Using Query Log and Social Tagging to Refine Queries Based on Latent Topics*

Lidong Bing Wai Lam
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
Shatin, Hong Kong
{ldbing, wlam}@se.cuhk.edu.hk

Tak-Lam Wong
Department of Mathematics and Information Technology
The Hong Kong Institute of Education
Tai Po, Hong Kong
tlwong@ied.edu.hk

ABSTRACT

An important way to improve users' satisfaction in Web search is to assist them to issue more effective queries. One such approach is query refinement (reformulation), which generates new queries according to the current query issued by users. A common procedure for conducting refinement is to generate some candidate queries first, and then a scoring method is designed to assess the quality of these candidates. Currently, most of the existing methods are context based. They rely heavily on the context relation of terms in the historical queries, and cannot detect and maintain the semantic consistency of queries. In this paper, we propose a graphical model to score queries. The proposed model exploits a latent topic space, which is automatically derived from the query log, to assess the semantic dependency of terms in a query. In the graphical model, both term context dependency and topic context dependency are considered. This also makes it feasible to score some queries which do not have much available historical term context information. We also utilize social tagging data in the candidate query generation process. Based on the observation that different users may tag the same resource with different tags of similar meaning, we propose a method to mine these term pairs for new candidate query construction.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query Formulation, Search Process*

* The work described in this paper is substantially supported by grants from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Codes: CUHK4128/07 and CUHK413510) and the Direct Grant of the Faculty of Engineering, CUHK (Project Codes: 2050442 and 2050476). This work is also affiliated with the Microsoft-CUHK Joint Laboratory for Human-centric Computing and Interface Technologies.
We also acknowledge the suggestions received from Xuanhui Wang on the baseline method implementation.

General Terms

Algorithms, Experimentation

Keywords

Query Refinement, Social Tagging, Query Log Mining

1. INTRODUCTION

Extensive research works have been conducted to improve the performance of search engines by exploiting search logs [1, 11, 21, 29]. One important area is to help users issue more effective queries. There are two major directions in this research, namely, query suggestion (recommendation) and query refinement (reformulation). In query suggestion, historical queries are recommended to users according to a certain similarity measure with the current query issued by users. This suggestion can be performed based on the click graph [13, 26] or query flow graph [5, 6]. However, it cannot provide new queries which are not contained in the query log. On the other hand, in query refinement, the query log is used in a more delicate manner that the current query is refined by exploiting the term dependency information in the historical queries. The generated query may be an existing query in the query log, or a totally new query. According to the operations performed, there are three major types of refinement, namely, substitution [22, 32], expansion [3, 11, 18], and deletion [24, 36]. A common procedure of carrying out refinement is to generate some candidate queries first, and then a scoring method is used to assess the quality of these candidates.

A contextual model was proposed by investigating the context similarity of terms in historical queries [32]. Two terms in a pair with similar contexts are used to substitute each other in new query generation. Then, a context based translation model is employed to score the generated queries. Jones et al. employed hypothesis likelihood ratio to identify those highly related query phrase or term pairs in all user sessions [22]. Then, these phrase pairs are utilized to generate new queries for the input query. All the above methods make use of query log, and exploit the context information to generate term pairs and score new queries. For example, Table 1 shows top 15 output queries of an existing context based method, which is used as the comparison baseline in our experiments, for the original query "wrestling ring instructions". We can observe that the context based method has two shortcomings. First, the context based candidate term generation process is easily affected by noise when deal-

Table 1: Top 15 results given by a context based method for original query "wrestling ring instructions". "◇" represents the unchanged original terms.

1	◇ rings ◇	6	◇ ◇ poncho	11	◇ ◇ steps
2	championship ◇ ◇	7	◇ ◇ stitches	12	◇ ◇ instruction
3	◇ ◇ instrutions	8	◇ ◇ scarf	13	◇ ◇ manual
4	◇ ◇ instuctions	9	◇ ◇ loom	14	championships ◇ ◇
5	◇ ◇ afghans	10	◇ ◇ afghan	15	◇ blaylock ◇

ing with ambiguous terms. For example, considering the term "instructions", due to the fact that "instructions" has very diverse contexts in historical queries, a lot of noise is involved in its candidate list such as "afghans", "poncho", etc. Furthermore, typos such as "instrutions" and "instuctions" also degrade the performance when dealing with the typo-prone terms. Second, context based method cannot detect and maintain consistency of semantic meaning when scoring a new query. For example, as shown in Table 1, a good output query is "wrestling ring manual", but it is ranked as the 13th. The reason is that the scoring function highly depends on the co-occurrence of the terms in the contexts of historical queries. "championship ring" is quite popular in query log owing to NBA. Consequently, "championship ring instructions" is assigned a high score although the semantic meaning of the terms in this refined query is not consistent.

Furthermore, when an unfamiliar or rare query is issued by a user, existing context based methods cannot work well too. The definition of unfamiliar query is given as follows:

DEFINITION 1 (UNFAMILIAR QUERY). *Let $(t_1, \cdots, t_n)$ denote a particular query q with length greater than 1, i.e. $n > 1$. Let $(t_i, \cdots, t_j)$ denote a sub-sequence of q, where $1 \leq i < j \leq n$. If none of $(t_i, \cdots, t_j)$s has ever been observed before as a whole query or a sub-sequence of other queries, q is regarded as an unfamiliar query.*

The main reason for ineffective handling of unfamiliar queries by existing context based methods is that these methods mine the historical queries to obtain hints for refining the issued query. When there is no or very little related information, the performance will inevitably be affected. First, it is not easy to figure out the candidate terms to refine the query. Second, the statistical information may not be reliable to differentiate the quality of different candidate queries. One may expect that as the amount of available query log increases, the number of unfamiliar queries will decrease. We conducted an investigation on the percentage of unfamiliar queries in an existing query log used in our experiments and found out that the situation is not as optimistic as what we expect. This query log consists of queries issued to a search engine for a period of 3 months. We started with the queries of the first 2 months as historical queries on hand. At the beginning, the queries in the first day of the third month were

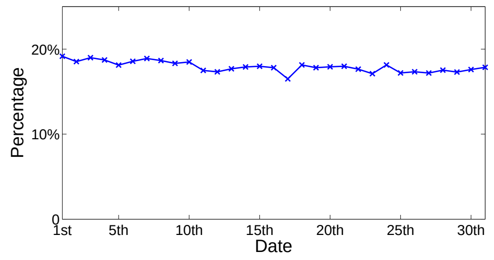

Figure 1: Percentage of unfamiliar queries.

Table 2: Top 30 tags of "http://www.dvguru.com/". The tag frequency is given on the right of each tag.

video	237	news	26	daily	13
dv	156	tech	22	blogging	13
blog	145	howto	19	audio	12
filmmaking	113	editing	17	reviews	10
digital	87	tools	16	movie	10
film	79	tips	16	digitalvideo	10
blogs	50	tv	15	entertainment	9
technology	41	movies	15	business	8
tutorial	29	software	14	cinema	7
tutorials	27	vlog	13	advice	7

used to simulate the newly issued queries, and we calculated the percentage of unfamiliar queries in that day. Then, these queries were added into historical query set so that they were treated as historical queries when we proceeded to the second day of the third month. The same statistical procedure was repeated for all the subsequent days. The statistical information is shown in Figure 1. We can see that the percentage of unfamiliar queries does not change significantly when the amount of historical queries accumulate. Thus, it shows that a certain percentage of unfamiliar queries will always be issued by users. Consequently, it poses another challenge in generating candidate terms for those unfamiliar queries due to the lack of context information. Moreover, it also causes difficulty in scoring of new queries.

