Transactions on Machine Learning
and Data Mining
Vol. 9, No. 1 (2016) 3-26
ISSN 1864-9734
ISBN: 879-3-942952-41-5

ibai Publishing
www.ibai-publishing.org

Query Recommendation based on Query Relevance Graph

Sejal D[1], Shailesh K G[1], Tejaswi V[2], Dinesh Anvekar[3], Venugopal K R[1], S S Iyengar[4], and L M Patnaik[5]

[1]Department of Computer Science and Engineering
University Visvesvaraya College of Engineering, Bangalore University, Bangalore-1
sej_nim@yahoo.co.in
[2]National Institute of Technology, Surathkal, Karnataka
[3]Alpha College of Engineering, Bangalore
[4]Florida International University, USA
[5]Indian Institute of Science, Bangalore, India

Abstract. With the explosive and diverse growth of web contents, query recommendation is a critical aspect of the search engine. Different kind of recommendation like query, image, movie, music and book etc. are used every day. Different kinds of data are used for the recommendations. If we model the data into various kinds of graphs then we can build a general method for any recommendation. This paper presents a general method to recommend queries by combining two graphs: 1) query click graph which uses the knowledge of link between user input query and clicked URLs and 2) query text similarity graph which finds the similarity between two queries using Jaccard similarity. The proposed method provides literally as well as semantically relevant queries for users' need. Experiment results show that the proposed algorithm outperforms heat diffusion method by providing more number of relevant queries. It is also useful to other recommendations like image, query and product recommendation.

Keywords: Image Recommendation, Query Recommendation, Query Relevance, Suggestion.

1 Introduction

In the last decade, web has grown tremendously which results in becoming highest publicly available data source in the world. Web mining intents to determine valuable

knowledge and information from logs, web page contents and URLs. Web mining can be classified into three types of mining process based on data usage : Web content mining, web usage mining and web structure mining. Web content mining finds valuable knowledge and information from contents of web pages. Web usage mining uses users' navigation history to extract user access patterns. Web structure mining determines information from link of web pages.

Exponential gain of web information is a challenging task for the search engines to meet the users requirement. Handling web information appropriately and organising adequately is more demanding on the web. To get any information from web, the user issue queries, follows some links in web snippets, clicks on advertisement, and spends some time on pages. The user reformulates his query, if he is not convinced with clicked page information. In order to enhance the user experience, search engine provides various kind of query recommendation.

On the Web, query recommendation is a technique to recommend related queries to users' input query by finding association between queries from users' search log. The simple way to recommend a query is spelling correction. In this paper, our interest is to recommend more elaborate forms of queries. For example, if the user submits a query *java*, then the user may be prompted to other queries like *java for windows, java 32 bit* or *java download*, and also a related concept like *sun micro system*.

Basically, query recommendation on a commercial site like flipkart.com, Myntra.com etc., recommends products based on collaborative filtering [1], [2]. Collaborative filtering is a method to predict users' interests by collecting rating information or preferences from many users. Therefore, collaborative filtering algorithm needs to build product-user matrix which determines relationship between user preferences to that product. The constraint with this approach is that the rating data are not present. However, on the web, search log is always available to us. It retrieves users' search information on the web and how they rephrase their query.

Motivation: There are several challenges to design recommendation framework on the web. First, usually users' submit short queries between one and three terms and are generally ambiguous. We observe from America On-Line (AOL) [3] data that 9.82% of web queries contain one term, 27.31% of web queries contain two terms, and 26.99% of web queries contain three terms. Fig. 1 represents the query length distribution based on the number of words. Second, in most of the cases, the users do not have sufficient knowledge the topic that are searched, and they are not able to clearly phrase the query words. Then, users have to rephrase the query words and rephrase their queries frequently. Hence, it is necessary to solve query recommendation to satisfy users' information need and to increase the search engine usability. Various kind of data are used for suggestions and these data can be converted to graphs and can be used to solve many suggestions problems by designing a generic graph recommendation approach.

Contribution: This paper presents a generic method for the query recommendation on the web by using query relevance directed graph generated from a search log. Query click graph is constructed by capturing the relationship between queries frequently clicked on common URLs. Then, query text similarity graph is constructed using Jaccard similarity between queries. Finally, query click graph and query text similarity graph is combined to construct query relevance graph and has several advantages.

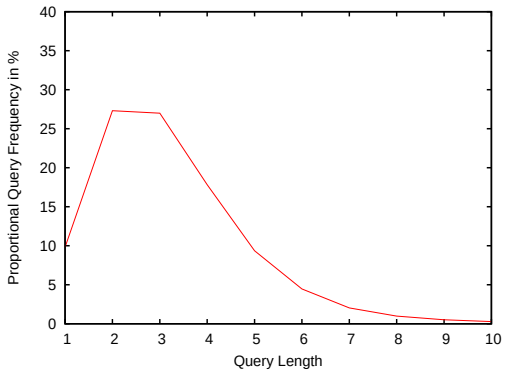


Fig. 1: Distribution of Query Length

This generic method can be used for query, image and product recommendation. It provides semantically relevant queries to meet the original users' need and is scalable to the large dataset.
Organization: This paper is organized as follows: Query recommendation techniques are reviewed under section 2. Section 3 describes the background work while section 4 presents query relevance model and algorithm. Section 5 discusses experiment results, query recommendation results comparison, efficiency analysis and image recommendation. Finally, conclusion is presented in section 6.

2 RELATED WORK

It is a difficult task for any search engine to interpret users' search intention. Various techniques have been studied extensively in the past decade to improve performance and quality of query recommendations. In this section, we have reviewed several papers related to query expansion methods, query recommendation based on various graphs and with snippets and various recommendation applications.

2.1 Query Expansion

Query expansion is a method to reformulate original query by providing synonyms terms and searching, suggesting correlated terms by fixing spelling error. In [4] Kunpeng et al.,proposed a method for query expansion by using user interaction information from search log rather than extracting terms from relevant documents. Correlations among queries are extracted by determining frequently selected documents by a user. Then, expansion terms are selected by analyzing relationship of documents. This method performs better than local context analysis, an effective method of query expansion and provides query expansion for ambiguous queries efficiently than for clear

queries. It fails to retrieve topically relevant documents. Latent Dirichlet Allocation (LDA) model is used to get topic information and learn hidden topics from the documents and finds out a term conditional probability on topics [5]. This term is used as a suggestion that is topically consistent with input query. Adam [6] suggested probable path to extend search once user executes a query rather than suggesting terms. A clustering-by-directions (CBD) algorithm for interactive query expansion is used to choose various paths.

