Personalized Query Suggestion With Diversity Awareness

Di Jiang, Kenneth Wai-Ting Leung, Jan Vosecky, Wilfred Ng

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
{dijiang, kwtleung, jvosecky, wilfred}@cse.ust.hk

***Abstract*— Query suggestion is an important functionality provided by the search engine to facilitate information seeking of the users. Existing query suggestion methods usually focus on recommending queries that are the most relevant to the input query. However, such relevance-oriented strategy cannot effectively handle query uncertainty, a common scenario that the input query can be interpreted as multiple different meanings. To alleviate this problem, the concepts of *diversification* and *personalization* have been individually introduced to query suggestion systems. These two concepts are often seen as incompatible alternatives, because diversification considers multiple aspects of the input query to maximize the probability that some query aspect is relevant to the user while personalization aims to adapt the suggestions to a specific aspect that aligns with the preference of a specific user. In this paper, we refute this antagonistic view and propose a new query suggestion paradigm, *Personalized Query Suggestion With Diversity Awareness* (PQS-DA) to effectively combine diversification and personalization into one unified framework. In PQS-DA, the suggested queries are effectively diversified to cover different potential facets of the input query while the ranking of suggested queries are personalized to ensure that the top ones are those that align with a user's personal preference. We evaluate PQS-DA on a real-life search engine query log against several state-of-the-art methods with respect to a variety of metrics. The experimental results verify our hypothesis that diversification and personalization can be effectively integrated and they are able to enhance each other within the PQS-DA framework, which significantly outperforms several strong baselines with respect to a series of metrics.**

I. Introduction

Query suggestion is an important functionality provided by contemporary web search engines to help users formulate more effective search queries [1]. Most of the existing query suggestion methods [2][3][4][1][5] belong to the category of relevance-oriented query suggestion, which focuses on maximizing the overall *relevance* of the suggested queries in response to an input query. However, since search queries are typically short and ambiguous [6], the simplistic relevance-oriented methods usually fail in the face of query uncertainty, which widely exists in the scenario of general web search.

Consider the following example of query uncertainty: *When the search query "sun" is submitted to the search engine, the underlying information need can be related to at least one of the three facets: the star of the solar system, the computer manufacturer named Sun Microsystems or a newspaper in the United Kingdom. In this case, the relevance-oriented approaches usually generate suggestions that cover a few or even one facet, such as only suggesting queries about Sun Microsystems. Obviously, the results are unsatisfactory when the user is searching for information about the other interpretations.*

To alleviated the aforementioned problem, the recent strands of query suggestion research can be broadly separated into two categories: introducing either *diversification* or *personalization* to the conventional query suggestion. Some researchers have introduced *diversification* to web search results [7][8][9], recommendation systems [10][11] and query suggestion systems [6][12]. The logic of diversification is to cover as many facets of the input query as possible with a single query suggestion list. More specifically, diversification aims to minimize the number of the totally unsatisfied users, trading degrees of satisfaction in exchange for increasing the size of the satisfied population. The downside of diversification is that, for a specific user, the query suggestion lists may contain irrelevant suggestions and the irrelevant ones may even be ranked much higher than the relevant ones. For instance, when a user submits the query "sun" to search for some information about Sun Microsystems, presenting a suggestion list in which queries such as "solar energy" or "sun daily uk" are ranked higher than "oracle sun" or "sun solaris" is disappointing. As an alternative approach to tackle query uncertainty, some researchers propose to apply *personalization* in order to identify the suggestion candidates that are most similar to the user's search history [13][14]. In contrast to *diversification*, *personalization* strives to get further knowledge through a user's search history, in order to reduce the uncertainty of the input query. Essentially, personalization narrows down the scope of the possible interpretations to those that only align with the a user's personal preference. However, web search is essentially dynamic and a user's preference changes over time. For example, when a computer scientist submits "sun" to search for information about a hot topic of solar energy, suggesting queries that are only focused on Sun Microsystems is rather unsatisfactory. Therefore, traditional personalization approaches risk over-personalization and cannot effectively handle the dynamic change of a user's preference.

From the above discussion, we can see that both *diversification* and *personalization* have their own advantages and drawbacks. The result of *diversification* can be overly broad for a specific user but is flexible to handle the dynamic