In summary, the context based methods have limitations in both major tasks, namely, candidate generation and query scoring. In candidate generation, noise will be involved as exemplified in Table 1 for the terms with diverse meaning. The candidates cannot be generated properly when lacking context information for some terms. In query scoring, the context based methods cannot detect and maintain the semantic consistency of the terms in the query. Furthermore, if the query is not familiar, the scoring task becomes even harder. To tackle the problems in the first task, we propose a method to generate better candidate term pairs by considering useful information from social tagging data. Social tagging is an important application and a valuable semantic repository, and its potential in enhancing Web search performance has been studied in personalized search [7] and query suggestion [16]. Currently, millions of users are involved in some famous social tagging systems such as Delicious[1] and Flickr[2], and they tag and share their favorite online resources. For example, in Delicious data, the tag set of URL "http://www.dvguru.com/", which is a Web site for users to share their experience on digital videos, is given in Table 2. From this tag set, term pairs with similar meaning such as "film" and "movie", "tips" and "tutorial" can be easily found. This is due to the fact that different users may choose their own preferred tags to express the same concept. Therefore, if two tags co-occur frequently in different resources' tag sets, they are very likely to have closely related meaning. Moreover, users regularly maintain their own tagging data and correct some typos and inaccurate tags, so social tagging data contains less noise than query log.

To tackle the problems in the second task, we propose a graphical model which can score a query taking into account the semantic consistency of the terms in the query as well as the term context. First we discover the latent topics in the

[1]`http://www.delicious.com/`
[2]`http://www.flickr.com`

query log, and these topics are utilized to construct a latent topic sequences of the terms in a query. When we score a query with the proposed graphical model, the latent topic sequence of the query is used as hidden evidence to guide the semantic dependency assessment.

2. OVERVIEW OF OUR FRAMEWORK

As mentioned above, there are two major tasks to refine a query. The first task is to generate some candidate queries for the original query via operations such as substitution. The second task is to score the generated candidate queries, and rank them according to the predicted quality.

In the first task, we focus on query term substitution. We first determine a set of possible substitution terms for each term in the original query. These terms are utilized to substitute one or more original terms in the original query. As discussed in Section 1, the noise and the lack of sufficient context information are the major problems when using query log. To tackle these problems, we develop a method to generate better candidate term pairs by considering useful information from social tagging data. Based on the observation that terms with similar meaning are often used to tag the same resource by different users, we can extract some candidate substitution pairs. Then a filtering method is proposed with a delicately designed similarity function. This method is capable of filtering out different types of noisy pairs such as "north" and "carolina", "free" and "music".

In the second step, we propose a graphical model to score the quality of the candidate queries generated in the first step. In the proposed graphical model, the semantic consistency of the queries is considered. Two groups of variables are designed in the graphical model. One group corresponds to the terms, and the other group corresponds to the concepts of the terms. Unlike existing context based methods, we examine the query quality taking into account the semantic consistency of the consecutive terms in a latent concept space. We first derive a set of pseudo-documents from the query log data. Then these pseudo-documents are used to generate a latent concept space. The graphical model can capture the concept dependency between consecutive terms. Therefore, even when the term context information is not sufficient in the query log, it can still give reliable score by considering the semantic dependency of the consecutive terms in the latent concept space.

3. CANDIDATE TERM GENERATION

To generate candidate terms from social tagging data for query refinement, we make use of a bookmark data, namely Delicious, to illustrate our method and test the performance in the experiment. Note that the proposed method is applicable to other types of social tagging data. In this component, we first develop a method for mining term pairs from the bookmark data. After that, we consider the pseudo-context of the candidates to filter out noise.

3.1 Candidate Term Extraction

A piece of bookmark, denoted by $\mathcal{B}_{lu} = \{t_1, t_2, \cdots\}$, is a set of tags which are given by a user u to summarize the content of a URL l, such as {film, tutorial, blog} and {movie, technology, tips}. Our candidate term extraction method is based on the observation that different users may use tags with similar meaning to describe the same resource. To capture this kind of co-occurrence, we aggregate all tags from different users of the same URL together, and compose a set of distinct tags for each URL, which is denoted by $\mathcal{T}_l$.

The mutual information of two tags, t_1 and t_2, can be calculated as in Equation 1:

$$I(t_1, t_2) = \sum_{X_{t_1}, X_{t_2} \in \{0,1\}} P(X_{t_1}, X_{t_2}) \log \frac{P(X_{t_1}, X_{t_2})}{P(X_{t_1})P(X_{t_2})}, \quad (1)$$

where X_{t_i} is a binary random variable indicating whether term t_i appears in a particular tag set $\mathcal{T}_l$. For example, $P(X_{t_1} = 1, X_{t_2} = 1)$ is the proportion of the tag sets which contain t_1 and t_2 simultaneously. Because commonly used terms may be used to tag a lot of URLs such as "free" and "good". To cope with this problem, we adopt the normalized form of mutual information:

$$NMI(t_1, t_2) = \frac{I(t_1, t_2)}{\big[H(t_1) + H(t_2)\big]/2}, \quad (2)$$

where $H(t_i)$ is the entropy of t_i's distribution among all the tag sets, and calculated as in Equation 3:

$$H(t_i) = -\sum_{X_{t_i} \in \{0,1\}} P(X_{t_i}) \log P(X_{t_i}). \quad (3)$$

If $NMI(t_1, t_2)$ is greater than a threshold, the term pair t_1 and t_2 will be stored for the next stage of term filtering.

3.2 Candidate Term Filtering

The term pairs generated in the previous step may involve noise. The first kind of noise is mainly caused by noun phrases. For example, "north" is found as the top 1 candidate for "carolina", and "united" is top 1 for "states". These are obviously not good candidates for term refinement. The main reason is that the phrases, i.e. "north carolina" and "united states", are used by many users to tag URLs. Thus, the NMI values of two terms in the same phrase are large. Another kind of noise is caused by the frequently used terms. For example, "free" is found as one of the candidates for many terms such as "music", "movie", "film", etc. By manually examining the data, we find that users maintain a large amount of bookmarks for these free resources on the Web. We propose a method to filter out this undesirable candidate terms.