Song et al.,[7] proposed a method for query expansion by constructing term-transition graph from search log. It finds user preferences within each topic by random walk. Relevant queries are suggested by term transition probability and topic distribution in each topic for given user input query. This model has constraint of considering only one term from the queries. Goyal et al., [8] provided generalized framework for query expansion. Lexical association between terms is used to derive global query expansion model. System relevance criteria are used to derive lexical association function. Hence, it is applicable to the Vector Space Model, Best Matching 25, and Language Model scenarios.

2.2 Snippet based Query Recommendation

Snippet is the brief description of the document which helps users to click that document. Personalized query suggestion methods are proposed based on query-concept bipartite graph [9],[10]. Web snippets of the query results are used to extract concepts for user input query to identify related queries [9]. While concepts are obtained by calculating occurrence of frequencies in the users selected web pages [10]. Concept clusters are generated by agglomerative clustering in both these methods. A similar P-QC using Beeferman and Bergers agglomerative algorithm is used where users' personal preferences for input query is captured from user profile to provide personalize query suggestion [11]. Web-snippets are collected from clicked document. Euclidean distance between every interval of query has been calculated and dataset is divided into clusters by a Decision Tree and merging similar query clusters.

Liu et al., [12] constructed Term Frequency (TF) based recommendation algorithm that analyzes snippet click behavior model which recognize users information need with their search behaviors rather than discovering most related terms from user previous queries. Keywords are extracted by TF-based method from snippet and then recommendation keywords are generated by local and global snippet click model. Snippets are used to expand original Thai language user queries to a new set of more meaningful queries in the same language [13]. Thai word segmentation is developed on corpus-based approach to extract all terms in titles. Genetic algorithm as in [14–18] is applied on the remaining words and new expanded terms are displayed to the user. A query suggestion method with phrasal concept for literature search is proposed in [19]. Pseudo-relevant top-k documents are generated for baseline query and used to extract noun phrase as candidate concept.

In [9]-[19] authors have used snippet information of clicked URLs or search results returned from a query for query recommendation in different ways. These methods are

not general and the extensibility is very low.

2.3 Graph based Query Recommendation

In graph based query recommendations query-URLs clicked information is used to construct query-URL bipartite graph for query recommendation. Kruschwitz et al., [20] presented various query modification methods analysis based on search log and non-log methods. The log based approaches significantly outperform non-log based approaches. Query Flow Graphs provides more number of recommendations for less frequent queries. Markov random walk model [21] is applied on bipartite graph to achieve a probabilistic documents ranking for a given query from a large click log. Further, query suggestion algorithm [22],[23] is proposed by calculating hitting time on bipartite graph to rank queries. Subgraph [22] is constructed for given input query by using Depth First Search and random walk resulting in computation of hitting time. The top-k queries with smallest hitting time is displayed as output. While suggestions are derived by computing hitting time by Markov random walk on query-URL bipartite graph [23]. These algorithms outperforms forward and backward random model. The advantage of these algorithms is that it recommends semantically relevant and log tails queries to input queries. But sometimes, it recommends log tails queries as top queries in recommendation results.

Anagnostopoulos et al., [24] have developed a framework which models the querying behavior of users for query recommendation by a query flow graph. A series of queries provided by a user can be obtained as path for the graph. Markov chain random walk is applied on this graph. As the user clicked information is not considered to create a graph, this approach results in lower accuracy. Query suggestion method [25] is presented by considering immediately preceding queries in query log as context. Click through bipartite graph is used to clusters concepts to compile queries. This approach outperforms the two baseline methods adjacency and N-gram. Adjacency method arranges queries based on frequencies in the training sessions and provides top queries as suggestions. N-gram method arranges queries based on frequencies of immediately following queries in a given sequence of queries and provides top queries as suggestions.

Queries are suggested based on query-URL-tag tripartite graph obtained from query-URL and URL-tag bipartite graph through URL [26]. A query URL bipartite graph is constructed from search log and URL tag bipartite graph is generated from social annotation data. The recommendations are generated by applying random walk on a graph and calculating the hitting time. Random walk with restart algorithm [27] is applied to generate query suggestion candidate on query log. The suggested queries are re-ranked based on diversification function values. This method computes dissimilarity among search results and suggestion queries. This results in significant improvement in query suggestion as compared to the random walk model. Candidate suggestions structured representation is generated from related query by using common clicks and common sessions [28]. The quality of suggestions is identified by proposed different query independent features such as popularity of suggestion terms in number of sessions and clicks, length of the suggestion tokens and characters. Vahabi et al., [29] have presented a query suggestion algorithm which identifies related queries that do not have

any common terms in users input query. This method does not require any training, computationally efficient and accommodate to any search engine efficiently.

In the above query based recommendation algorithms [20]-[29], the authors have used query-URLs clicked information to construct query-URL bipartite graph for query recommendation in various ways. There are two major problems with query-URL based query recommendation : 1) the common clicks on documents are limited for various queries. 2) The two queries may be irrelevant though they share common URLs because the content of that URLs may be totally different. These methods ignore the rich information between two queries relevance. Our approach in this work is to consider relevance information between two queries to enhance user recommendation.

2.4 Various Recommendation Applications

Sarwat et al., [30] have proposed a location aware recommender system(LARS). It produces recommendation by location based rating i.e., closer to travel distance to querying users. It avoids exhaustive access to all spatial items.

Cao et al., [31] have designed user friendly patent search prototype to enhance search experience and discover relevant patents. It uses topic-based query suggestion, error correction and query expansion techniques to make patent search more user friendly. Topic-based query suggestion provides meaningful keywords when users type query keywords. Query expansion suggests relevant or synonyms keywords of user input query keywords.

McFee et al., [32] have proposed an approach for music recommendation. It improves content-based audio similarity from collaborative trained data. Item similarity is derived from the sample of collaborative data and an optimal distance metric over audio descriptors is trained. When collaborative filter data is not available then distance metric is applied to the data.

Gao et al., [33] have presented query suggestion method which suggests relevant queries in different language when input query is from another language. Support Vector Machine (SVM) regression algorithm is used to learn cross-lingual term similarity function. They have tested results with respect to French to English queries. Regression error is measured with Mean Square Error. Training set in SVM requires more data thus reducing its speed.

Table 1 shows comparison of closely related works with our proposed method.

3 BACKGROUND

Ma et al. [34] have presented query suggestion algorithm by using heat diffusion method on directed query URL bipartite graph. An undirected bipartite graph is considered where, B_{ql} = (V_{ql}, E_{ql}); 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_i,l_j) exists if a URL l_k is clicked by user for a query q_j. The weight on the edges is calculated by number of times a query is clicked on a URL.

