Learning Latent Semantic Relations from Clickthrough Data for Query Suggestion

Hao Ma, Haixuan Yang, Irwin King, Michael R. Lyu
Dept. of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
{hma, hxyang, king, lyu}@cse.cuhk.edu.hk

ABSTRACT

For a given query raised by a specific user, the *Query Suggestion* technique aims to recommend relevant queries which potentially suit the information needs of that user. Due to the complexity of the Web structure and the ambiguity of users' inputs, most of the suggestion algorithms suffer from the problem of poor recommendation accuracy. In this paper, aiming at providing semantically relevant queries for users, we develop a novel, effective and efficient two-level query suggestion model by mining clickthrough data, in the form of two bipartite graphs (user-query and query-URL bipartite graphs) extracted from the clickthrough data. Based on this, we first propose a joint matrix factorization method which utilizes two bipartite graphs to learn the low-rank query latent feature space, and then build a query similarity graph based on the features. After that, we design an online ranking algorithm to propagate similarities on the query similarity graph, and finally recommend latent semantically relevant queries to users. Experimental analysis on the clickthrough data of a commercial search engine shows the effectiveness and the efficiency of our method.

Categories and Subject Descriptors

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

General Terms

Algorithms, Performance, Experimentation

Keywords

Similarity Propagation, Matrix Factorization, Query Suggestion, Clickthrough Data, Web Search

1. INTRODUCTION

With the exponential growth of information on the World Wide Web, Web search engines provide an indispensable interface for Web users to obtain any kind of information they may seek. Although current commercial search engines have been proved to be successful for recommending the most relevant Web pages to users, there are several outstanding issues that can potentially degrade the quality of search results, and these merit investigation. The first one is the ambiguity which commonly exists in the natural language. Queries containing ambiguous terms may confuse the search engine into retrieving Web pages which do not satisfy the information needs of users. Another consideration, as reported in [12, 25], is that users tend to submit short queries consisting of only one or two terms under most circumstances, and short queries are more likely to be ambiguous. Through the analysis of a commercial search engine's query logs recorded over three months in 2006, we observe that 19.4% of Web queries are single term queries, and a further 30.5% of Web queries contain only two terms. Thirdly, in most cases, the reason why users search is that they have little or even no knowledge about the topic they are searching for. In order to find satisfactory answers, users have to rephrase their queries constantly.

To overcome all of these problems, a valuable technique, query suggestion, has been employed by some famous commercial search engines, such as *Yahoo!*[1], *Live Search*[2], *Ask*[3] and *Google*[4], to recommend relevant queries to users. However, due to the commercial reasons, few public papers have been released to unveil the methods they adopt.

Typically, query suggestion is based on local (i.e., search result sets) and global (i.e., thesauri) document analysis [31], or anchor text analysis [19]. However, these traditional methods have difficulty summarizing the latent meaning of a Web document due to the huge noise embedded in each Web page. Moreover, this noise is not easily removed by machine learning methods. In order to avoid these problems, some additional data sources are likely to be very helpful to improve the recommendation quality.

In fact, clickthrough data is an ideal source for mining relevant queries. In the typical search scenario, a user initiates a query, and submits it to a search engine. The search engine returns a set of ranked related Web pages or documents to this user. The user then clicks some pages of interest. Some users even refine their queries in order to find the desired information. Therefore, the collection of queries is likely to well reflect the relatedness of the target Web pages [26].

[1]http://www.yahoo.com
[2]http://www.live.com
[3]http://www.ask.com
[4]http://www.google.com

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

Table 1: Samples of search engine clickthrough data

ID	Query	URL	Rank	Time
358	facebook	http://www.facebook.com	1	2008-01-01 07:17:12
358	facebook	http://en.wikipedia.org/wiki/Facebook	3	2008-01-01 07:19:18
3968	apple iphone	http://www.apple.com/iphone/	1	2008-01-01 07:20:36
1398	Michael Jordan	http://www.youtube.com/watch?v=OFxXSXGd4hs	4	2008-01-01 07:30:18
...	...	...	...	...

Recently, as a valuable data source, clickthrough data has been employed in some research work in order to optimize ranking of Web search results [1, 15, 16, 26, 29], improve clustering accuracy [4, 30], or conduct other interesting work [22, 24]. However, most of these references extract only the query-URL bipartite graph of the clickthrough data for analysis, and ignore the information of users who issued the queries. Actually, users perform as the most important role in the clickthrough data, since all the queries are issued by the users, and which URLs to click are also decided by the users. The connections between queries and URLs are essentially bridged by different kinds of users. Moreover, if two distinct users issued the similar set of queries, we can assume that these two users are very similar since they have similar information needs. From the above analysis, we cannot ignore the users in the clickthrough data.

In this paper, by analyzing the clickthrough data, we develop a query suggestion framework using two-level latent semantic analysis. We first extract two bipartite graphs, which are user-query and query-URL bipartite graphs. Then we give solutions to the following two problems: (1) How to learn the query latent feature space from these two bipartite graphs, and (2) How to recommend semantically relevant queries to users?

As to the first problem, we develop a joint matrix factorization method which fuse user-query and query-URL bipartite graphs together to learn the low-dimensional query latent feature space. Then we build a query graph based on the representation of query space. In order to address the second problem, we develop a novel, effective, and efficient similarity propagation model, which not only suggests a list of queries relevant to the queries submitted by users, but also ranks the query list based on the similarity scores.

We evaluate our model for query suggestion using clickthrough data of a commercial search engine. We evaluate our model from different angles: (1) First, it is assessed by a panel of three experts. (2) Then, we evaluate it in terms of the ground truth extracted from the ODP[5] (Open Directory Project) database. (3) Finally, we measure the efficiency of our online query suggestion algorithm by measuring how much CPU time that it needs. The results show that our method is both effective and efficient for improving the recommendation quality, as well as generating semantically related queries to users.

The rest of the paper is organized as follows. We review related work in Section 2. Section 3 describes the construction of two bipartite graphs, and proposes a joint matrix factorization method of learning query latent feature space. Section 4 presents the similarity propagation model as well as the method for recommending queries. In Section 5, we demonstrate the empirical analysis of our models and algorithms. Finally, conclusions and future work are given in Section 6.

[5]http://www.dmoz.org

2. RELATED WORK

Our work addresses two important research topics: query suggestion or query recommendation, and clickthrough data analysis.