For each tag t, we first construct its pseudo-context in the bookmark environment, denoted by $C(t)$, which is composed of all tags that co-occur with t in at least one $\mathcal{B}_{lu}$. $cnt\big(t_i|C(t)\big)$ denotes the frequency of t_i in the pseudo-context of t. It is equivalent to the co-occurrence frequency of t and t_i in all $\mathcal{B}_{lu}$s. Then the pseudo-context document of tag t, $\mathcal{D}_t$, is defined as:

$$\mathcal{D}_t \triangleq (w_1^t, w_2^t, \cdots, w_T^t), \quad (4)$$

where w_i^t is t_i's weight in $\mathcal{D}_t$, which can be computed as:

$$w_i^t = \frac{cnt\big(t_i|C(t)\big)}{\sum_{t_j} cnt\big(t_j|C(t)\big)} \times \log \frac{|\mathbb{D}_T|}{|\{\mathcal{D}_{t_k}|t_i \in \mathcal{D}_{t_k}\}|}, \quad (5)$$

where $\mathbb{D}_T$ denotes the set of all pseudo-context documents, and T is the vocabulary of the bookmark data. Then, we calculate a refined cosine similarity between a pair of pseudo-context documents:

$$sim(\mathcal{D}_{t_i}, \mathcal{D}_{t_j}) = \frac{\sum_{k=1}^{|T|}(w_k^{t_i} \times w_k^{t_j})\big(1 - \gamma(t_i, t_j|t_k)\big)}{|\mathcal{D}_{t_i}||\mathcal{D}_{t_j}|}, \quad (6)$$

where $\gamma(t_i, t_j|t_k)$ is the relation factor between term t_i and t_j given t_k, which is calculated as:

$$\gamma(t_i, t_j|t_k) = \frac{cnt\big(t_i, t_j|C(t_k)\big)}{\min\big\{cnt\big(t_i|C(t_k)\big), cnt\big(t_j|C(t_k)\big)\big\}}, \qquad (7)$$

where $cnt\big(t_i, t_j|C(t_k)\big)$ is the number of $\mathcal{B}_{lu}$s that contain t_i, t_j and t_k simultaneously. Therefore, the terms which co-occur with "north carolina" frequently will contribute less when calculating the similarity of $\mathcal{D}_{carolina}$ and $\mathcal{D}_{north}$. The design of $\gamma(t_i, t_j|t_k)$ is based on the following observation: one URL is seldom tagged by the same user with two terms that have similar meaning such as "film" and "movie", "car" and "automobile". According to Equation 7, the value of γ("car", "automobile"|"repair") tends to be small, while, the value of γ("carolina", "north"|"lottery") tends to be large. Consequently, the contribution of the term "repair" to the similarity between $\mathcal{D}_{car}$ and $\mathcal{D}_{automobile}$ is increased, and the contribution of the term "lottery" to the similarity between $\mathcal{D}_{carolina}$ and $\mathcal{D}_{north}$ is decreased. The refined cosine similarity can also filter out the terms which are frequently used such as "free" and "howto", because their contexts are quite diverse, and cannot have a high similarity with other specific terms' contexts.

4. GRAPHICAL MODEL BASED SCORING

In this section, we first present the graphical model and the scoring method. Then we describe the details of the model, namely, latent topic discovery from the query log, and model parameter training.

4.1 The Graphical Model

A query q is composed of a sequence of terms, which can be denoted by $q : \ t^1 \cdots \ t^n$, where the superscript indicates the position of the term, and $t^r (1 \le r \le n)$ may take any term in the vocabulary. We propose a graphical model which considers the dependency among latent semantic topics. Figure 2 depicts the design of the graphical model. The query terms, i.e. t^r, are observable and represented by filled nodes, and the latent topics, denoted by z^r, of the corresponding terms are unobservable and represented by empty nodes. The discovery of latent topics will be discussed later.

The transition probability, denoted by $P(z^{r+1}|z^r)$, between topics is considered, where z^r and z^{r+1} are the semantic topics of t^r and t^{r+1} respectively. We also make use of the probability of obtaining the term t^{r+1} given t^r and z^{r+1}, i.e. $P(t^{r+1}|z^{r+1}, t^r)$. Hence, in this model, each term depends on its latent topic and the preceding term, and each topic depends on its preceding topic. When we score a query with this graphical model, the latent topic sequence of the query is used as hidden evidence to guide the semantic consistency assessment. For queries with good semantic consistency, their corresponding topic sequences will increase the score of their quality. For these queries, the transition probability between the consecutive topics in the topic sequence is higher.

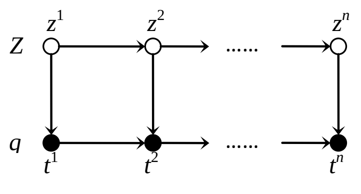

Figure 2: The graphical model for scoring a query.

When we employ this graphical model to score a candidate query, the score of a query q can be denoted as $P(t^1 \cdots t^n)$, which is given in Equation 8:

$$P(t^{1:n}) = \sum_{z^r = z_1, 1 \le r \le n}^{z_N} P(t^{1:n}, z^{1:n}). \qquad (8)$$

We marginalize over all possible latent semantic topic sequences to calculate the joint probability of generating the term in query q. Based on the structure of the designed graphical model, $P(t^{1:n}, z^{1:n})$ can be calculated as in Equation 9 by the chain rule:

$$P(t^{1:n}, z^{1:n}) = \prod_{1 \le r \le n} P(z^r|z^{r-1}) P(t^r|z^r, t^{r-1}). \qquad (9)$$

When $r = 1$, $P(z^1|z^0) = P(z^1)$ and $P(t^1|z^1, t^0) = P(t^1|z^1)$.

This graphical model shares some resemblance to Hidden Markov Models (HMM). However, one major difference is that there are dependencies between neighboring terms, which capture their co-occurrence.

4.2 Scoring Algorithm

As mentioned above, the scoring function needs to concretize the latent topic sequence in the graphical model depicted in Figure 2. The possible latent topic paths compose a trellis structure, which is shown in Figure 3. To marginalize the joint probability $P(t^{1:n}, z^{1:n})$ (refer to Equation 9) over all possible latent topic paths, we can employ a dynamic programming method, similar to the forward algorithm. The forward variable $\alpha_r(i)$ is defined as:

$$\alpha_r(i) \triangleq P(t^{1:r}, z^r = z_i), \qquad (10)$$

which is the score of the partial query $t^1 \cdots t^r$ given the topic z_i at the position r. When $r = 1$, we set $\alpha_1(i) = P(z^1 = z_i)P(t^1|z^1 = z_i, t^0)$. The recursive calculation of α is:

$$\alpha_r(i) = \Big[\sum_{z_j \in Z} \alpha_{r-1}(j) P(z^r = z_i|z^{r-1} = z_j)\Big] P(t^r|t^{r-1}, z_i), \qquad (11)$$

where $P(z^r = z_i|z^{r-1} = z_j)$ is independent of the position r, and equal to $P(z_i|z_j)$. Finally the score of a query can be calculated by summing over all possible z_i for $\alpha_n(i)$:

$$P(t^{1:n}) = \sum_{z_i \in Z} \alpha_n(i). \qquad (12)$$

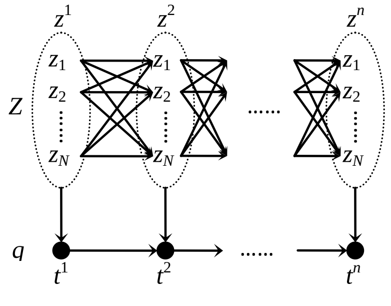

Figure 3: Trellis structure of latent topic sequences.

Therefore, to conduct the scoring, we need three kinds of model parameters, namely, $P(z_i)$, $P(z_j|z_i)$, and $P(t_l|z_j, t_k)$, which can be learned in the model training phase.