This undirected bipartite graph does not correctly measures the relationship of URLS

Table 1: Related Work Comparison

Author	Concept	Performance	Advantages/Disadvantages
Leung et al.,[9]	Personalized query suggestion based on concept based clustering extracted from web snippets of search results	Obtains best F-measures value compared to Query-URL and Query Word methods	Obtains personalized query suggestions for individual users It is not a general method and extensibility is very low
Mei et al.,[22]	Query suggestion by calculating hitting time on query-URL bipartite graph	Provides suggestions efficiently by using a smaller sub-graph and a few iterations	Suggests some long tail infrequent queries Sometimes infrequent queries are ranked higher in results
Santos et al.,[28]	Proposed a ranking approach for producing effective query suggestion	It gives efficient suggestions with smaller query logs in comparison with commercial search engine logs	Suggestions are generated from related query with common clicks and common sessions which helps to overcome data sparsity for long-tail queries Computation time is more
Ma et al.,[34]	Query suggestion based on heat diffusion method on directed query-URL bipartite graph	Computation time for the query suggestion is around 0.10 seconds	Generates semantically related queries to inputs The common clicks on documents are limited for various queries
Our Work	Query recommendation based on depth first search method on query relevance graph	Recommends more number of queries than heat diffusion method [34] with same computation time	Provides more number of relevant queries to inputs

and queries. Hence, they are converted into directed query-URL bipartite graph in which each undirected edge is transformed to two directed edges. Query URL and URL Query edge weight is normalized by the total number of times query is used and total number of times the document is clicked by user respectively. Query suggestion algorithm is applied on the converted graph.

A converted bipartite graph G = (V ∪ U , E), where V and U represents query and URL set. Depth first search algorithm is used to construct sub graph on G, for given input query *q* in V. Heat diffusion process is applied on the sub-graph. Top-*k* queries are suggested with largest heat value. This method outperforms Forward random walk [35], SimRank [36] and backward random walk [35].

Hwang et al. [37] has developed online query grouping method by generating graph which combines the connection of queries regularly used by users and clicked on similar documents. Related query clusters are generated by Monte Carlo random walk simulation method for a given query.

4 QUERY RELEVANCE MODEL AND ALGORITHM

4.1 Problem Definition

Given a user input query *q* and *search log* of search engine, we convert search log into a graph, where nodes represent queries and edges represent relationship between queries. The objective is to provide semantically relevant query recommendation to meet the original users' need.

4.2 Assumptions

It is assumed that the user is online and enters input query with less than six terms.

4.3 Query Relevance Model

The query relevance model captures the relevant queries from user search log. This model constructs query relevance graph by combining query click graph which uses the knowledge of (i) queries frequently clicked on common URLs and (ii) query text similarity graph using Jaccard similarity between queries.

Query Click Graph Relevant queries can be obtained by considering those queries that are clicked on same set of documents by the users in the search logs. For example, the queries *solar system* and *planet* are not textually similar, but they are relevant. This information can be achieved by analysing common clicked URLs on queries in the search log.

Consider a URL-Query undirected bipartite graph, $BG_{qu} = (V_{qu}, E_{qu})$, where V_{qu} = Q ∪ U, Q = $\{q_1,q_2,...q_m\}$ and U=$\{u_1,u_2,...u_n\}$. E_{qu} is the set of all edges. The edge (q_i,u_j) exists if and only if user has clicked a URL u_j after issuing query q_i. Often, the

user issues query and by mistake clicks on some URL, which has no relation. In order to reduce noise and outliers, those edges which have only one click between query and URL are removed. Fig. 2 illustrates an example of URL-Query bipartite graph.

From BG_{qu} Query Click directed Graph, QC=(V_q,E) is constructed, where V_q are queries and E is a directed edge from q_i to q_j which exists if and only if there is at-least one common URL u_k, that both q_i and q_j link in BG_{qu}.

The weight of edge (q_i,q_j) in QC, $w_c(q_i,q_j)$ is calculated by counting number of

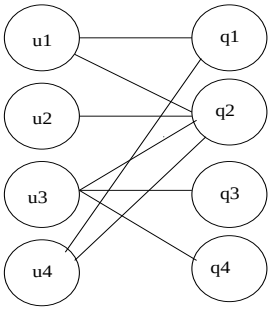


Fig. 2: URL-Query Bipartite Graph

common URLs. Fig. 3 shows Query click graph generated from Fig. 2. Here, q_1 and q_2 has two common URLs u_1 and u_4. Hence, the weight of (q_1,q_2) is 2 and vice versa.

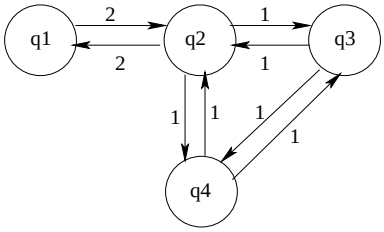


Fig. 3: Query Click Graph

Query Text Similarity Graph It is not necessary for the user to always click the same URL for different queries. In such a case, we may not get two or more queries having common URL, but queries may be relevant since they share common words. For example, queries *cloud computing* and *cloud computing books* shares common words. To get these relevant queries, query text similarity graph is constructed.

For this graph, query text similarity by Jaccard coefficient J_c is calculated. Jaccard coefficient is defined as the fraction of common words between two queries as given in

equation 1.

$$J_c(q_i, q_j) = \frac{words(q_i) \cap words(q_j)}{words(q_i) \cup words(q_j)} \quad (1)$$

Query Text Similarity directed graph is defined as QG_{ts}=(V_q,E), where V_q is distinct queries in search log and E an edge between q_i to q_j exists if $J_c(q_i,q_j)>0.6$. The weight of edge (q_i,q_j) in QG_{ts}, $w_{ts}(q_i,q_j)$ is calculated by counting the occurrence of q_j in the search log followed by q_i. Fig. 4 shows the example of Query Text Similarity Graph.

In Fig. 4, q_1, q_2 and q_5 are distinct queries in search log. The Jaccard value of q_1 and

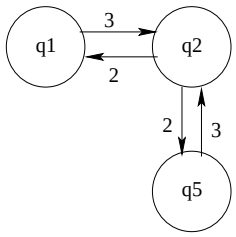


Fig. 4: Query Text Similarity Graph

q_2, is assumed to be greater than 0.6, therefore q_1 and q_2 are included as nodes in the graph. The weight of q_1 to q_2 is 3 as occurrence of q_2 is 3 followed by q_1 in the search log and weight of q_2 to q_1 is 2 as occurrence of q_1 is 2 followed by q_2 in the search log. Similarly, Jaccard value of q_2 and q_5 is greater than 0.6. The weight between q_2 and q_5 is calculated using the same procedure as q_1 and q_2.