2.1 Query Suggestion

The intention of query suggestion is similar to that of query expansion [6, 7, 27, 31], query substitution [17] and query refinement [19, 28], which all focus on improving the queries submitted by users. Query suggestion is closely related to query expansion or query substitution which extends the original query with new search terms to narrow the scope of the search. But different from query expansion, query suggestion aims to suggest full queries that have been formulated by previous users so that query integrity and coherence are preserved in the suggested queries [10]. Query refinement is another closely-related notion, since the objective of query refinement is interactively recommending new queries related to a particular query.

In [31], local and global documents are employed in query expansion by applying the measure of global analysis to the selection of query terms in local feedback. Although experiment shows that this method is generally more effective than global analysis, it performs worse than the query expansion method proposed in [7] based on user interactions recorded in user logs. In another approach reported in [19], anchor texts are employed for the purpose of query refinement. This work is based on the observation that Web queries and anchor texts are highly similar.

In [3] and [8], two query recommendation methods based on clickthrough data are proposed. The main disadvantage of these two algorithms is that they ignore the rich information embedded in the query-click bipartite graph, and consider only queries that appear in the query logs, potentially losing the opportunity to recommend highly semantically related queries to users.

In addition, query suggestion technology has also been developed and applied into several other promising topics, such as pay-for-performance search [11], question-answering service [14], personalized search [6], etc.

2.2 Clickthrough Data Analysis

In the field of clickthrough data analysis, the most common usage is for optimizing Web search results or rankings [1, 15, 16, 26, 29]. In [29], Web search logs are utilized to effectively organize the clusters of search results by (1) learning "interesting aspects" of a topic and (2) generating more meaningful cluster labels. In [16], a ranking function is learned from the implicit feedback extracted

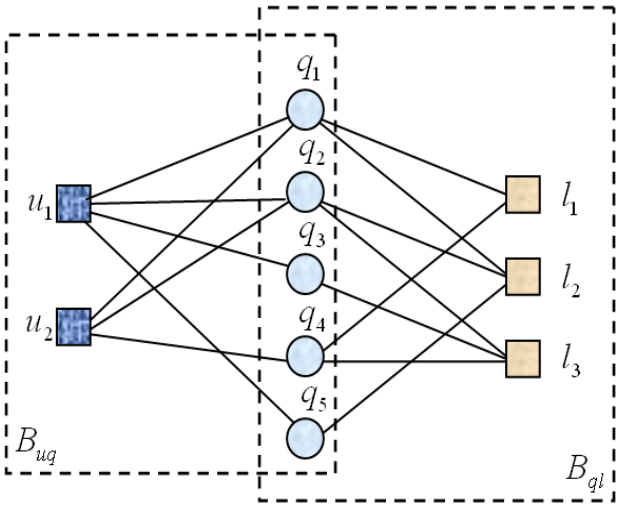


Figure 1: An example of two bipartite graphs

from search engine clickthrough data to provide personalized search results for users. Besides ranking, clickthrough data is also well studied in the query clustering problem [4, 30]. Query clustering is a process used to discover frequently asked questions or most popular topics on a search engine. This process is crucial for search engines based on question-answering [30]. Recently, clickthrough data has been analyzed and applied to several interesting research topics, such as Web query hierarchy building [24] and extraction of class attributes [22].

3. MATRIX FACTORIZATION

Before suggesting queries to users, we first need to preprocess clickthrough data. In this section, we first introduce the structure of clickthrough data and how to construct two bipartite graphs. Then we present a novel joint matrix factorization method to learn the low-rank representation for queries.

3.1 Construction of Bipartite Graph

Clickthrough data records the activities of Web users, which reflects their interests and the latent semantic relationships between users and queries, as well as queries and clicked Web documents. As shown in Table 1, each line of clickthrough data has the following information: a user ID (u), a query (q) issued by the user, a URL (l) on which the user clicked, the rank (r) of that URL, and the time (t) at which the query was submitted for search. Thus the clickthrough data can be represented by a set of quintuples $\langle u, q, l, r, t\rangle$. From a statistical point of view, the query word set corresponding with a Web page contains human knowledge on how the pages are related with their issued queries [26]. Thus, in this paper, we utilize the relationships of users and queries, as well as queries and Web pages for the construction of two bipartite graphs containing three types of vertices $\langle u, q, l\rangle$. The information of ranks and times is ignored.

For the user-query bipartite graph, consider an undirected bipartite graph $B_{uq} = (V_{uq}, E_{uq})$, where $V_{uq} = U \cup Q$, $U = \{u_1, u_2, ..., u_m\}$, and $Q = \{q_1, q_2, ..., q_n\}$. $E_{uq} = \{(u_i, q_j)|$ there is an edge from u_i to $q_j\}$ is the set of all edges. The edge (u_i, q_j) exists in this bipartite graph if and only if a user u_i issued a query q_j.

For the query-URL bipartite graph, consider another undirected bipartite graph $B_{ql} = (V_{ql}, E_{ql})$, where $V_{ql} = Q \cup L$, $Q = \{q_1, q_2, ..., q_n\}$, and $L = \{l_1, l_2, ..., l_p\}$. $E_{ql} = \{(q_i, l_j)|$ there is an edge from q_i to $l_j\}$ is the set of all edges. The edge (q_j, l_k) exists if and only if a user u_i clicked a URL l_k after issuing an query q_j.

Figure 1 is an example of the representation of two bipartite graphs of clickthrough data. The left side rectangle in dashed line contains the user-query bipartite graph B_{uq}, while the right side rectangle in dashed includes the query-URL bipartite graph B_{ql}.

In order to perform matrix factorization task on bipartite graph B_{uq} and B_{ql} in the following sections, we transform these two bipartite graphs into two matrices R and S, respectively. For the $m \times n$ user-query matrix R, rows represent users, and columns represent queries. The value of r_{ij} specifies how many times that user u_i issued query q_j. From another aspect, this value indicates how much u_i need information about query q_j. For the $n \times p$ query-URL matrix S, similar to R, we employ s_{jk} to quantify how many times that a query q_j has been connected to the URL l_k by different users. Here, "different" means that if a user click the same query-URL pair several times, we only count it once. This consideration can best discover the relationship between queries and URLs.