4.3 Latent Topic Analysis and Training Algorithm

The latent topics of the terms in queries are discovered from the query log. These latent sematic topics discovered are transferred to initialize the model parameters of the graphical model. Then, the historical queries are treated as training sequences to learn the model parameters.

4.3.1 Latent Topic Analysis of Query Log

A typical format of the records in query log can be represented as a triple $(query, clicked\_url, time)$. One direct way of using query log data for latent topic analysis is to treat each *clicked_url* as a single document unit. This approach will suffer from the data sparseness problem since most of URLs only involve very small number of queries. Instead of adopting such a simple strategy, we aggregate all the queries related to the same host together, and construct one pseudo-document for each host, denoted by $\mathcal{H}$. For example, the pseudo-document of "www.mapquest.com" consists of the queries such as "mapquest", "travel map", "driving direction", etc. Some general Web sites such as "en.wikipedia.org" and "www.youtube.com" are not suitable for latent topic analysis, because they involve large amount of queries which cover very diverse aspects. To tackle this problem, we first sort the host pseudo-documents in descending order according to the number of distinct query terms they have. Then, the top ranked pseudo-documents are eliminated in our latent topic discovery process.

We employ the standard Latent Dirichlet Allocation (LDA) algorithm [4] to conduct the latent semantic topic analysis on the collection of pseudo-documents. In particular, GibbsLDA[3] package is used to generate a set of latent topics $Z = \{z_1,\ z_2,\ \cdots,\ z_N\}$. Each topic z_i is associated with a multinomial distribution of terms, denoted by $P(t_k|z_i)$, where t_k is a term. Each t_k in a certain pseudo-document $\mathcal{H}$ is assigned a topic, denoted by $z(t_m|\mathcal{H})$ in this paper.

4.3.2 Latent Topic Transfer

We transfer the latent topics discovered above to initialize the model parameters in the beginning of training phase:

$$\hat{P}(z_i) = \frac{1}{|Z|}, \tag{13}$$

$$\hat{P}(z_j|z_i) = \frac{\exp\left(-D_{KL}(z_j||z_i)\right)}{\sum\limits_{z_k \in Z} \exp\left(-D_{KL}(z_k||z_i)\right)}, \tag{14}$$

$$\hat{P}(t_l|z_j, t_k) = \frac{\hat{P}(t_k, t_l, z_j)}{\hat{P}(t_k, z_j)} = \frac{\hat{P}(t_k, t_l|z_j)}{\hat{P}(t_k|z_j)}, \tag{15}$$

where $D_{KL}(z_j||z_i) = \sum_{t_k} P(t_k|z_j) \log \frac{P(t_k|z_j)}{P(t_k|z_i)}$ is the Kullback-Leibler divergence between two distributions $P(\cdot|z_i)$ and $P(\cdot|z_j)$. When z_i and z_j are highly related topics, the value of $\hat{P}(z_j|z_i)$ is large. $\hat{P}(t_k, t_l|z_j)$ is the probability that t_k and t_l co-occur in the topic z_j, and $\hat{P}(t_k|z_j)$ is the probability of t_k in z_j. $\hat{P}(t_k, t_l|z_j)$ and $\hat{P}(t_k|z_j)$ are calculated as:

$$\hat{P}(t_k, t_l|z_j) = \frac{cnt(t_k, t_l|z_j)}{\sum_m \sum_n cnt(t_m, t_n|z_j)}, \tag{16}$$

$$\hat{P}(t_k|z_j) = \frac{\sum_n cnt(t_k, t_n|z_j)}{\sum_m \sum_n cnt(t_m, t_n|z_j)}, \tag{17}$$

where $cnt(t_m, t_n|z_j)$ denotes the co-occurrence time of t_m and t_n in the topic z_j, and calculated as:

$$cnt(t_m, t_n|z_j) = \sum_{\mathcal{H}} \mathbf{1}_{\{t_m, t_n \in \mathcal{H}\ \wedge\ z(t_m|\mathcal{H}) = z(t_n|\mathcal{H}) = z_j\}}, \tag{18}$$

where $\mathbf{1}_{\{\}}$ is an indicator function, and $z(t_m|\mathcal{H})$ is the topic assigned to t_m in the pseudo-document $\mathcal{H}$, which can be obtained from the result of LDA in the previous step. Then we utilize a global language model to smooth $\hat{P}(t_l|z_j, t_k)$:

$$\tilde{P}(t_l|z_j, t_k) = \frac{cnt(t_k, t_l|z_j) + \mu_1 P(t_k) P(t_l)}{\sum_n cnt(t_k, t_n|z_j) + \mu_1 P(t_k)}, \tag{19}$$

where $P(t) = \frac{cnt(t)}{\sum_{t_i} cnt(t_i)}$ is the global probability of term t in the query log data, $cnt(t)$ denotes the frequency of t.

Note that we do not directly apply the probability $P(t_k|z_j)$ generated by LDA as $\hat{P}(t_k|z_j)$ in Equation 15. The reason is that $\hat{P}(t_l|z_j, t_k)$ should satisfy the constraint of $\sum_l \hat{P}(t_l|z_j, t_k) = 1$, and using $P(t_k|z_j)$ may violate this constraint.

4.3.3 Latent Topic Parameter Estimation

To conduct the training process, we refine the standard Baum-Welch algorithm [2]. Given a set of historical queries, we preprocess them so that the frequency of distinct queries is aggregated and collected. Since we assume that each query is independent of the others, the intermediate calculation can be illustrated with a single query. Then, the intermediate results for each single query are aggregated to obtain the final updating formulas with the similar way in [25].

For a query $q : t^1 \cdots t^n$, we define backward variable $\beta_r(i)$:

$$\beta_r(i) \triangleq P(t^{r+1:n}|z^r = z_i, t^r), \tag{20}$$

which is the score of partial query $t^{r+1} \cdots t^n$ given z^r and t^r. We define $\beta_n(i) = 1$. The recursive calculation of β is:

$$\beta_r(i) = \sum_{z_j \in Z} P(z_j|z_i) P(t^{r+1}|t^r, z_j) \beta_{r+1}(j). \tag{21}$$

Then the joint probability of a query q and $z^r = z_i$ can be derived as:

$$\begin{aligned} P(q, z^r = z_i) &= P(t^{1:r-1}, t^{r+1:n}|t^r, z^r = z_i) P(t^r, z^r = z_i) \\ &= P(t^{1:r-1}, t^r, z^r = z_i) P(t^{r+1:n}|z^r = z_i, t^r) \\ &= \alpha_r(i) \beta_r(i). \end{aligned} \tag{22}$$

Therefore, $P(q)$ can also be computed by $\sum_i \alpha_r(i)\beta_r(i)$.