Query Relevance Graph The query click graph *QC* and query text similarity graph QG_{ts} captures two useful characteristics of relevant queries. To utilize both the characteristics, the query click graph and query text similarity graph are combined into a single graph, Query Relevance Graph QRG = (V_q,E), where V_q is set of queries either from *QC* or QG_{ts} and E an edge between q_i to q_j exists either from *QC* or QG_{ts}. The weight of edge (q_i,q_j) in QRG is calculated as follows : $w_r(q_i,q_j) = w_c(q_i,q_j) + w_{ts}(q_i,q_j)$. Fig. 5 represents query relevance graph from the combined graphs of Fig. 3 and Fig. 4.

The weight of edge (q_i,q_j) in Query Relevance Graph is normalized by equation 2. The normalized graph is shown in Fig. 6.

$$Normalized w_r(q_i, q_j) = \frac{w_r(q_i, q_j)}{\sum_{(q_i,q_n)\in E} w_r(q_i, q_n)} \quad (2)$$

4.4 Query Recommendation Algorithm

The QRGQR-Query Relevance Graph for Query Recommendation algorithm is shown as Algorithm 1. Given a search log of search engine, query relevance graph is con-

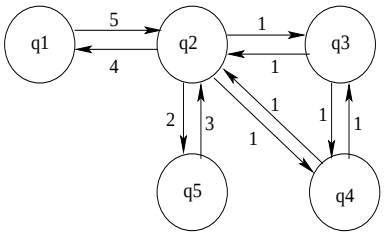


Fig. 5: Query Relevance Graph

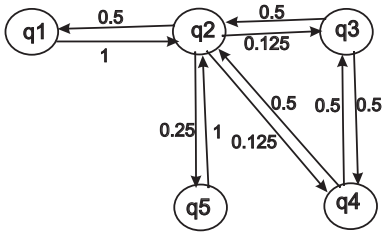


Fig. 6: Normalized Query Relevance Graph

structed. Then, given an user input query, depth first search algorithm is used to recommends queries to meet the original users' need. In Fig. 6. let q_1 be an user input query, then using depth first search the next visiting node is q_2 since there is only one outgoing edge from q_1. The next visiting node from q_2 is node q_1 with maximum probability. As q_1 is already a visited node, the next node q_5 is considered with maximum probability. Similarly, all visited nodes are obtained for recommendations.

Algorithm 1: QRGQR : Query Relevance Graph for Query Recommendation

Input : Input query q
Output: Top-5 Recommended Queries
begin

1. Construct query relevance graph *G=(V,E)* using the method shown in the section IV-C. The directed edges E are weighted by normalization as shown in Fig. 6.
2. Given a query q, apply depth first search method on query relevance graph's nodes V.
3. The first Top-5 results are the recommended queries.

5 EXPERIMENTS

5.1 Data Collection

Publicly available America On-line(AOL) search engine data[3] is used to construct query recommendation graph. Nearly 3813395 click through information with 1293620 unique queries and 400694 unique URLs have been considered.

The Web user activities can be obtained by click through data records. The users' interest and latent semantic relationship of users and queries, queries and clicked URLs can be retrieved. Table 2 shows sample of search engine click through data. Each line of click through data contains information about user ID, user query, query submission time for search, item rank of clicked document and clicked URL domain name. Each line in the data represents either user has not clicked the returned result after issuing query or clicked the document in returned results from a query.

In this paper, we have used the relationship of queries and URLs for construction

Table 2: Search Engine Click through Data Sample

AnonID	Query	QueryTime	ItemRank	ClickURL
142	rentdirect.com	2006-03-01 07:17:12		
142	staple.com	2006-03-17 21:19:29		
142	westchester.gov	2006-03-20 03:55:57	1	http://www.westchestergov.com
142	vera.org	2006-04-08 08:38:42	1	http://www.vera.org

of query click graph and query to query relationship for construction of query text similarity graph. We have ignored user ID, rank and time information of click through data.

5.2 Data Cleaning

Click through data of search engine consists of noise which affects the efficiency of query recommendation algorithm. Data is filtered by keeping only English queries i.e., queries with characters *a, b .. z* and space. After cleaning, data is reduced by 58.45%. Nearly 490866 distinct queries and 334224 distinct URLs have been used in this experiments.

As discussed previously, most of the time, the user tends to submit short queries with only one, two or three terms and therefore filtered data is obtained by keeping queries less than six terms. Further, data is reduced by 6.20%, resulting in 448299 distinct queries and 318644 distinct URLs. Query Recommendation results are generated by including all queries and by including only those queries which have less than six terms. We observe that both the results are similar.

5.3 Varying of Parameter-Jaccard Coefficient

As discussed in section IV, Jaccard coefficient(J_c) is defined as the fraction of common words between two queries. Query Recommendation results are generated by varying the value of J_c greater than 0.5 and 0.6 as shown in Table 3 on portion of the data. Finally, it is observed from the results, that when the optimal value for Jaccard coefficient is greater than 0.6, it yields more related queries.

When the query text similarity graph is constructed with J_c value greater than 0.5, queries which are only literally similar are obtained from the generated query relevance graph. For example, in Table 3, for the query *java*, queries *new one campaign commercial* and *one campaign commercial* are also obtained which are not relevant for the given input query. For the query *bank of america*, queries *bank of america banking centers, bank of america online banking* and *bank of america banking on line* are obtained which are only literally similar.

When the value of J_c is greater than 0.6, it results in those queries which are not only literally similar but also latent semantically relevant queries. For input query *java*, queries *java sun systems* and *sun java* are obtained which are latent semantically relevant queries.

If the value of J_c is equal to 0.5, then irrelevant queries are being selected. For example, consider two queries *California state university* and *California state polls* in which the value of J_c is 0.5, and both the queries are not relevant. Even when the value of J_c is less than 0.5, words matching between the two queries reduces, and hence relevant queries are not obtained.

If the value of J_c is greater than 0.7 then the number of recommendations obtained are less as number of matching queries decreases with higher value of J_c. Similarly, less recommendations are obtained for J_c greater than 0.8 and 0.9.

Table 3: Query Recommendation results by varying Jaccard Coefficient

Query	Jaccard >0.5	Jaccard >0.6
java	new one campaign commercial 0.6 one campaign commercial 0.3333 java sun systems 0.2 sun java 0.1111	java sun systems 0.2 sun java 0.1111 download java for windows 0.1111 java download 0.0588
bank of america	bank of america banking centres 0.0583 bank of america online banking 0.0163 bank of america banking on line 0.0063	www bank of america 0.0575 opening savings account 0.0033 certificates of deposit interest rates california banks 0.0435

5.4 Query Recommendation results

Experiments have been conducted on 8GB memory and intel Xeon(R) CPU E31220 @ 3.10GHz Quad Core processor workstation. Dataset used for both Hdiff and QRGQR

method are same. We have displayed the Top-5 recommendation results of Heat Diffusion algorithm [34] and our QRGQR algorithm in Table 4. For Heat Diffusion Algorithm, the numeric values shows the heat value for that query. For QRGQR algorithm, the numeric value shows the normalized value of query in the query relevance graph.

