and "cakes". All these terms can help users refine their queries and thus find more coherent results. Furthermore, all these terms give good guidance for a user when he/she wants to prepare a "wedding". Such a recommendation is more useful if a user is not satisfied

q ="song lyrics"		q ="baby names"		q ="wedding"	
pattern	refined query	pattern	refined query	pattern	refined query
+christian|_song lyrics	christian song lyrics	+boy|baby_names	baby boy names	+dresses|wedding_	wedding dresses
+country|_song lyrics	country song lyrics	+girl|baby_names	baby girl names	+cakes|wedding_	wedding cakes
+gospel|_song lyrics	gospel song lyrics	+popular|_baby names	popular baby names	+invitations|wedding_	wedding invitations
+love|__song lyrics	love song lyrics	+meanings|baby names_	baby names meanings	+songs|wedding_	wedding songs
+spanish|_song lyrics	spanish song lyrics	+girl|_baby names	girl baby names	+favors|wedding_	wedding favors
+search|song lyrics_	song lyrics search	+unusual|_baby names	unusual baby names	+flowers|wedding_	wedding flowers
+worship|__song lyrics	worship song lyrics	+unique|_baby names	unique baby names	+gowns|wedding__	wedding gowns
+search|song_lyrics	song search lyrics	+irish|_baby names	irish baby names	+rings|wedding_	wedding rings
+praise|_song lyrics	praise song lyrics	+italian|_baby names	italian baby names	+toasts|wedding__	wedding toasts
+search|_song lyrics	search song lyrics	+twins|baby names_	baby names twins	+vows|wedding__	wedding vows

Table 9: Examples of term addition patterns. All the patterns are ordered according to their probabilities in decreasing order.

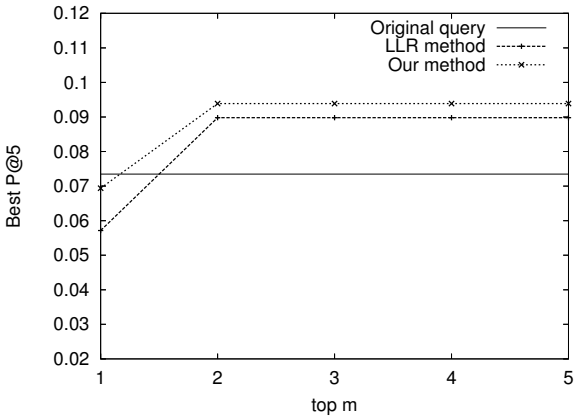

Figure 4: Comparison with the LLR method of term addition patterns.

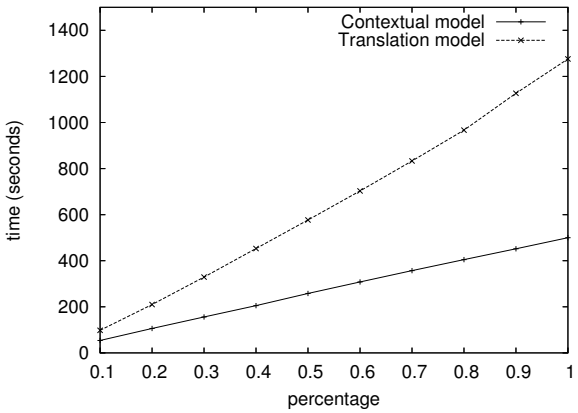

Figure 5: The time complexity of building the contextual models and translation models.

with the current results but lacks the necessary knowledge to think of effective words to refine his/her query.

6.3.2 *Effectiveness Comparison*

We compare our method with LLR method in a similar way as in Section 6.2. We use the same test set and construct our test cases similarly. The only difference is that we only retain those recommended queries which are refinements of the original queries for the LLR method. We also use 50 test cases in this experiments and use $P@5$ as the major evaluation metric. Figure 4 shows the comparison between different methods. In this figure, we have similar observations to the term substitution patterns: Both our and LLR methods can outperform the baseline method. Compared with LLR method, our method can achieve better results. This is because LLR method only consider the query refinement within user sessions. Our method can utilize information across sessions since our contextual models are built over the whole collection.

6.4 Implementation and Efficiency

The efficiency of the algorithm is quite important since a query collection is huge. We test the efficiency of our method in this section.

We implement our algorithm using the Lemur toolkit[1]. The original data is a collection of queries. We build the standard Lemur index by treating each query as a document. This part is as efficient as the standard document indexing, thus it can be applied to very large data set pretty easily. The basic knowledge we discovered from the query collection includes the contextual models and translation models. Both are processed offline based on the Lemur index of the query collection. Based on the translation models and contextual models, the context-sensitive term substitution and term addition patterns are discovered in an online manner. Once the knowledge is built, the online part needs only fetch the corresponding knowledge we stored, and thus can be quite efficient. Since all the online parts are very efficient, in the following, we only test the efficiency of the offline part which is to compute the contextual models and translation models.