3.2 Matrix Factorization for User-Query and Query-URL Matrices

The idea of user-query matrix factorization is to derive a high-quality low-dimensional feature representation U and Q of users and queries based on analyzing the user-query matrix R. Here, U is a $d \times m$ matrix, with each column being the d-dimensional feature vector of a user, while Q is a $d \times n$ matrix, with each column being the d-dimensional feature vector of a query. Then we define the optimization function as follows:

$$\begin{aligned}\mathcal{H}(R, U, Q) &= \min_{U,Q} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij}^{R} (r_{ij} - U_i^T Q_j)^2 \\ &+ \frac{\alpha_u}{2} \|U\|_F^2 + \frac{\alpha_q}{2} \|Q\|_F^2, \end{aligned} \quad (1)$$

where α_u and α_q are small positive numbers, $\|\cdot\|_F^2$ denotes the Frobenius norm, and I_{ij}^R is the indicator function that is equal to 1 if user i issued query j and equal to 0 otherwise. The optimization aims to approximate observed value r_{ij} by $U_i^T Q_j$, a product of two low-rank vectors, with regularization on U and Q. A local minimum of the objective function given by Eq. (1) can be found by performing gradient descent in U, Q.

In order to decrease the noise, we normalize the value of r_{ij} by the maximum value of row i in user-query matrix R, denoted as r_{ij}^*. Now the value of r_{ij}^* is in the range of $[0, 1]$. For the multiplication $U_i^T Q_j$, we also bound it into range $[0, 1]$ by employing the logistic function $g(x) = 1/(1 + \exp(-x))$. Hence, Eq. (1) is enhanced to

$$\begin{aligned}\mathcal{H}(R, U, Q) &= \min_{U,Q} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij}^{R} (r_{ij}^* - g(U_i^T Q_j))^2 \\ &+ \frac{\alpha_u}{2} \|U\|_F^2 + \frac{\alpha_q}{2} \|Q\|_F^2. \end{aligned} \quad (2)$$

Now let us consider the query-URL matrix. Similar to user-query matrix, we also use two d-dimensional matrices Q and L to represent queries and URLs, respectively, where L is a $d \times p$ matrix, with each column being the d-dimensional feature vector of a URL. Similar to Eq. (2), the idea of query-URL matrix factorization can be represented by

$$\begin{aligned}\mathcal{H}(S,Q,L) &= \min_{Q,L} \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{p} I_{jk}^{S}(s_{jk}^{*} - g(Q_j^T L_k))^2 \\ &+ \frac{\alpha_q}{2}\|Q\|_F^2 + \frac{\alpha_l}{2}\|L\|_F^2, \end{aligned} \quad (3)$$

where I_{jk}^{S} is the indicator function that is equal to 1 if query j is clicked to URL k and equal to 0 otherwise, α_q and α_l are small positive numbers. Similar to user-query matrix, s_{jk}^{*} is normalized too.

3.3 Joint Matrix Factorization

In order to learn the latent query feature space from user-query and query-URL bipartite graphs together, we fuse Eq. (2) and Eq. (3) together by sharing the same query feature space. Hence, we have

$$\begin{aligned}&\mathcal{H}(S,R,U,Q,L) = \\ &\frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{p} I_{jk}^{S}(s_{jk}^{*} - g(Q_j^T L_k))^2 + \frac{\alpha_r}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{R}(r_{ij}^{*} - g(U_i^T Q_j))^2 \\ &+ \frac{\alpha_u}{2}\|U\|_F^2 + \frac{\alpha_q}{2}\|Q\|_F^2 + \frac{\alpha_l}{2}\|L\|_F^2, \end{aligned} \quad (4)$$

where α_r is a small positive number to determine how much information need to be taken from user-query matrix. A local minimum of the objective function given by Eq.(4) can be found by performing simple gradient descent in U_i, Q_j and S_k,

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial U_i} &= \alpha_r \sum_{j=1}^{n} I_{ij}^{R} g'(U_i^T Q_j)(g(U_i^T Q_j) - r_{ij}^{*})Q_j + \alpha_u U_i, \\ \frac{\partial \mathcal{H}}{\partial Q_j} &= \sum_{k=1}^{p} I_{jk}^{S} g'(Q_j^T L_k)(g(Q_j^T L_k) - s_{jk}^{*})L_k \\ &+ \alpha_r \sum_{i=1}^{m} I_{ij}^{R} g'(U_i^T Q_j)(g(U_i^T Q_j) - r_{ij}^{*})U_i + \alpha_q Q_j, \\ \frac{\partial \mathcal{H}}{\partial L_k} &= \sum_{j=1}^{n} I_{jk}^{S} g'(Q_j^T L_k)(g(Q_j^T L_k) - s_{jk}^{*})Q_j + \alpha_l L_k, \end{aligned} \quad (5)$$

where $g'(x)$ is the derivative of logistic function $g'(x) = \exp(x)/(1+\exp(x))^2$. In order to reduce the model complexity, in all of the experiments we conduct in Section 5, we set $\alpha_u = \alpha_q = \alpha_l$. Moreover, in all the experiments we conduct in Section 5, when building the query graph, we use fixed 20 dimensions to represent the query feature space.

Although recently, similar factor analysis methods have been employed in [33, 34] for document retrieval and document classification, our approach has essential difference compared with these methods. Our method only fits the observed data, and treat unobserved data unknown or missing, while their methods fit unobserved data with zero, which will potentially distort the latent feature space. Moreover, since we only fit the observed data, our consideration can accelerate the computation time of the gradient descent, which will be analyzed in the next section.

3.4 Complexity Analysis

Although the feature learning part is the offline computation in our query suggestion framework, we still need to analyze the complexity of it to show that our approach scalable to very large datasets. The main computation of gradient methods is evaluating the object function $\mathcal{H}$ and its gradients against variables. Because of the sparsity of matrices R and S, the computational complexity of evaluating the object function $\mathcal{H}$ is $O(\rho_R d + \rho_S d)$, where ρ_R and ρ_S are the numbers of nonzero entries in matrices R and S, respectively. The computational complexities for gradients $\frac{\partial \mathcal{H}}{\partial U}$, $\frac{\partial \mathcal{H}}{\partial Q}$ and $\frac{\partial \mathcal{H}}{\partial L}$ in Eq. (5) are $O(\rho_R d)$, $O(\rho_R d + \rho_S d)$ and $O(\rho_S d)$, respectively. Therefore, the total computational complexity in one iteration is $O(\rho_R d + \rho_S d)$, which indicates that the computational time of our method is linear with respect to the number of observations in the two sparse matrices. This complexity analysis shows that our proposed approach is very efficient and can scale to very large datasets.

4. SIMILARITY PROPAGATION AND QUERY SUGGESTION

Let us first consider the following problem: given a query q which is issued by a user u, how can we recommend a set of latent semantically relevant queries to this user? A baseline method is to recommend to u the top 5 similar queries of q. This method is simple and seems natural, but since all the queries of clickthrough data are user-generated contents, they contain a lot of noise. Some people are good at formulating queries, but some are not. Simply recommending q's top similar queries has a high probability to introduce noise to u. To address this problem, in this section, we propose a novel similarity propagation and query suggestion model, which not only provides semantically relevant queries to users, but also ranks the results.