Now we introduce two conditional probabilities. $P(z^r = z_i|q)$ is the probability that z^r takes the value z_i in q, which can be calculated as:

$$P(z^r = z_i|q) = \frac{\alpha_r(i)\beta_r(i)}{P(q)}. \tag{23}$$

The joint probability of $z^r = z_i$ and $z^{r+1} = z_j$ in a given q is $P(z^r = z_i, z^{r+1} = z_j|q)$:

$$P(z^r = z_i, z^{r+1} = z_j|q) = \frac{\alpha_r(i) P(z_j|z_i) P(t^{r+1}|t^r, z_j) \beta_{r+1}(j)}{P(q)}. \tag{24}$$

[3]http://gibbslda.sourceforge.net/

The next step is to aggregate the intermediate results and obtain the final updating formulas of $P'(t_l|z_j, t_k)$, $P'(z_j|z_i)$ and $P'(z_i)$. $P'(t_l|z_j, t_k)$ is designed as follows:

$$P'(t_l|z_j,t_k) = \mu_2 \frac{\sum_{q\in Q}\Bigg(\sum_{\substack{r=1 \\ s.t.\ t^r=t_k, t^{r+1}=t_l}}^{n-1} P(z^{r+1}=z_j|q)F(q)\Bigg)}{\sum_{q\in Q}\Bigg(\sum_{\substack{r=1 \\ s.t.\ t^r=t_k}}^{n-1} P(z^{r+1}=z_j|q)F(q)\Bigg)} + (1-\mu_2)\tilde{P}(t_l|z_j,t_k), \quad (25)$$

where Q is the set of training queries, and $F(q)$ is the frequency of q, μ_2 is used to control the weights of two parts, namely the initial value $\tilde{P}(t_l|z_j, t_k)$, and the newly estimated part. In this way, the value of $P'(t_l|z_j, t_k)$ does not solely depend on the queries which have sub-sequence $(t_k,\ t_l)$, a proportion of the initial $\tilde{P}(t_l|z_j, t_k)$ is also kept in $P'(t_l|z_j, t_k)$. Therefore, when a new query contains term sequence $(t_k,\ t_l)$, and no previous queries contain $(t_k,\ t_l)$, the value of $P'(t_l|z_j, t_k)$ can still be calculated relying on $\tilde{P}(t_l|z_j, t_k)$, and the score of the new query is still reliable. The updating formulas of $P'(z_j|z_i)$ and $P'(z_i)$ are straightforward and they are given as follows:

$$P'(z_j|z_i) = \frac{\sum_{q\in Q}\sum_{r=1}^{n-1} P(z^r = z_i, z^{r+1} = z_j|q)F(q)}{\sum_{q\in Q}\sum_{r=1}^{n-1} P(z^r = z_i|q)F(q)}, \quad (26)$$

$$P'(z_i) = \frac{1}{|Q|}\sum_{q\in Q} P(z^1 = z_i|q)F(q), \quad (27)$$

After one iteration, new parameters, namely, $P'(t_l|z_j, t_k)$, $P'(z_j|z_i)$, and $P'(z_i)$, will be used in the next iteration. This iterative procedure will stop when a pre-defined condition is satisfied.

5. DATA PREPROCESSING AND EXPERIMENT SETUP

5.1 Data Sets and Preprocessing

The query log data used in our experiments is the AOL data [27] spanning three months from 1 March, 2006 to 31 May, 2006. The raw data is composed of queries and clicks recorded by a search engine. Since it contains a lot of noise, we first clean the raw data as done by most of the existing works. If a query contains non-alphabetical character, or is a host navigation query such as "google.com" and "www.yahoo.com", it is removed. Then, the stop words are removed from the remaining queries. We adopt a hybrid method to detect user session boundary [19]. First, consecutive queries in the same session have to share at least one term. Second, the interval between two consecutive queries should be less than 10 minutes. After sessions are detected, we remove those sessions without any clicked URL. In each session, we remove the queries which are at the end of the session and have no clicked URL. Finally we obtain $7,041,319$ sessions, in which $1,276,498$ are multi-query sessions. There are $4,315,124$ unique queries and $673,073$ distinct terms.

The data set is split into two subsets by the time stamp: the historical set and the test set. The historical set contains the first two months' log, and the test set contains the third month's log. The pseudo-documents for latent topic analysis are constructed with the historical set as mentioned in Section 4.3. If a host involves less than 5 queries, it will be eliminated. Then all pseudo-documents are ranked in descending order according to the number of distinct terms they have. In order to filter out those Web sites that are too general, we remove URLs from the top 0.1% of the pseudo-documents. Finally, 189,670 pseudo-documents are retained and fed into the latent topic discovery algorithm, where the number of topics is set to 30. In the historical set, 1,882,638 queries with at least one clicked URL are used to build the training set, which is used in the parameter estimation.

Currently, the largest existing bookmark data collection is provided by Wetzker et al. [34]. In our experiment, we use a subset of the data spanning two years, namely, 2006 and 2007. In this period, 101,485,670 bookmarks were assigned to 39,232,530 distinct URLs by 891,479 users, including 2,523,190 distinct tags. To tackle the vocabulary gap between the AOL log and bookmark data, we filter out the tags that are not contained by the query log such as "socialsoftware", "visualstudio", etc. To tackle the problem of data sparseness, we only consider the URLs which are tagged by at least five users. After this filtering, 1,910,547 URLs are collected and used in candidate term generation. And the similarity threshold value used is 0.19 when filtering the noise. The smoothing parameter value of μ_1 is 3,000 in Equation 19.

5.2 Experiment Setup

For conducting the comparison, we implement a context based term association (CTA) method proposed by Wang and Zhai [32], and follow the advice of the authors on some implementation details. This method is based on an observation that terms with similar meaning tend to co-occur with the same or similar terms in the queries. For example, both "auto" and "car" often occur with "rental", "pricing", etc. Therefore, both candidate term generation and query scoring in CTA are conducted by exploiting the association of term contexts in the historical queries. A contextual model is defined to capture context similarity, and a translation model is defined to score the quality of a candidate query. We denote the two steps of CTA, namely, candidate term generation, and query scoring, as "CTA-CAND" and "CTA-SCR" respectively. We generate the contextual and translation models for the top 100,000 terms in the historical set. Then the threshold given in [32] is applied to filter out the noise. We mainly compare the performance of our framework and CTA for query term substitution. A conservative strategy is adopted, specifically, only one term in the original query will be replaced. This is also the strategy adopted by some previous works such as [32].

Because manual evaluation of the output of query refinement methods is time-consuming and very labor intensive, we conduct an automatic evaluation by utilizing the session information of query log. In a search session, when users feel unsatisfied with the results of current query, they may refine current query and search again. After obtaining satisfactory results, they may stop searching. The importance of the terminal URL was well discussed in previous works [14, 35]. Therefore, an automatic and reliable evaluation can be conducted based on this observation. We introduce the definitions of two kinds of query as follows:

DEFINITION 2 (SATISFIED QUERY). *In a user session, the query which causes at least one URL clicked and is located in the end of the session is called a satisfied query.*

Table 3: Substitution term pairs generated by our method.