To study the efficiency and scalability, we randomly sample $f\%$ of the original queries from the whole query collection. We vary f from 10 to 100 with step 10. For each value of f, we record the time needed for our offline part. Figure 5 shows the time complexity of our algorithms. In this figure, x-axis is the value of $f\%$ and y-axis is the time. It can be seen that both lines are roughly linear and thus our offline part is linearly scalable. This shows that our proposed methods can mine those patterns very efficiently and can be applicable to very large query collections.

[1]http://www.lemurproject.org/

7. CONCLUSIONS AND FUTURE WORK

In the paper, we studied the problem of mining term association patterns from the vast amount of search engine log data. We defined two novel term association patterns (i.e., context-sensitive term substitution and term addition

patterns) and proposed new methods for mining such patterns from search engine logs. Our methods are based on the contextual and translation models which are mined from a query collection. The two types of discovered term association patterns can be used to address the mis-specification and under-specification problems of ineffective queries. Experiment results on search engine logs show the effectiveness of our proposed methods.

There are a few limitations of our work. First, all the experiments are based on clickthroughs instead of real relevance judgments, so an interesting future work would be to further test the proposed methods with real relevance judgments. Second, building an interactive user interface which can allow a user to modify his/her queries using our suggested terms can help evaluate our algorithms. Third, search logs have more meaningful click-through information besides queries and sessions. We can extend our pattern mining algorithms to incorporate this click-through information in the future.

8. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable suggestions. This work is in part supported by the National Science Foundation under award numbers IIS-0347933, IIS-0713581, and a gift grant from Microsoft Research.

9. REFERENCES

[1] E. Agichtein, E. Brill, and S. T. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR*, pages 19–26, 2006.

[2] P. Anick. Using terminological feedback for web search refinement: a log-based study. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 88–95, 2003.

[3] J. A. Aslam, E. Pelekov, and D. Rus. The star clustering algorithm for static and dynamic information organization. *Journal of Graph Algorithms and Applicatins*, 8(1):95–129, 2004.

[4] D. Beeferman and A. L. Berger. Agglomerative clustering of a search engine query log. In *KDD*, pages 407–416, 2000.

[5] P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer. The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist.*, 19(2):263–311, 1993.

[6] S. Chien and N. Immorlica. Semantic similarity between search engine queries using temporal correlation. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 2–11, 2005.

[7] S. Cucerzan and E. Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *EMNLP*, pages 293–300, 2004.

[8] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic query expansion using query logs. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 325–332, 2002.

[9] R. Green. Syntagmatic relationships in index languages: A reassessment. *Library Quarterly*, 65(4):365–385, 1995.

[10] D. Inkpen and G. Hirst. Building and using a lexical knowledge-base of near-synonym differences. *Computational Linguistics*, 32(2):223–262, June 2006.

[11] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142, 2002.

[12] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW*, pages 387–396, 2006.

[13] D. Lin. Automatic retrieval and clustering of similar words. In *COLING-ACL*, pages 768–774, 1998.

[14] Microsoft Live Labs. Accelerating search in academic research, 2006. http://research.microsoft.com/ur/us/fundingopps/RFPs/Search_2006_RFP.aspx.

[15] F. Peng, N. Ahmed, X. Li, and Y. Lu. Context sensitive stemming for web search. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 639–646, 2007.

[16] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *KDD*, pages 239–248, 2005.

[17] R. Rapp. The computation of word associations: comparing syntagmatic and paradigmatic approaches. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[18] J. J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval SystemExperiments in Automatic Document Processing*, pages 313–323, 1971.

[19] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *WWW*, pages 377–386, 2006.

[20] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[21] D. Shen, M. Qin, W. Chen, Q. Yang, and Z. Chen. Mining web query hierarchies from clickthrough data. In *AAAI*, pages 341–346, 2007.

[22] X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *SIGIR*, pages 43–50, 2005.

[23] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *SIGMOD*, pages 131–142, 2004.

[24] X. Wang, H. Fang, and C. Zhai. A study of methods for negative relevance feedback. In *SIGIR*, pages 219–226, 2008.

[25] X. Wang and C. Zhai. Learn from web search logs to organize search results. In *SIGIR*, pages 87–94, 2007.

[26] J.-R. Wen, J.-Y. Nie, and H. Zhang. Clustering user queries of a search engine. In *WWW*, pages 162–168, 2001.

[27] J. Xu and W. B. Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM Trans. Inf. Syst.*, 18(1):79–112, 2000.

[28] C. Zhai and J. D. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR*, pages 334–342, 2001.