4.1 Similarity Propagation

Our similarity propagation model is modeled on heat diffusion. Heat diffusion is a natural physical phenomenon. In a medium, heat always flows from a position with high temperature to a position with low temperature. Recently, heat diffusion-based approaches have been successfully applied in various domains such as classification and dimensionality reduction problems [5, 18, 20]. Reference [20] approximated the heat kernel for a multinomial family in a closed form, from which great improvements were obtained over the use of Gaussian or linear kernels. In [18], Kondor et al. proposed the use of a discrete diffusion kernel for categorical data, and showed that the simple diffusion kernel on the hypercube can result in good performance for such data. Belkin et al. employed a heat kernel to construct the weight of a neighborhood graph, and apply it to a nonlinear dimensionality reduction algorithm in [5]. In [32], Yang et al. proposed a ranking algorithm known as the DiffusionRank using heat diffusion process; simulations showed that it is very robust to Web spamming.

The heat flows throughout a geometric manifold with initial conditions can be described by the following second order differential equation:

$$\begin{cases} \frac{\partial f(x,t)}{\partial t} - \Delta f(x,t) = 0, \\ f(x,0) = f_0(x), \end{cases} \quad (6)$$

where $f(x,t)$ is the temperature at location x at time t,

beginning with an initial distribution $f_0(x)$ at time zero, and Δf is the *Laplace-Beltrami operator* on a function f [20].

In this paper, we model similarity propagation as a process of heat diffusion. First, we construct the query similarity graph G based on the query feature space learned in Section 3. Consider a directed weighted graph $G = (V, E, W)$, where V is the vertex or query set, and $V = \{q_1, q_2, ..., q_n\}$. $E = \{(q_i, q_j)|$ there is an edge from q_i to q_j, and q_j is in the set of q_i's k nearest neighbors$\}$. The edge (q_i, q_j) is considered as a pipe that connects nodes q_i and q_j. $W = \{w_{ij}|$ similarity that associated with edge (q_i, q_j), or it can be interpreted as the probability that edge (q_i, q_j) exists$\}$. The value $f_i(t)$ describes the heat at node q_i at time t, beginning from an initial distribution of heat given by $f_i(0)$ at time zero. $\mathbf{f}(t)$ denotes the vector consisting of $f_i(t)$.

On a directed graph $G(V, E, W)$, in the pipe (q_i, q_j), heat flows only from q_i to q_j. Suppose at time t, each node q_i receives $RH = RH(i, j, t, \Delta t)$ amount of heat from q_j during a period of Δt. We have four assumptions: (1) RH should be proportional to the time period Δt; (2) RH should be proportional to the weight w_{ji} of the directed edge (q_j, q_i); (3) RH should be proportional to the heat at node q_j; and (4) RH is zero if there is no link from q_j to q_i. As a result, q_i will receive $\sum_{j:(q_j,q_i)\in E} \sigma_j w_{ji} f_j(t)\Delta t$ amount of heat from all its neighbors that points to it.

At the same time, node q_i diffuses $DH(i, t, \Delta t)$ amount of heat to its subsequent nodes. We assume that: (1) The heat $DH(i, t, \Delta t)$ should be proportional to the time period Δt; (2) The heat $DH(i, t, \Delta t)$ should be proportional to the heat at node q_i; (3) Each node has the same ability to diffuse heat; (4) The heat $DH(i, t, \Delta t)$ should be distributed to its subsequent nodes proportional to the weight on each edge. As a result, node q_i will diffuse $(\alpha f_i(t)\Delta t/d_i)\sum_{k:(q_i,q_k)\in E} w_{ik}$ amount of heat to its subsequent nodes, and each of its subsequent nodes q_k should receive $\alpha f_i(t) w_{ik}\Delta t/d_i$ amount of heat, where d_i is the outdegree of node i. Therefore $\sigma_j = \alpha/d_j$. In the case that the outdegree of node i equals zero, we assume that this node will not diffuse heat to others. To sum up, the heat difference at node q_i between time $t + \Delta t$ and t will be equal to the sum of the heat that it receives, less what it diffuses. This is formulated as

$$\frac{f_i(t+\Delta t) - f_i(t)}{\Delta t} = \alpha\left(-\frac{\tau_i}{d_i} f_i(t) \sum_{k:(q_i,q_k)\in E} w_{ik} + \sum_{j:(q_j,q_i)\in E} \frac{w_{ji}}{d_j} f_j(t)\right), \quad (7)$$

where τ_i is a flag to identify whether node i has any outlinks, such that $\tau_i = 0$ if node i does not have any outlinks, otherwise, $\tau_i = 1$. Solving Eq. (7), we obtain

$$\mathbf{f}(1) = e^{\alpha \mathbf{H}} \mathbf{f}(0), \quad (8)$$

where

$$H_{ij} = \begin{cases} w_{ji}/d_j, & (q_j, q_i) \in E, \\ -(\tau_i/d_i)\sum_{k:(i,k)\in E} w_{ik}, & i = j, \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

Moreover, $e^{\alpha t \mathbf{H}}$ could be extended as:

$$e^{\alpha t \mathbf{H}} = \mathbf{I} + \alpha t \mathbf{H} + \frac{\alpha^2 t^2}{2!}\mathbf{H}^2 + \frac{\alpha^3 t^3}{3!}\mathbf{H}^3 + \cdots. \quad (10)$$

The matrix $e^{\alpha t \mathbf{H}}$ is called the diffusion kernel in the sense that the heat diffusion process continues infinitely many times from the initial heat diffusion.

Parameter α plays an important role in the diffusion process. α is the thermal conductivity, i.e., the heat diffusion coefficient. If it has a high value, heat will diffuse very quickly. Otherwise, heat will diffuse slowly. In the extreme case, if it is infinitely large, then heat will diffuse from one node to other nodes immediately.

In fact, there are random relations among different queries even if these queries are unrelated in their literal meaning: people of different cultures, genders, ages, and environments, may implicitly link these queries together, but we do not know these latent relations. To capture these relations, we propose to add a uniform random relation among different queries. More specifically, let γ denote the probability that such phenomena happen, and $(1 - \gamma)$ is the probability of taking a "random jump". Without any prior knowledge, we set $\mathbf{g} = \frac{1}{n}\mathbf{1}$, where $\mathbf{g}$ is a uniform stochastic distribution vector, $\mathbf{1}$ is the vector of all ones, and n is the number of queries. Based on the above consideration, we modify our model to

$$\mathbf{f}(1) = e^{\alpha \mathbf{R}} \mathbf{f}(0), \quad \mathbf{R} = \gamma \mathbf{H} + (1 - \gamma)\mathbf{g}\mathbf{1}^T. \quad (11)$$

Following the setting of γ in PageRank [9, 21], we set $\gamma = 0.85$ in all of our experiments conducted in Section 5.