air		birthday		car		cd		children		health	
cand.	*NMI*	*cand.*	*NMI*	*cand.*	*NMI*	*cand.*	*NMI*	*cand.*	*NMI*	*cand.*	*NMI*
apollo	0.135	birthdays	0.085	cars	0.319	dvd	0.130	kids	0.207	medical	0.135
airline	0.127	greeting	0.061	auto	0.260	cds	0.081	parenting	0.136	fitness	0.131
airlines	0.113	ecards	0.059	automotive	0.176	iso	0.077	child	0.080	medicine	0.126
flights	0.105	anniversary	0.053	automobile	0.132	burning	0.071	baby	0.072	nutrition	0.124
flight	0.100	ecard	0.043	vehicle	0.082	cdrom	0.064	parents	0.053	diet	0.103
house		lotto		mail		msn		music		pc	
cand.	*NMI*	*cand.*	*NMI*	*cand.*	*NMI*	*cand.*	*NMI*	*cand.*	*NMI*	*cand.*	*NMI*
houses	0.161	lottery	0.321	email	0.267	messenger	0.219	audio	0.135	computer	0.092
housing	0.122	lotteries	0.127	webmail	0.099	im	0.115	musica	0.118	computers	0.088
realestate	0.104	loto	0.085	smtp	0.096	icq	0.105	musik	0.087	hardware	0.074
estate	0.079	powerball	0.081	gmail	0.091	aim	0.103	bands	0.076	windows	0.069
property	0.063	lotterie	0.042	spam	0.090	chat	0.074	rock	0.060	xp	0.060

Table 4: Comparison of substitution pair generation for high frequency terms.

google		yahoo		county		lyrics		school		myspace	
Our	*CTA*	*Our*	*CTA*	*Our*	*CTA*	*Our*	*CTA*	*Our*	*CTA*	*Our*	*CTA*
googlemaps	goggle	yui	msn	berks	count	songs	lyric	education	schools	facebook	space
search	yahoo	msn	aol	orange	couty	lyric	listen	college	speed	socialnetworking	xanga
seo	googl	google	google	counties	parish	song	song	schools	heels	socialnetworks	profile
gmail	satellite	pipes	aim	speedway	township	letras	lryics	teaching	scool	socialnetwork	html
maps	googles	search	voice	carfax	couny	chords	titles	university	heel	sns	premade
searchengine	gogle	searchengine	ako	alamo	conty	canciones	lyrcis	learning	scholl	friendster	glitter
googleearth	goole	messenger	yaho	hertz	dade	paroles	lyricks	educational	shool	socialsoftware	cool
map	virtual	woman	electronic	kia	coutny	tablature	lyics	teacher	fiber	profile	page
mashup	planet	engine	yhoo	armored	okeechobee	tabs	remix	students	cholesterol	social	background
adsense	msn	aim	monkey	towing	twp	songtexte	album	edu	protein	socialmedia	website
university		ebay		college		bank		city		mapquest	
Our	*CTA*	*Our*	*CTA*	*Our*	*CTA*	*Our*	*CTA*	*Our*	*CTA*	*Our*	*CTA*
college	college	auction	outboard	university	colleges	banking	bankof	cities	yankees	drivingdirections	maps
universities	univ	auctions	trolling	colleges	collage	banks	holland	urban	knicks	onlinemapping	defensive
academic	athletics	sell	general	school	colege	money	morning	urbanism	mets	navteq	drunk
uni	univeristy	selling	boat	universities	webshots	finance	guns	oklahoma	jets	sidesearch	mcnally
school	colleges	paypal	mitsubishi	schools	colleg	cash	scouts	urbanplan	ciy	teleatlas	quest
academia	unversity	ecommerce	kia	education	outreach	savings	mid	chicago	aviation	routeplanners	google
colleges	fullerton	shipping	crate	students	collge	finances	mall	newyork	ave	routefinder	petty
education	parks	marketplace	rebuilt	highered	colledge	payment	mbna	miami	cty	direcciones	aaa
edu	universtiy	sale	clock	academia	policing	account	reserve	nyc	blvd	kaardid	drinking
gradschool	prisons	sales	yamaha	edu	helpers	hsbc	latin	places	counties	strade	directions

DEFINITION 3 (UNSATISFIED QUERY). *The query which is located just ahead of the satisfied query in the same user session.*

We collect a set of unsatisfied queries used as the input, and collect their corresponding satisfied queries treated as the benchmark query set of the refinement task. Note that this requirement is only for conducting automatic evaluation, and our framework can take any query as input.

6. EXPERIMENTAL RESULTS

6.1 Candidate Term Generation

The candidate terms for 12 selected terms are given in Table 3. For each term, the top 5 candidates are given along with their NMI scores. In general, there are four kinds of relations between the terms in a substitution pair. The first kind refers to the fact that the candidates' meaning is identical with input terms' meaning such as "automotive" and "auto" for "car", "kids" for "children", "lottery" for "lotto", etc. The second kind is that the candidate terms' meaning is related to the input term's meaning, such as "dvd" for "cd", "fitness" for "health", "estate" for "house", "messenger" for "msn". etc. The third kind refers to the fact that the candidate terms have a similar property of the input terms such as "icq" and "aim" for "msn". The last kind refers to other types of strongly associated relation. In the last case, although the candidate terms do not have similar meaning or property of the inputs, the relationship between them is quite obvious. For example, "powerball" is a kind of lottery ticket, "rock" is a genre of popular music. Interestingly, two non-English terms are suggested for "music", namely, "musica" from Spanish and "musik" from Swedish, and both are correct. We can see that social tagging data can provide us some useful term pairs with similar or related meaning.

We find that when dealing with high frequency terms, CTA suggests a lot from typos. The candidates of the top 12 most frequent terms in the AOL log are given in Table 4. It can been seen that typos are recommended by CTA for 8 input terms, except for "myspace", "ebay", "bank" and "mapquest". After the filtering is performed with the threshold given in [32], the top 5 remaining candidates for "county" are "count", "couty", "couny", "conty" and "dade", and for "lyrics" are "listen", "song", "lryics", "lyrcis" and "lyics". The main reason of this observation is that their candidate generation method is context based, and the filtering method relies on the co-occurrence of two terms among different sessions. After users issue a query containing typo terms and cannot get satisfied results, they may correct it manually. But this typo is already recorded in the query log. Worse still, most of search engines can return some results even for a query with typo. So after users click any one of

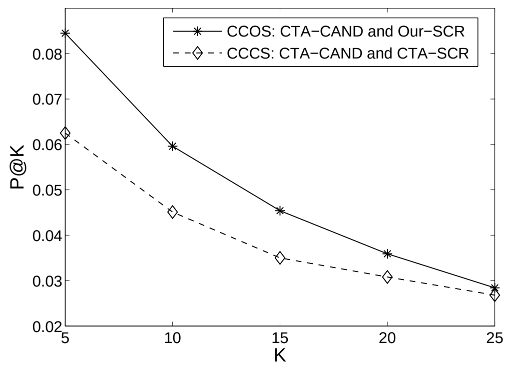

Figure 4: Comparison of scoring performance between CTA-SCR and Our-SCR.