5.5 Performance Analysis

From the results shown in Table 4, it is observed that our query relevance model is recommending literally similar queries as well as latent semantically relevant queries. As discussed earlier, most of the time, the user tends to submit short queries with only one, two or three terms. Therefore, 60 test queries with one, two or three terms have been considered and different topics for input queries, such as Health, Shopping, Computer, Art have been covered in our experiments.

For example, given an input query *java*, the algorithm recommends *java download*, and *download java script*, which are literally similar queries. While a *sun microsystems* query recommends the company name of the java platform. This query is latent semantic to the input query *java*. Similarly, for the input query *free music*, the query recommendation results are *free music downloads*, *music downloads*, *shareware music downloads*, *broadway midi files* and *midi files*. Midi file is the Musical Instrument Digital Interface protocol for music. Here, it is observed that the resulting queries are literally similar and latent semantically to the given input query.

The performance of our QRGQR algorithm is compared with Heat Diffusion algorithm (Hdiff) [34]. In the heat diffusion algorithm, the query-URL bi-partite graph is constructed from the *search log*. For a given input query, depth first search algorithm is used to construct a sub graph on bi-partite graph by visiting nodes with high probability. The diffusion process is applied on the sub graph to calculate stationary probability for ranking recommendations results. The top-5 queries based on heat value (stationary probability) are used as recommended results.

To evaluate the quality of semantic relation is not easy in query recommendation as the queries taken as input is issued by users, and no linguistic resources are available. This paper evaluates quality of semantic relation manually with the help of three human experts and automatic evaluation based on Open Directory Project (ODP) database. We have adopted the method used in [38] to evaluate the quality of query recommendation results.

In manual evaluation, twelve research students are asked to rate the query recommendation results. We have asked them to evaluate relevance between testing queries and recommended results in the range of 0 to 1, in which 0 means totally irrelevant and 1 means totally relevant. The average value of rating results is shown in Fig. 7. It is observed that the accuracy of our algorithm QRGQR increases by 25.2% in comparison with Hdiff.

Variance is computed to find out whether rating provided by three research students agree upon each other, for query recommendation results. If variance is zero then rating values are identical and smaller variance indicate that the ratings are very close to mean hence to each other. The variance of user rating value is calculated for each testing query for both QRGQR and HDiff algorithm. The average variance of all 60 queries for QRGQR algorithm is 0.00781 and for HDiff is 0.01181. It is observed from both

Table 4: Query Recommendation Results Comparison

Sr.No	Query	Heat Diffusion Algorithm	QRGQR Algorithm
1	java	word whomp game 0.9044	download java 0.012 download java script 0.087 java download 0.32 sun microsystems 0.0455 sun microsystems colorado 0.0625
2	google	internet explorer 0.9869 microsoft 0.9024 windows update 0.8813 google com 0.8662 windowsxp updates 0.8652	google search 0.026 google search engine 0.1084 search engine 0.0494 lyrics search engine 0.1324 lyrics search 0.3158
3	fireworks	phantom fireworks 0.8651	firecrackers 0.25 phantom fireworks 0.25 how to make a firework 0.25 phantomfireworks 0.25
4	free music	sad songs 0.9174 bored 0.9075 humorous pictures 0.9051 free music downloads 0.8962 funny quotes 0.8928	free music downloads 0.2523 music downloads 0.0588 shareware music downloads 0.007 broadway midi files 0.0159 midi files 0.5
5	six flags	sixflags 0.8653	six flags kentucky 0.0027 six flags kentucky kingdom 0.0444 sixflags st louis 0.0286 splash water kingdom 0.0286 sixflags over texas 0.0294
6	college financial aid	department of education 0.879 fafsa 0.8705 nysed 0.8652	financial aid 0.5122 student financial aid 0.2222 pell grant 0.0556 pell grants 0.0857 sat tests 0.05
7	hurricane	gold flowers 0.9437 encyclopedia 0.9169 hurricane katrina 0.9129 wikipedia 0.911 online encyclopedia 0.8966	hurricane rita 0.1429 national hurricane center 0.1429 national weather service 0.1429 national weather 0.7956 weather 0.014
8	wedding	wedding channel 0.8653	bridal shows nj 0.0714 weddingchannel 0.0119 wedding dresses 0.125 lavender wedding dresses 0.0287 tea length wedding dress 1.0
9	hospitals	greenville ky hospital telephone number 0.8652	greenville ky hospital teleph. number 0.25 fairview hospital by columbus 0.25 operating room cost center charges 0.25 hospital operating room charges 0.25
10	poems	teen poems 0.9255 bored 0.9076 humorous pictures 0.9051 friendship poems 0.8945 love poems 0.8936	love poems 0.0426 best love poems 0.0465 teen poems 0.0189 types of poems 0.0175 different types of poems 0.0667

the variance values that all the rating values are identical i.e., rating provided for query recommendations by three research students agrees with each other.

In ODP, for user input query categories are matches in the form of paths between di-

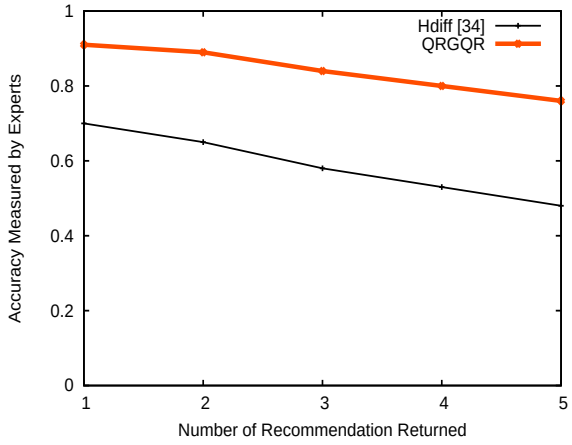


Fig. 7: Accuracy comparison measured by Experts

rectories which are ordered by relevance. For example, the query *solar system* provides the category *Science : Astronomy : Solar System*, while one of the results for *planets* would be *Science: Astronomy : Solar System : Planets*. Here, ":" is used to separate different categories. Hence, to measure the relation between two queries, similarity between two categories D and D_1 is measured by the longest common path CP(D,D_1) divided by the longest path between D and D_1. This similarity is defined as $sim(D,D_1)$ = $|\text{CP}(D,D_1)|/max\{|D|,|D_1|\}$, where $|D|$ is defined as the length of path of category D. The similarity between the two queries above mentioned is 3/4. Since they share the path *Science : Astronomy : Solar System* and the longest path is made of the four directories. The similarity between two queries is evaluated by measuring the similarity between the most similar categories of the two queries among the top five answers provided by ODP.