4.2 Query Suggestion

With the diffusion model proposed in the above section, we can now make query suggestions by the following three steps.

(1) First, for a given query q, we select a set of n queries $S = \{\hat{q}_1, \hat{q}_2, ..., \hat{q}_n\}$, each of which contains at least one word in common with query q, as the heat sources. Then we employ Eq. (12) to calculate the similarities between q and all of the queries in set S as the initial heat values.

$$f_{\hat{q}_i}(0) = \frac{|\mathcal{W}(q) \cap \mathcal{W}(\hat{q}_i)|}{|\mathcal{W}(q) \cup \mathcal{W}(\hat{q}_i)|}, \quad (12)$$

where $\mathcal{W}(q)$ is the set of all the words that query q contains. For an example, if a user submits the query "Sony", and suppose we have three previous queries containing "Sony": "Sony", "Sony Electronics", and "Sony Vaio Laptop". We treat these three queries as the heat sources, and the initial heat values are 1, 1/2, and 1/3, respectively.

(2) Then, we employ Eq. (11) to start the similarity propagation process, and calculate the value of each query in vector $\mathbf{f}(1)$.

(3) Finally, we sort the results of $\mathbf{f}(1)$ in decreasing order, and recommend the Top-N queries to the user who issued the search task.

4.3 Complexity Analysis

Since query suggestion is computed online, the computational complexity of suggestion algorithm should be as small as possible. Search engine users also do not have the patience to wait for suggestions for a long time. In this section, we will analyze the complexity of our proposed method, and introduce some very efficient techniques to reduce the complexity, and to ensure our algorithm is scalable for very large query similarity graphs.

When the graph of the query similarity graph is very large, a direct computation of $e^{\alpha \mathbf{H}}$ is very time-consuming. We

Table 2: Examples of LSQS Query Suggestion Results ($k = 50$)

Testing Queries	Suggestions				
	$\alpha = 10$			$\alpha = 1000$	
	Top 1	Top 2	Top 3	Top 4	Top 5
michael jordan	michael jordan shoes	michael jordan bio	pictures of michael jordan	nba playoff	nba standings
travel	travel insurance	abc travel	travel companions	hotel tickets	lowest air fare
java	sun java	java script	java search	sun microsystems inc	virtual machine
global services	ibm global services	global technical services	staffing services	temporary agency	manpower professional
walt disney land	world of disney	disney world orlando	disney world theme park	disneyland grand hotel	disneyland in california
intel	intel vs amd	amd vs intel	pentium d	pentium	centrino
job hunt	jobs in maryland	monster job	jobs in mississippi	work from home online	monster board
photography	photography classes	portrait photography	wedding photography	adobe elements	canon lens
internet explorer	ms internet explorer	internet explorer repair	internet explorer upgrade	microsoft com	security update
fitness	fitness magazine	lifestyles family fitness	fitness connection	womens health magazine	family fitness
m schumacher	schumacher	red bull racing	formula one racing	ferrari cars	formula one
solar system	solar system project	solar system facts	solar system planets	planet jupiter	mars facts
sunglasses	replica sunglasses	cheap sunglasses	discount sunglasses	safilo	marhon
search engine	audio search engine	best search engine	search engine optimization	song lyrics search	search by google
disease	grovers disease	liver disease	morgellons disease	colic in babies	oklahoma vital records
pizzahut	pizza hut menu	pizza coupons	pizza hut coupons	papa johns pizza coupon	papa johns
health care	health care proxy	universal health care	free health care	great west healthcare	uhc
flower delivery	global flower delivery	online florist	flowers online	send flowers	virtual flower
wedding	wedding guide	wedding reception ideas	wedding decoration	unity candle	centerpiece ideas
astronomy	astronomy magazine	astronomy pic of the day	star charts	space pictures	comet

adopt its discrete approximation to compute the heat diffusion equation:

$$\mathbf{f}(1) = \left(\mathbf{I} + \frac{\alpha}{P}\mathbf{R}\right)^P \mathbf{f}(0), \tag{13}$$

where P is a positive integer. In order to reduce the computational complexity, we introduce three techniques: (1) Since $\mathbf{f}(0)$ is a vector, we iteratively calculate $(\mathbf{I} + \frac{\alpha}{P}\mathbf{R})^P\mathbf{f}(0)$ by applying the operator $(\mathbf{I} + \frac{\alpha}{P}\mathbf{R})$ to $\mathbf{f}(0)$. (2) For matrix $\mathbf{R}$, we employ a data structure which only stores the information of non-zero entries, since it is a very sparse matrix. (3) For every heat source, we constrain it by only diffusing heat to its neighbors within three steps, which indicates that $P = 3$ in Eq. (13). This consideration is feasible since a query far from the heat sources would be less similar than the queries near the heat sources. Thus, since every query in query similarity graph G has less than k neighbors (see the definition of G in Section 4.1), the complexity of our query suggestion algorithm is $O(h \cdot k^3)$, where h is the number of heat sources. This shows that our algorithm is very efficient since $k \in [20, 50]$ can generally suggest queries with very good qualities, as shown in Section 5. We will show the impact of parameters k and P in Section 5.4 and Section 5.5, respectively.

5. EXPERIMENTAL EVALUATION

We conduct several experiments to measure the effectiveness and efficiency of our proposed query suggestion framework. In Section 5.1, we describe the statistics of the dataset we utilize. Section 5.2 shows the recommendation results generated by our online query suggestion algorithm of 20 sample queries. In Section 5.3, we design two measure methods to both manually and automatically evaluate the effectiveness of our algorithm. At the same time, we compare our Latent Semantic Query Suggestion (LSQS) method with the query suggestion method using SimRank [13]. Section 5.4 analyzes the impact of parameters k. Finally, Section 5.6 presents the empirical analysis of the efficiency of our online suggestion algorithm.

5.1 Data Collection

We construct our dataset based on the clickthrough data of *AOL* search engine [23]. In total, this dataset spans 3 months from 01 March, 2006 to 31 May, 2006. There are a total of 19,442,629 lines of clickthrough information, 657,426 unique user IDs, 4,802,520 unique queries, and 1,606,326 unique URLs.