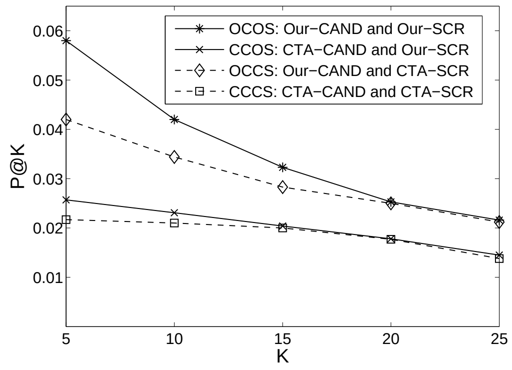

Figure 5: Refinement performance of different combinations on QS2.

the results for the query with typo, we cannot easily filter out this query. Therefore, the typo and its corresponding correct term have very similar context and often co-occur in the sessions. In contrast, our method utilizes the bookmark data, which has a better quality compared with the query log. When a user provides a typo tag, it may be corrected when the user comes back to maintain his or her bookmarks. On the other hand, the bookmark data also has its own shortcoming. A user may connect several terms as a single tag, such as "googlemaps" and "searchengine". Fortunately, it is not difficult to split this kind of tags into single terms.

6.2 Comparison of Scoring Performance

Recall that our framework contains two major components, namely, the graphical model for scoring and the candidate term generation from social tagging data, and they aim at solving different steps or tasks in query refinement. To compare the performance of query scoring between our graphical model and CTA, we randomly select 1,000 multi-query sessions from the test set of the query log, and the unsatisfied queries in these sessions are used as input of the query refinement task. This query set is named as Query Set 1 (QS1). As mentioned before, the corresponding satisfied queries in these sessions are used as the benchmark results.

The scoring component of CTA, i.e. CTA-SCR, is a term context based method, and it is designed to score the candidate query generated from the context based candidate generation method, i.e. CTA-CAND. To compare the performance of two scoring methods, namely, CTA-SCR, and our graphical model (called Our-SCR), we apply CTA-CAND to generate candidate terms from the historical query set for them. For a query q in QS1, CTA-CAND outputs a list of candidate terms for each term of q. Then, each candidate term is used to generate a candidate query. All candidate queries of q are fed into one scoring method to assess the quality. Then, according to their score, the top candidates are determined as the final output results for q.

We adopt the metrics P@K (precision at K) to evaluate the result, and K is the number of top result queries given by the model. For each method, at most 25 result queries are considered here. The performance of both models is given in Figure 4, where CCOS denotes the scoring performance of our graphical model. It can be observed that our graphical model outperforms CTA (denoted by CCCS) significantly when K is small. The P@5 value of our graphical model is 0.085, while the P@5 value of CTA is 0.062. The differences between the performances of our method and CTA are significant with significance level 0.05. The major reason for the different performance is that our graphical model scores a query taking into account the semantic consistency of the terms, which cannot be captured by the comparison baseline. Because most of the input queries have less than 25 candidate queries, both methods achieve similar P@25.

The effect of the parameter μ_2 is depicted in Figure 6. In general, a large value of μ_2 can achieve better result than a small value. Recall that we randomly generate QS1 from the test set. Commonly used queries have more chance to be selected, because they are issued many times by users. Therefore, there is more information from historical queries that can be used in scoring their candidate queries. Refer to Equation 25, a large value of μ_2 reserves small amount of initial estimation value of $\tilde{P}(t_l|z_j,t_k)$, and the value of $P'(t_l|z_j,t_k)$ depends more on the re-estimation part from the historical queries. This weight distribution is preferred when dealing with queries in QS1. If the value of $P'(t_l|z_j,t_k)$ only depends on the re-estimation part, i.e., $\mu_2 = 1.0$, the result is worse than that when we reserve some initial estimation of $\tilde{P}(t_l|z_j,t_k)$, for example, $\mu_2 = 0.7$. It shows that the initial estimation from the latent topic analysis can capture some intrinsic feature of the queries in Web search.

6.3 Performance of Unfamiliar Query Refinement

To investigate the efficacy of our candidate term generation method based on social tagging data, we select another query set from the test set in a similar way of selecting QS1, named as Query Set 2 (QS2). The only difference is that the queries in QS2 are unfamiliar, so that we can investigate whether social tagging data can help solve the problem of lacking information.

The queries in QS2 are unfamiliar, therefore, this poses some challenges in both tasks of query refinement as we discussed in Section 1. Figure 5 depicts the results of different combinations. Our-CAND denotes the candidate term generation component of our framework. It can be observed that the combination OCOS performs the best, and followed by combination OCCS. Combinations CCOS and CCCS achieve similar performance. In summary, the combinations with Our-CAND as the candidate term genera-

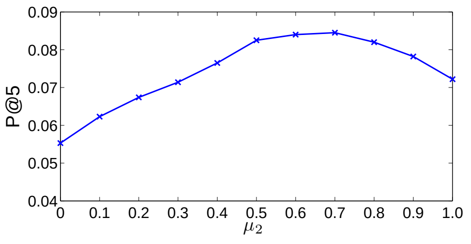

Figure 6: The effect of μ_2 under the CCOS method in the comparison of the scoring performance.

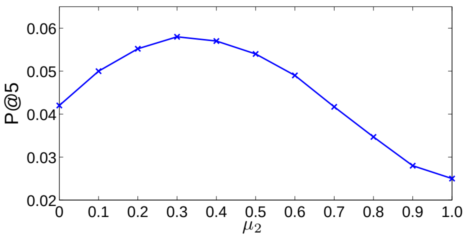

Figure 7: The effect of μ_2 under the OCOS method in unfamiliar query refinement.

tion component can perform better. This is mainly due to the lack of information for these unfamiliar query, and good candidate terms cannot be generated by CTA-CAND. Furthermore, Our-SCR performs better than CTA-SCR. Our graphical model scores the query in a latent semantic space. Therefore, even when the term context information is very sparse, Our-SCR can still achieve a reliable score.

The effect of parameter μ_2 in the combination OCOS is given in Figure 7. It can be observed that when refining the unfamiliar queries, a small value of μ_2, say 0.3, can achieve a better performance in query scoring, because a small value of μ_2 reserves larger amount of initial estimation value, i.e. $\tilde{P}(t_l|z_j, t_k)$. When the historical query information is sparse, reserving more of $\tilde{P}(t_l|z_j, t_k)$ value can offer a reliable score.

6.4 Case Study of Refined Queries

We conduct some case studies to analyze the query refinement result qualitatively. Table 5 depicts the top 5 output queries from our model for 4 input queries, namely "dutch folklore" and "racing logistics" from QS2, and "kids anger disorders" and "discount gucci eyeglasses" from QS1. To refine "dutch folklore" and "racing logistics", the main difficulty is the generation of candidate terms. For example, "folklore" is not a frequently used term in the historical query set, but in the bookmark data it is employed to tag many URLs. Our candidate generation method can figure out some very good candidate terms such as "myths", "mythology", and "legend". On the other hand, to refine "kids anger disorders", the main difficulty is to detect and maintain the semantic meaning consistency of the candidate queries. Because each term in "kids anger disorders" has tens of candidate substitution terms, the semantic consistency is not easy to maintain when generating a new query.

7. RELATED WORK

Huang et al. analyzed and evaluated different types of query refinement in the query log [20]. Except the three mentioned types, namely, substitutions, expansion, and deletions, they also investigated some other fine-grained types such as stemming [28], spelling correction [10], abbreviation expansion [33], etc. In [17], a Conditional Random Field model is proposed to conduct these fine-grained refinement operations. Chirita et al. exploited user's personal information repository such as text documents, emails and cached Web pages to implicitly personalize the query expansion for the user [9]. Sadikov et al. conducted an investigation on clustering the refinements of the same original query [30]. The clusters provide a summary of the possible diverse interests of the users who issue the same original query. Their method can also be used to achieve a better personalized query refinement. Some works investigated the effectiveness of using anchor texts in query refinement [12, 23]. They find that anchor text is also an effective resource.