Accuracy comparison measured by ODP is shown in Fig. 8, from which it is observed that the accuracy of our algorithm QRGQR increases by 23.2% in comparison with Hdiff. From the above two evaluation process, it can be concluded that our proposed query recommendation algorithm is efficient.

The graph construction method is the major difference between our proposed model and heat diffusion model. In heat diffusion model, query-URL bi-partite graph is used. The weight of edge between query-URL is calculated by number of times query-URL pair is clicked. In the proposed model URL-query relation and query text similarity relation is used to construct query-query graph i.e., query relevance graph. The weight of edge between *query-query* is calculated by counting the number of common URLs in URL-query relation. The weight of edge between $query_1$-$query_2$ is calculated by count-

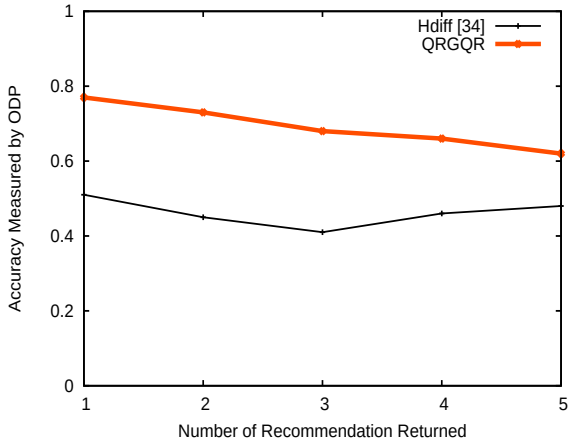


Fig. 8: Accuracy comparison measured by ODP

ing the occurrence of $query_1$ in the search log followed by $query_2$ in query text similarity relation. The query relevance graph identifies richer query relevance information as there are more available paths to follow in the graph. Query-Query pair occurrence is considered for weight calculation of an edge in proposed method rather than calculating the number of times query-URL pair clicked information. Hence our proposed algorithm outperforms the heat diffusion algorithm.

To analyse the above discussed graphs, nearly 454226 data is considered. For the user input query *java* query-URL bi-partite graph is considered as shown in Fig. 9. In Hdiff method, for input query *java*, sub-graph is constructed by visiting *www.java.com*, *download java for windows* and *www.mvps.com* nodes after normalizing weight. Only *download java for windows* is displayed as recommendations result as there is no path available to proceed. In the proposed method, query-click graph and query text similarity graph is constructed as shown in Fig. 10 and Fig. 11 respectively. In query click graph, edges exist between nodes representing other queries as they are connected with common URL *www.java.com* which is not represented in the graph. Query relevance graph is constructed by combining Fig. 10 and Fig. 11. In query relevance graph node *Java sun systems* is added from query text similarity graph which is constructed by calculating Jaccard value between two queries in search log. The query text similarity graph helps to boost weight between two queries if that queries are already present in query click graph or new node is added while constructing query relevance graph. It is observed from query relevance graph that for input query *java*, there are more path to follow. Hence, the proposed algorithm produces more number of relevant queries than Hdiff method. Top-4 Recommended queries by proposed method are *java sun systems*, *sun java*, *dowload java for windows* and *java download*.

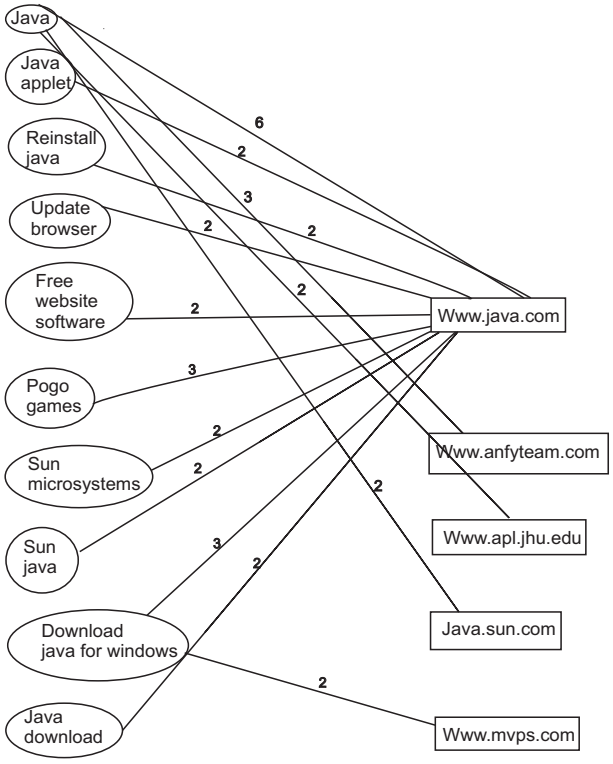


Fig. 9: Query-URL bipartite graph for input query *java*

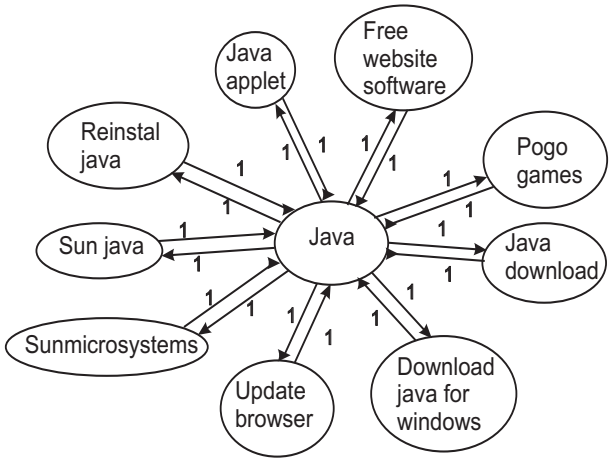


Fig. 10: Query Click Graph for input query *java*

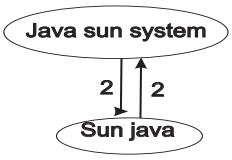


Fig. 11: Query Similarity Graph for input query *java*

5.6 Efficiency

The complexity of Hdiff and QRGQR algorithms are computed in terms of graph construction time. Consider there are m queries and n URLs in search log, where $m > n$. The query-URL bipartite graph requires $O(m)$ time for construction. The query click graph requires $O(n)$ time for generating p nodes, where $p < m$ and query text similarity graph requires $O(m)$ time for generating q nodes, where $q < m$. The query relevance graph requires $O(p+q)$ time for construction which combines output of query click graph and output of query text similarity graph. The total Time required to construct query relevance graph is $O(m) + O(n) + O(p+q)$ which is higher than the time required for query-URL bipartite graph. But the graph construction process comes under preprocessing stage. Once the graph is ready; it is stored in the search engine's memory for recommendation. In recommendation stage, the QRGQR gives larger number of relevant queries than heat diffusion method in the same retrieval time. It is observed the retrieval time for the query recommendation of both Hdiff and QRGQR method is around 0.01 seconds.