This dataset is the raw data recorded by search engine, and contains a lot of noise which will potentially affect the effectiveness of our query suggestion algorithm. Hence, we conduct a similar method employed in [29] to clean the raw data. We clean the data by only keeping those frequent, well-formatted, English queries (queries which only contain characters 'a', 'b', ..., 'z', and space, and appear more than 3 times). After removing duplicates and cleaning, we get 19,2371 unique users, 224,165 unique queries and 343,302 unique URLs in our data collection in total. After the construction of two bipartite graphs using this data collection, we observe that a total of 5,220,660 edges exist in the user-query bipartite graph, which indicates that each user has at least issued 27.14 queries. we also observe that a total of 1,333,798 edges exist in the query-URL bipartite graph, which indicates that each query has 5.95 distinct clicks, and each URL is clicked by 3.89 distinct queries. Moreover, taken as a whole, this data collection has 69,937 unique words which appear in all the queries.

5.2 Examples of Query Suggestion Results

Before presenting the query suggestion results, let us first discuss some interesting characteristics of parameter α. As mentioned in Section 4.1, α is the thermal conductivity, and it plays an important role in the propagation process.

Following physical intuition, when α tends to infinity, the heat diffusion process will become stable, and does not depend on the initial temperature distribution, but only on the graph structure, the same as in PageRank. On the other hand, in the extreme case when $\alpha = 0$, then no heat will diffuse, and temperature distribution will remain exactly at the initial values.

Table 3: Comparisons between LSQS and SimRank

	Top 1	Top 2	Top 3	Top 4	Top 5
jaguar					
LSQS	jaguar cat	jaguar commercial	jaguar parts	jaguarundi	leopard
SimRank	american black bear	bottlenose dolphin	leopard	margay	jaguarundi
apple					
LSQS	apple computers	apple ipod	apple diet	apple vacations	apple bottom
SimRank	ipod troubleshooting	apple quicktime	apple ipods	apple computers	apple software

In the intermediate case, when α is small, the diffusion results will depend more on the initial temperature than the graph structure. In this case, queries suggested by our algorithm will have higher literal similarities with the initial query issued by a user. If α is relatively large, the results will depend more on the query graph structure, the the latent sematic relations hidden in the underlying query graph will be uncovered.

Hence, in our experiments, in order to recommend more latently similar queries to users, we combine the results generated from small α and large α together. Empirically, in our dataset, we set the small α to 10 and the relatively large α to 1000. For every test query, we conduct the similarity propagation (i.e, heat diffusion) process twice, the first using $\alpha = 10$ and another using $\alpha = 1000$. Then we combine the top 3 queries from the first diffusion process and the top 2 queries from the second diffusion process together, as the final top 5 suggestion results which will be recommended to users.

In total, we create a set of 50 queries as the testing queries, covering a wide range of topics, such as Computers, Arts, Business, and others. The recommendation results are shown in Table 2. All the results in this table are generated based on the query similarity graph built using parameter $k = 50$, which indicates that in this directed graph, the outdegree of each query is less than or equal to 50. Due to space limitation, we only list the results of 20 testing queries generated by our query suggestion method. All the queries are converted to lowercase.

From the results, we observe that our suggestion algorithm not only suggests queries which are literally similar to the test queries, but also provides latent semantically relevant recommendations. For instance, as to the results using $\alpha = 1000$, if the test query is a company, such as "intel", we suggest "pentium" and "centrino", which are two most successful sub-brands of semiconductor company *Intel*. If the test query is a technique, such as "java", we recommend "sun microsystems inc" and "virtual machine". The former suggestion is the company who owns the *Java Platform*, and the latter suggestion is a key feature of the *Java* programming language. They both have high latent semantic relations to query "java". If the test query is a human name, such as "m schumacher", the most successful *Formula 1* driver, the latent semantic suggestions are "ferrari cars" and "formula one". All of the results show that our latent semantic query suggestion algorithm has a promising future.

5.3 Evaluation of Suggestion Results

In this section, we first compare our Latent Semantic Query Suggestion (LSQS) method with the approach using SimRank [13]. Then we employ two different metrics to evaluate these two methods.

Table 4: Accuracy Comparisons

Accuracy	**LSQS**	**SimRank**
By Experts	0.8413	0.7101
By ODP	0.6823	0.5789

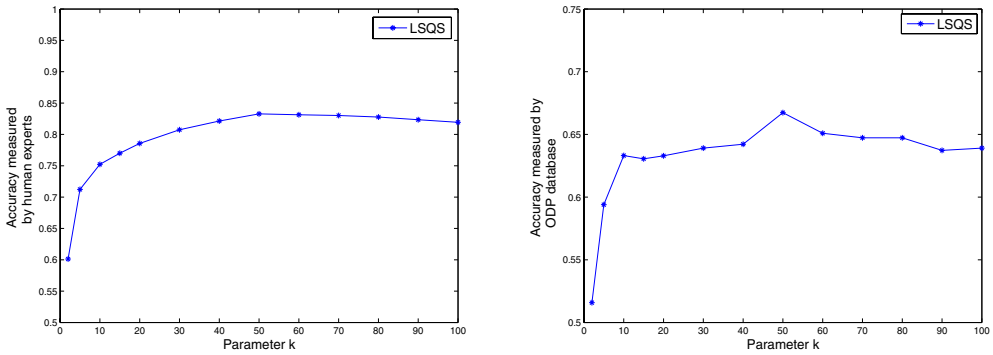
In the method of SimRank, we use the query-URL bipartite graph to calculate the similarities between queries. Then based on the similarities, recommend the top-5 similar queries to users. SimRank based on the intuition that two queries are very similar if they link to a lot of similar URLs. On the other hand, two URLs are very similar if they are clicked as a result of several similar queries. Based on this intuition, in SimRank, we first calculate the similarities between URLs, then we compute the similarities for queries based on the similarities of URLs. We iteratively update the similarities until they converge.

In Table 3, we show the query suggestions of LSQS and SimRank using two ambiguous word "jaguar" and "apple". We can observe that when using "jaguar" as the keyword, our LSQS algorithm can basically suggest more diverse and relevant queries, while all the suggestions using SimRank are different creatures. "apple" is another example to show the suggestion results. SimRank only suggest queries that related to the "apple company", while our LSQS can suggest more kinds of queries related to "apple".

Evaluating the quality of semantic relations is difficult, in particular for the contents that generated by users, as there are no linguistic resources available. In this paper, we conduct both a manual evaluation by a panel of three human experts, and automatic evaluation based on the ODP database.