In query suggestion, a commonly used method is mining the click graph [13, 26]. In [26], a sub-bipartite graph of queries and URLs is generated for a particular input query. Then the hitting time of each node is computed by performing a random walk on the subgraph. After that, the query nodes which have the top n earliest hitting time are used as the suggestions. Deng et al. constructed a query-to-query graph instead of the bipartite graph, and the random walk is conducted on the entire graph [13]. They also proposed an entropy based method to calculate the weight of the edges. Cao et al. investigated context aware query suggestion by mining query log [8]. They used a clustering method on the bipartite graph to summarize queries into concepts. Then user sessions are employed to mine the concept patterns, which can help to provide more precise suggestions.

Table 5: Some examples of query refinements.

dutch folklore	kids anger disorders
dutch literature	kids feelings disorders
dutch myths	kids anxiety disorders
dutch mythology	kids emotions disorders
dutch legend	kids mental disorders
holland folklore	teens anger disorders
racing logistics	discount gucci eyeglasses
sports logistics	discount gucci sunglasses
sport logistics	discount vuitton eyeglasses
racing operations	discount gucci glasses
race logistics	discount versace eyeglasses
playing logistics	discount chanel eyeglasses

8. CONCLUSIONS AND FUTURE WORK

In this paper, we present a framework for performing query refinement, and this framework mainly has two contributions. First, a graphical model is proposed to score candidate queries, and this model can detect and maintain the semantic consistency of terms in queries. Second, a method is presented to mine term pairs with similar meaning from social tagging data. From the experimental results, we observe that taking the semantic consistency into account can help to give a more reliable prediction of query quality. Furthermore, social tagging data can provide useful information in generating similar term pairs, which is especially important for some unfamiliar terms in previous queries.

Currently, phases are separated into single words in our framework in both latent topic analysis and term dependency estimation. One future direction is to promote our model from single word level to phrase level. Some models such as topical N-grams model [31] and phrase translation model [15] have attested the efficacy of the phrase level solution. In our current model, the score of a candidate query is independent of the current session in which the original query locates and the user who issues the original query. Therefore, another future direction is to consider more personalized or other available information in scoring queries such as the users' Web search history, existing queries in the current user session, etc.

9. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR'06*, pages 19–26, 2006.

[2] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Math. Statist.*, 41(1):164–171, 1970.

[3] B. Billerbeck, F. Scholer, H. E. Williams, and J. Zobel. Query expansion using associated queries. In *CIKM'03*, pages 2–9, 2003.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[5] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: model and applications. In *CIKM'08*, pages 609–618, 2008.

[6] P. Boldi, F. Bonchi, C. Castillo, D. Donato, and S. Vigna. Query suggestions using query-flow graphs. In *WSCD'09*, pages 56–63, 2009.

[7] Y. Cai and Q. Li. Personalized search by tag-based user profile and resource profile in collaborative tagging systems. In *CIKM'10*, pages 969–978, 2010.

[8] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *SIGKDD'08*, pages 875–883, 2008.

[9] P. A. Chirita, C. S. Firan, and W. Nejdl. Personalized query expansion for the web. In *SIGIR'07*, pages 7–14, 2007.

[10] S. Cucerzan and E. Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *EMNLP'04*, pages 293–300, 2004.

[11] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic query expansion using query logs. In *WWW'02*, pages 325–332, 2002.

[12] V. Dang and B. W. Croft. Query reformulation using anchor text. In *WSDM'10*, pages 41–50, 2010.

[13] H. Deng, I. King, and M. R. Lyu. Entropy-biased models for query representation on the click graph. In *SIGIR'09*, pages 339–346, 2009.

[14] D. Downey, S. Dumais, D. Liebling, and E. Horvitz. Understanding the relationship between searchers' queries and information goals. In *CIKM '08*, pages 449–458, 2008.

[15] J. Gao, X. He, and J.-Y. Nie. Clickthrough-based translation models for web search: from word models to phrase models. In *CIKM'10*, pages 1139–1148, 2010.

[16] J. Guo, X. Cheng, G. Xu, and H. Shen. A structured approach to query recommendation with social annotation data. In *CIKM'10*, pages 619–628, 2010.

[17] J. Guo, G. Xu, H. Li, and X. Cheng. A unified and discriminative model for query refinement. In *SIGIR'08*, pages 379–386, 2008.

[18] B. He and I. Ounis. Combining fields for query expansion and adaptive query expansion. *Inf. Process. Manage.*, 43:1294–1307, 2007.

[19] D. He, A. Göker, and D. J. Harper. Combining evidence for automatic web session identification. *Inf. Process. Manage.*, 38(5):727–742, 2002.

[20] J. Huang and E. N. Efthimiadis. Analyzing and evaluating query reformulation strategies in web search logs. In *CIKM'09*, pages 77–86, 2009.

[21] T. Joachims. Optimizing search engines using clickthrough data. In *SIGKDD'02*, pages 133–142, 2002.

[22] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW'06*, pages 387–396, 2006.

[23] R. Kraft and J. Zien. Mining anchor text for query refinement. In *WWW'04*, pages 666–674, 2004.

[24] G. Kumaran and J. Allan. Effective and efficient user interaction for long queries. In *SIGIR'08*, pages 11–18, 2008.

[25] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi. An introduction to the application of the theory of probabilistic functions of markov process to automatic speech recognition. *Bell System Technical Journal*, 62(4):1035–1074, 1983.

[26] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *CIKM'08*, pages 469–478, 2008.

[27] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *InfoScale'06*, 2006.

[28] F. Peng, N. Ahmed, X. Li, and Y. Lu. Context sensitive stemming for web search. In *SIGIR'07*, pages 639–646, 2007.

[29] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *SIGKDD'05*, pages 239–248, 2005.

[30] E. Sadikov, J. Madhavan, L. Wang, and A. Halevy. Clustering query refinements by user intent. In *WWW'10*, pages 841–850, 2010.

[31] X. Wang, A. McCallum, and X. Wei. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *ICDM'07*, pages 697–702, 2007.

[32] X. Wang and C. Zhai. Mining term association patterns from search logs for effective query reformulation. In *CIKM'08*, pages 479–488, 2008.

[33] X. Wei, F. Peng, and B. Dumoulin. Analyzing web text association to disambiguate abbreviation in queries. In *SIGIR'08*, pages 751–752, 2008.

[34] R. Wetzker, C. Zimmermann, and C. Bauckhage. Analyzing social bookmarking systems: A del.icio.us cookbook. In *ECAI'08*, pages 26–30, 2008.

[35] R. W. White, M. Bilenko, and S. Cucerzan. Studying the use of popular destinations to enhance web search interaction. In *SIGIR '07*, pages 159–166, 2007.

[36] X. Xue, S. Huston, and W. B. Croft. Improving verbose queries using subset distribution. In *CIKM'10*, pages 1059–1068, 2010.