5.7 Image Recommendation

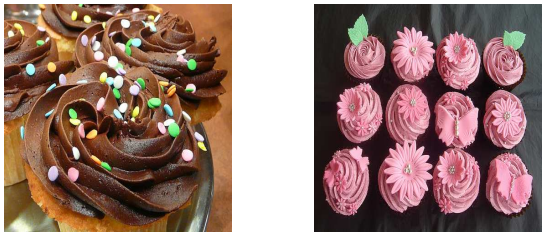
Image Recommendation is a challenging task as the tremendous amount of multimedia information such as images and videos are available freely on the internet. It is necessary to retrieve relevant images as per user requirement. People are uploading various images on popular websites like Flicker, Facebook, Picasa etc. and tag to it. Later, these images are searched on web by issuing query. As a general method for recommendation, QRGQR can also be used for Image Recommendation. This section presents related images recommendation to the given query i.e., tag of image.

In this work, we have considered 500 images and 1500 tags associated with it. With this data, the proposed algorithm can be applied for *Tag to Image* and *Tag to Tag* recommendation. Results for *Tag to Image* and *Tag to Tag* recommendation are shown below. The Graph construction method for *Tag-Tag* to recommendation is as follows: First, *Image-tag* bipartite graph is constructed using similar procedure for constructing query-URL bipartite graph. *Tag-Tag* graph is constructed using similar procedure for constructing query click graph. The weight of edge is calculated by counting number of times the tag pairs occur together in different images. Weights are normalized as per equation 2, in which instead of a query the tag is considered. Here, Jaccard similarity is not calculated between two tags as tags have usually a single word. For a given input

tag, the depth first search algorithm is applied on the tag-tag graph for recommendations. Similarly, the tag-image recommendation is generated by considering the tag to image mapping from the image-tag graph. For a given tag as input, top 2 images are displayed as shown in Fig 12. for the tag *sky*, in Fig. 13 for the tag *cake* and in Fig. 14 for the tag *ladybug*. The results of Tag recommendation are shown in Table 5. In order to evaluate the quality of our image recommendation, twelve research students are asked to rate(in the range of 0 to 1) the image recommendation results. The average value of the rating results are 0.83 and 0.67 for our method and heat diffusion method respectively. It is observed the accuracy of our algorithm increases by 16% in comparison with heat diffusion method.

(a) (b)

Fig. 12: Image Recommendation for tag 'sky'



(a) (b)

Fig. 13: Image Recommendation for tag 'cake'

6 CONCLUSIONS

We have proposed a general method for query recommendation by combining *query click graph* which captures the relationship between queries frequently clicked on common URLs and *query text similarity graph* which finds the similarity between two

(a) (b)

Fig. 14: Image Recommendation for tag 'ladybug'

Table 5: Tag Recommendation Results

sky	cake	ladybug
blue 0.0837	chocolate butter cream 0.2	beetle 0.25
cloud 0.0582	chocolate 0.25	insect 0.25

queries using Jaccard similarity. The proposed method generates *literally* as well as *semantically* relevant queries for users' need. Extensive experiments are performed on America On-line (AOL) search data. Query Relevance Graph for Query Recommendations (QRGQR) outperforms Heat diffusion method [34] by providing more relevant queries for a given input query with almost the same computation time as Heat diffusion method.

Quality of semantic relation is evaluated manually with help of human experts and automatic evaluation based on Open Directory Project (ODP). The accuracy of QRGQR increases by 25.2% and 23.2% in comparison with Hdiff for manual and ODP evaluation respectively. QRGQR provides better semantically relevant queries since the query relevance graph identifies richer query relevance information as there are more available paths to follow in the graph. This is due to *Query-Query* pair occurrence that is considered for weight calculation of an edge in the proposed method rather than calculating the number of times query-URL pair clicked information.

The time complexity of the Hdiff and QRGQR is linear. The query-URL bipartite graph used in Hdiff requires $O(m)$ time for construction while the total time required to construct query relevance graph in QRGQR is $O(m) + O(n) + O(p+q)$ and can be ignored as it is a one time computation. QRGQR provides enhanced semantic accuracy at the cost of slightly larger storage space. The QRGQR algorithm provides a comprehensive enhanced recommendation system for query, image, and product. The query click graph has relationship between query and URLs, and hence the QRGQR algorithm can also be used to recommend URLs. Further, topic based text similarity computation to construct query text similarity graph needs to be explored.