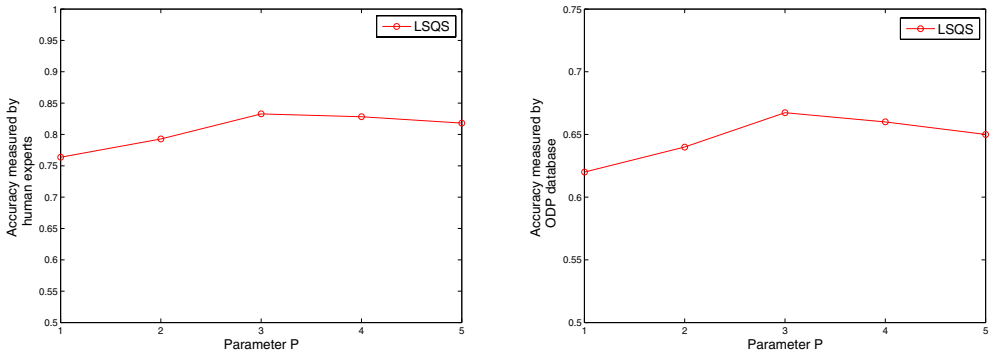
In the evaluation by human experts, we ask all the experts to rate the query suggestion results (we use the same 50 testing queries adopted in Section 5.2). We define a 6-point scale (0, 0.2, 0.4, 0.6, 0.8, and 1) to measure the relevance between the testing queries and the suggested queries, in which 0 means "totally irrelevant" while 1 indicates "entirely relevant". The average values of evaluation results are shown in Table 4. We observe that, when measuring the results by human experts, our LSQS algorithm increases the accuracy for about 18.47% than the SimRank algorithm.

For the automatic evaluation, we utilize the ODP database. ODP, also known as dmoz, is the largest, most comprehensive human-edited directory of the Web. In this paper, we adopt the same method used in [2] to evaluate the quality of suggested queries. When a user types a query in ODP, besides site matches, we can also find *categories* matches in the form of paths between directories. Moreover, these categories are ordered by relevance. For instance, the query "Java" would provide the hierarchical category



(a) Evaluation by Experts (b) Evaluation by ODP Database

Figure 2: Impact of Parameter k ($P = 3$)



(a) Evaluation by Experts (b) Evaluation by ODP Database

Figure 3: Impact of Parameter P ($k = 50$)

"Computers : Programming : Languages : Java", where ":" is used to separate different categories. One of the results for "Virtual Machine" would be "Computers : Programming : Languages : Java : Implementations". Hence, to measure how related two queries are, we can use a notion of similarity between the corresponding categories (as provided by ODP). In particular, we measure the similarity between two categories $\mathcal{D}$ and $\mathcal{D}'$ as the length of their longest common prefix $\mathcal{F}(\mathcal{D}, \mathcal{D}')$ divided by the length of the longest path between $\mathcal{D}$ and $\mathcal{D}'$. More precisely, denoting the length of a path with $|\mathcal{D}|$, this similarity is defined as: $Sim(\mathcal{D}, \mathcal{D}') = |\mathcal{F}(\mathcal{D}, \mathcal{D}')|/max\{|\mathcal{D}|, |\mathcal{D}'|\}$. For instance, the similarity between the two queries above is 4/5 since they share the path "Computers : Programming : Languages : Java" and the longest one is made of five directories. We have evaluated the similarity between two queries by measuring the similarity between the most similar categories of the two queries, among the top 5 answers provided by ODP.

As shown in Table 4, we observe that, when evaluating using ODP database, our proposed LSQS algorithm increases the suggestion accuracy for about 17.86% than the SimRank algorithm. This indicates that our proposed query suggestion algorithm is very effective.

5.4 Impact of Parameter k

The parameter k defines the maximal outdegree of each query in the query similarity graph, and it performs as an important role in terms of both effectiveness and efficiency.

Figure 2(a) shows the impact of parameter k on the accuracy measured by human experts. The X-axis is parameter k, while the Y-axis is the accuracy measured by human experts. As increasing the number of k, the accuracy of LSQS raises, until $k = 50$. If $k > 50$, the performance drops a little bit. This is because when $k > 50$, and especially when $k = 100$, the possibility that noisy queries are becoming added into the query similarity graph is increased, which will potentially affect the quality of query suggestion results. Nevertheless, it still shows very good suggestion quality.

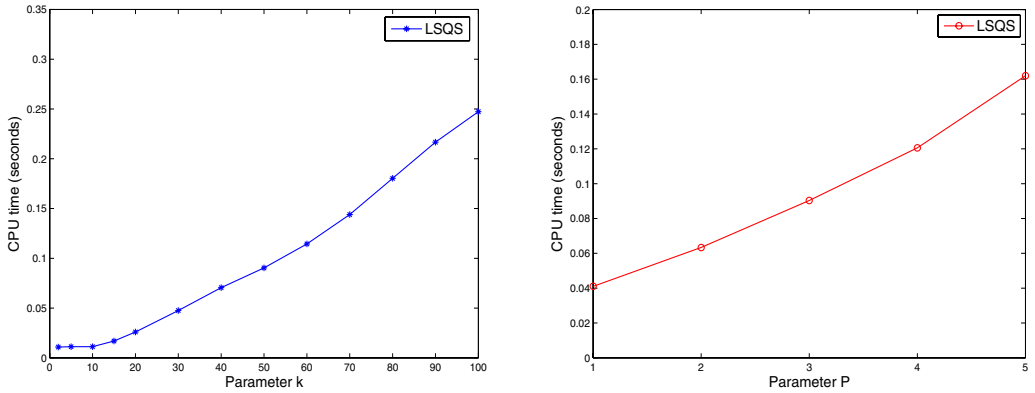
The evaluation results measured using ODP database is shown in Figure 2(b). It generally has the same trend as that observed in Figure 2(a). The main difference is that the accuracy scores using ODP database are generally smaller than the scores rated by the human experts. The reason is straightforward: human experts have a better understanding of the latent semantic similarities between two queries than does the ODP measure method.

5.5 Impact of Parameter P

The parameter P indicates how far the heat diffuses. From Figure 3, we observe that when $P = 3$, our algorithm achieves the best performance, and then the suggestion quality decreases as the increase of P. This phenomenon is consistent with the intuition that a query far from the heat sources would be less similar with the original queries.