References

1. Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google News Personalization: Scalable Online Collaborative Filtering. *WWW '07: In the Proceedings of 16th International Conference on World Wide Web*, pages 271–280, 2007.
2. Hao Ma, Irwin King, and Michael R Lyu. Effective Missing Data Prediction for Collaborative Filtering. *SIGIR '07: In the Proceedings of 30th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 39–46, 2007.
3. Greg Pass, Abdur Chowdhury, and Cayley Torgeson. A Picture of Search. *In the Proceedings of 1st International Conference on Scalable Information Systems*, June 2006.
4. Zhu Kunpeng, Wang Xiaolong, and Liu Yuanchao. A New Query Expansion Method based on Query Logs Mining. *International Journal on Asian Language Processing*, 19(1):1–12, 2009.
5. Ju Fan, Hao Wu, Guoliang Li, and Lizhu Zhou. Suggesting Topic Based Query Terms as You Type. *In the Proceedings of 12th International Asia-Pacific Web Conference*, pages 61–67, 2010.
6. Adam L Kaczmarek. Interactive Query Expansion with the Use of Clustering-by-Directions Algorithm. *IEEE Transactions on Industrial Electronics*, 58(8):3168–3173, August 2011.
7. Yang Song, Dengyong Zhou, and Li-wei He. Query Suggestion by Constructing Term-Transition. *WSDM '12 : In the Proceedings of 5th ACM International Conference on Web Search and Data Mining*, pages 353–362, 2012.
8. Pawan Goyal, Laxmidhar Behera, and Thomas Martin McGinnity. Query Representation through Lexical Association for Information Retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 24(12):2260–2273, December 2012.
9. KW-T Leung, Wilfred Ng, and Dik Lun Lee. Personalized Concept-Based Clustering of Search Engine Queries. *IEEE Transaction on Knowledge and Data Engineering*, 20(11):1505–1518, November 2008.
10. Yan Chen and Yan-Qing Zhang. A Personalized Query Suggestion Agent based on Query-Concept Bipartite Graphs and Concept Relation Trees. *International Journal of Advanced Intelligence Paradigms*, 1(4):398–417, June 2009.
11. Saurabh Sharma and Neeraj Mangla. Obtaining Personalized and Accurate Query Suggestion by using Agglomerative Clustering Algorithm and P-QC Method. *International Journal of Engineering Research and Technology*, 1(5):1–8, July 2012.
12. Yiqun Liu, Junwei Miao, Min Zhang, Shaoping Ma, and Liyun Ru. How do Users Describe their Information Need : Query Recommendation based on Snippet Click Model. *International Journal on Expert Systems with Applications*, 38(11):13874–13856, October 2011.
13. Warin Narawit, Siripinyo Chantamunee, and Salin Boonbrahm. Interactive Query Suggestion in Thai Library Automation System. *In the Proceedings of 10th International IEEE Conference on Computer Science and Software Engineering*, pages 76–81, 2013.
14. P Deepa Shenoy, KG Srinivasa, KR Venugopal, and Lalit M Patnaik. Evolutionary Approach for Mining Association Rules on Dynamic Databases. *Advances in Knowledge Discovery and Data Mining*, pages 325–336, 2003.
15. KR Venugopal, E Ezhil Rajan, and P Sreenivasa Kumar. Impact of Wavelength Converters in Wavelength Routed All-optical Networks. *Computer communications*, 22(3):244–257, 1999.
16. P Deepa Shenoy, KG Srinivasa, KR Venugopal, and Lalit M Patnaik. Dynamic Association Rule Mining using Genetic Algorithms. *Intelligent Data Analysis*, 9(5):439–453, 2005.
17. KR Venugopal, E Ezhil Rajan, and P Sreenivasa Kumar. Performance Analysis of Wavelength Converters in WDM Wavelength Routed Optical Networks. *HIPC'98 : In the Proceedings of 5th International Conference on High Performance Computing, 1998.*, pages 239–246, 1998.

18. KB Raja, Kiran Kumar, Satish Kumar, MS Lakshmi, H Preeti, KR Venugopal, and Lalit M Patnaik. Genetic Algorithm based Steganography using Wavelets. *Information Systems Security.*, pages 51–63, 2007.
19. Youngho Kim, Jangwon Seo, W Bruce Croft, and David A Smith. Automatic Suggestion of Phrasal-Concept Queries for Literature Search. *International Journal of Information Processing and Management*, 50(4):568–583, July 2014.
20. Udo Kruschwitz, Deirdre Lungley, M-Dyaa Albakour, and Dawei Song. Deriving Query Suggestions for Site Search. *Journal of the American Society for Information Science and Technology*, 64(10):1975–1994, October 2013.
21. Nick Craswell and Martin Szummer. Random Walks on the Click Graph. *SIGIR '07 : In the Proceedings of 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 239–246, 2007.
22. Qiaozhu Mei, Dengyong Zhou, and Kenneth Church. Query Suggestion using Hitting Time. *CIKM '08: In the Proceedings of 17th ACM Conference on Information and Knowledge Management*, pages 469–477, 2008.
23. Hao Ma, Michael R Lyu, and Irwin King. Diversifying Query Suggestion Results. *AAAI '10 : In the Proceedings of 24th AAAI International Conference on Artificial Intelligence*, pages 1399–1404, 2010.
24. Aris Anagnostopoulos, Luca Becchetti, Carlos Castillo, and Aristides Gionis. An Optimization Framework for Query Recommendation. *WSDM '10 : In the Proceedings of 3rd ACM International Conference on Web search and Data Mining*, pages 161–170, 2010.
25. Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. Context-Aware Query Suggestion by Mining Click-Through and Session Data. *KDD '08 : In the Proceedings of 14th International ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 875–883, 2008.
26. Jiafeng Guo, Xueqi Cheng, Gu Xu, and Huawei Shen. A Structured Approach to Query Recommendation with Social Annotation Data. *CKIM '10 : In the Proceedings of 19th ACM International Conference on Information and Knowledge Management*, pages 619–628, 2010.
27. Yang Song, Dengyong Zhou, and Li-wei He. Post Ranking Query Suggestion by Diversifying Search Results. *SIGIR '11 : In the Proceedings of 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 815–824, 2011.
28. Rodrygo LT Santos, Craig Macdonald, and Iadh Ounis. Learning to Rank Query Suggestions for Adhoc and Diversity Search. *ACM Journal of Information Retrieval*, 16(4):429–451, August 2013.
29. Hossein Vahabi, Margareta Ackerman, David Loker, Ricardo Baeza-Yates, and Alejandro Lopez-Ortiz. Orthogonal Query Recommendation. *RecSys '13: In the Proceedings of the 7th ACM Conference on Recommender System*, pages 33–40, 2013.
30. Mohamed Sarwat, Justin J Levandoski, Ahmed Eldawy, and Mohamed F Mokbel. LARS* : An Efficient and Scalable Location-Aware Recommender System. *IEEE Transactions on Knowledge and Data Engineering*, 26(6):1384–1399, June 2014.
31. Yang Cao, Ju Fan, and Guoliang Li. A User-Friendly Patent Search Paradigm. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1439–1443, June 2013.
32. Brian McFee, Luke Barrington, and Gert Lanckriet. Learning Content Similarity for Music Recommendation. *IEEE Transactions on Audio, Speech and Language Processing*, 20(8):2207–2218, October 2012.
33. Wei Gao, Cheng Niu, Jian-Yun Nie, Ming Zhou, Jian Hu, Kam-Fai Wong, and Hsiao-Wuen Hon. Cross-Lingual Query Suggestion using Query Logs of Different Languages. *SIGIR '07 : In the Proceedings of 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 463–470, 2007.

34. Hao Ma, Irwin King, and Michael R Lyu. Mining Web Graphs for Recommendations. *IEEE Transactions on Knowledge and Data Engineering*, 24(6):1051–1064, June 2012.
35. Nick Craswell and Martin Szummer. Random Walks on the Click Graph. *In the Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 239–246, 2007.
36. Glen Jeh and Jennifer Widom. Simrank: A measure of structural-context similarity. pages 538–543, 2002.
37. Heasoo Hwang, Hady W Lauw, Lise Getoor, and Alexandros Ntoulas. Organizing User Search Histories. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):912–925, May 2012.
38. Ricardo Baeza-Yates and Alessandro Tiberi. Extracting Semantic Relations from Query Logs. *KDD '07 : In the Proceedings of 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 76–85, 2007.