5.6 Evaluation of Efficiency

Efficiency is a very crucial measurement for evaluating online algorithms, especially for online query suggestion algorithm. If a query suggestion algorithm is not fast, then no



(a) $P = 3$ (b) $k = 50$

Figure 4: Efficiency Analysis

one would use it any more. We use the same testing set with previous experiments containing 50 testing queries to evaluate the efficiency of our proposed algorithm. The number of diffusion sources for each testing query scales from only a few to several hundreds. We record the average computational time for online suggestions. The computational time includes the similarity propagation time and the ranking time after the propagation. All the experiments are computed by a personal computer consisting of an *Intel Pentium D* CPU (3.0 GHz, *Dual Core*) and 1 *Giga* memory. The results are shown in Figure 4. We observe that when $k = 10$, the average computational time is less than 0.01 second, and this number only increases to 0.09 second if $k = 50$. Since $k = 50$ is the best parameter setting in terms of suggestion quality in our data collection, we can draw the conclusion that our algorithm is efficient.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a novel query suggestion framework incorporating two parts: an offline computation part and an online computation part. The offline computation part employs a novel joint matrix factorization method using user-query and query-URL bipartite graphs. The online part is implemented by a similarity propagation process modeled on the heat diffusion process, and can recommend both literally similar queries and latent semantically relevant queries to search engine users. The simulation results show that our proposed query suggestion framework is both effective and efficient.

The user IDs in clickthrough data are only used for the matrix factorization part. Actually, user ID information is very useful for improving the recommendation qualities, since some users can expertly formulate queries, while some are not experts. Hence, in the future, we plan to incorporate the rank information of all the users in the clickthrough data based on link information.

In Section 4.2, Eq. (12) is employed to calculate the initial similarity scores for the diffusion sources. Different calculation methods will definitely generate different query recommendation results. This is also a problem that is worth investigating.

Finally, due to the fact that question-answering services are becoming popular, we plan to develop a similar method to that proposed in this paper to match the question issued by a user to the most relevant question previously answered by human experts.

7. ACKNOWLEDGMENTS

The authors appreciate the anonymous reviewers for their extensive and informative comments for the improvement of this paper. The work described in this paper was fully supported by two grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK4150/07E and GRF #412507).

8. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26, New York, NY, USA, 2006. ACM.

[2] R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 76–85, New York, NY, USA, 2007. ACM.

[3] R. A. Baeza-Yates, C. A. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *EDBT Workshops*, pages 588–596, 2004.

[4] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416, New York, NY, USA, 2000. ACM.

[5] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[6] P. A. Chirita, C. S. Firan, and W. Nejdl. Personalized query expansion for the web. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 7–14, New York, NY, USA, 2007. ACM.

[7] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Query expansion by mining user logs. *IEEE Trans. Knowl. Data Eng.*, 15(4):829–839, 2003.

[8] G. Dupret and M. Mendoza. Automatic query recommendation using click-through data. In *IFIP PPAI*, pages 303–312, 2006.

[9] N. Eiron, K. S. McCurley, and J. A. Tomlin. Ranking the web frontier. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 309–318, New York, NY, USA, 2004. ACM.

[10] W. Gao, C. Niu, J.-Y. Nie, M. Zhou, J. Hu, K.-F. Wong, and H.-W. Hon. Cross-lingual query suggestion using query logs of different languages. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 463–470, New York, NY, USA, 2007. ACM.

[11] D. Gleich and L. Zhukov. Svd subspace projections for term suggestion ranking and clustering. In *Technical Report of Yahoo! Research Labs*, 2004.

[12] B. J. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real life information retrieval: A study of user queries on the web. *SIGIR Forum*, 32(1):5–17, 1998.

[13] G. Jeh and J. Widom. SimRank: a measure of structural-context similarity. In *KDD '02: Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543, New York, NY, USA, 2002. ACM.

[14] J. Jeon, W. B. Croft, and J. H. Lee. Finding similar questions in large question and answer archives. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 84–90, New York, NY, USA, 2005. ACM.

[15] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM.

[16] T. Joachims and F. Radlinski. Search engines that learn from implicit feedback. *Computer*, 40(8):34–40, 2007.

[17] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 387–396, New York, NY, USA, 2006. ACM.

[18] R. I. Kondor and J. D. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML '02: Proceedings of the 19th International Conference on Machine Learning*, pages 315–322, 2002.

[19] R. Kraft and J. Zien. Mining anchor text for query refinement. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 666–674, New York, NY, USA, 2004. ACM.

[20] J. D. Lafferty and G. Lebanon. Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research*, 6:129–163, 2005.

[21] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. In *Technical Report Paper SIDL-WP-1999-0120 (version of 11/11/1999)*, 1999.

[22] M. Pasca and B. V. Durme. What you seek is what you get: Extraction of class attributes from query logs. In *IJCAI '07: Proceedings of International Joint Conferences on Artificial Intelligence*, pages 2832–2837, 2007.

[23] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *The First International Conference on Scalable Information Systems*, Hong Kong, June, 2006.

[24] D. Shen, M. Qin, W. Chen, Q. Yang, and Z. Chen. Mining web query hierarchies from clickthrough data. In *AAAI '07: Proceedings of the 22th Conference on Artificial Intelligence*, pages 341–346, 2007.

[25] C. Silverstein, M. R. Henzinger, H. Marais, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

[26] J.-T. Sun, D. Shen, H.-J. Zeng, Q. Yang, Y. Lu, and Z. Chen. Web-page summarization using clickthrough data. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 194–201, New York, NY, USA, 2005. ACM.

[27] M. Theobald, R. Schenkel, and G. Weikum. Efficient and self-tuning incremental query expansion for top-k query processing. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 242–249, New York, NY, USA, 2005. ACM.

[28] B. Vélez, R. Weiss, M. A. Sheldon, and D. K. Gifford. Fast and effective query refinement. *SIGIR Forum*, 31(SI):6–15.

[29] X. Wang and C. Zhai. Learn from web search logs to organize search results. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 87–94, New York, NY, USA, 2007. ACM.

[30] J.-R. Wen, J.-Y. Nie, and H. Zhang. Query clustering using user logs. *ACM Trans. Inf. Syst.*, 20(1):59–81, 2002.

[31] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 4–11, New York, NY, USA, 1996. ACM.

[32] H. Yang, I. King, and M. R. Lyu. DiffusionRank: a possible penicillin for web spamming. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 431–438, New York, NY, USA, 2007. ACM.

[33] D. Zhou, S. Zhu, K. Yu, X. Song, B. L. Tseng, H. Zha, and C. L. Giles. Learning multiple graphs for document recommendations. In *WWW '08: Proceedings of the 17th international conference on World Wide Web*, pages 141–150, New York, NY, USA, 2008. ACM.

[34] S. Zhu, K. Yu, Y. Chi, and Y. Gong. Combining content and link for classification using matrix factorization. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 487–494, New York, NY, USA, 2007. ACM.