978-1-4799-2555-1/14/$31.00 © 2014 IEEE

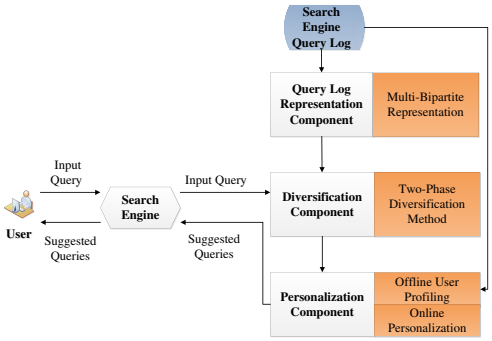


Fig. 1: System Architecture of PQS-DA

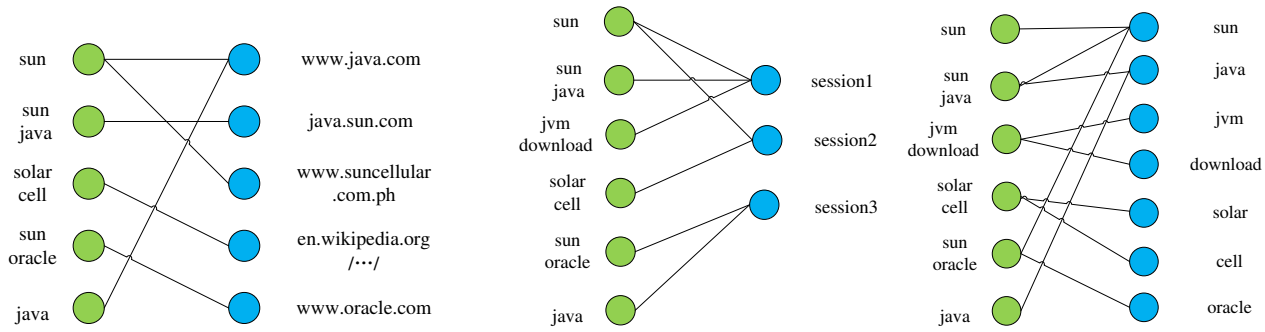
changes of a user's preference. In contrast, the result of *personalization* is too rigid to handle the changes of a user's preference but is effective for those who are looking for information relevant to his or her long-term preference. In order to achieve better query suggestion performance than only applying one of the two concepts, we propose a new query suggestion paradigm, *Personalized Query Suggestion With Diversity Awareness* (PQS-DA) to unify the two concepts which have been perceived to be incompatible so far. We hypothesize that a good query suggestion paradigm needs to integrally consider *diversification* and *personalization*, i.e., the suggested queries need to be diversified to cover different facets of the input query and the ranking of the suggested queries also needs to be personalized to align with the user's preferences. Consider the case of "sun" again: we cannot aggressively deny the probability that a computer scientist may search for information about solar energy, thus, the suggestions should be diversified to cover different facets such as Sun Microsystems, the solar system, the solar energy, etc. Furthermore, to improve a specific user's searching experience, the ranking of the suggested queries should be further personalized to align with the user's preference, e.g., ranking the queries such as"sun oracle", "sun java" higher than "sun solar system" higher for a computer scientist, since we know that the queries about computer science have better chance to be relevant to the user's current information need. In this way, we can better capture the broadness of a user's information needs and also facilitate the user's information seeking when the information need aligns with his or her long-term preference. While this seems to be a simple idea, we are not aware of any previous work that explores how to integrate both *diversification* and *personalization* to improve the performance of query suggestion.

As shown in Fig. 1, the architecture of PQS-DA can be divided into three major components: the query log representation component, the diversification component and the personalization component. We now outline the utility of each component. Conventionally, researchers utilize the click graph, which is a bipartite composed of the queries and URLs, to model the information in the query log. However, the simplistic query-URL bipartite has limitations such as low information coverage and being noise-prone [15], which limits its effectiveness in query suggestion. In the query log representation component, we propose a multi-bipartite representation for the query log data and this representation significantly improves the richness of the information available for the downstream query suggestion. In the diversification component, we propose a two-phase method to obtain a list of diversified query suggestion candidates. At the first phase, we utilize a context-aware regularization framework to identify the most relevant suggestion candidate by considering both the local and global information in the multi-bipartite representation. At the second phase, we iteratively identify the remaining suggestion candidates via a hitting time [14] based approach. In the personalization component, we employ a generative model named *User Profiling Model* (UPM) to integrate information such as search sessions, web dynamics and each individual user's preference of word usage and URL clicking. Based on the user profiles obtained from the UPM, the ranking of the suggested queries is organized to align with the current information need as well as the user's personal preference. To evaluate the effectiveness of PQS-DA, we conduct extensive experiments on a large-scale search engine query log. Compared with several strong baselines, the proposed framework demonstrates superior performance with respect to a variety of metrics.

The contributions of this paper are summarized as follows:

- We propose a new query suggestion paradigm, *Personalized Query Suggestion With Diversity Awareness* (PQS-DA), which paves the way for providing better query suggestions for each individual search engine user.
- We propose a new representation for query log data. The proposed multi-bipartite representation can comprehensively capture different kinds of relations between search queries in query log. A sophisticated mechanism is also designed to differentiate the importance of different relations between search queries.
- We design a novel two-phase method to obtain query suggestion candidates. This method utilizes technique of hitting time and the multi-bipartite representation to enhance both the relevance and diversity of the set of query suggestion candidates.
- We develop a new generative model, the *User Profiling Model* (UPM), to profile search engine users through integrating search sessions, web dynamics as well each user's preference of word usage and URL clicking. The user profile generated by the UPM provides concise and effective summary of each user's preference, which further personalizes the ranking of query suggestion candidates and generates the final query suggestion list.

The rest of this paper is organized as follows. In Section II, we review the related work. In Section III, we discuss the query log representation component. In Sections IV and V, we present the diversification component and the personalization component. In Section VI, we report the experimental results. Finally, the paper is concluded in Section VII.



(a) Query-URL Bipartite (Click Graph) (b) Query-Session Bipartite (c) Query-Term Bipartite

Fig. 2: Multi-Bipartite Query Log Representation of Table I

II. Related Work

Relevance-oriented query suggestion attracts much attention in recent years. Most of the relevance-oriented methods rely on the click graph to represent the information in query log. Mei *et al.* [14] proposed an algorithm using hitting time to make query suggestion on a click graph. Also based on the click graph, Cao *et al.* [2] proposed a query suggestion approach by using the concept sequence suffix tree. Youngho *et al.* [16] proposed a boolean query suggestion technique, which generates boolean queries by exploiting decision trees learned from pseudo-labeled documents. There also exists some work that focus on specific types of queries. For example, Szpektor *et al.* [5] proposed a method to extend the reach of query recommendation to long-tail queries by reasoning about rules between query templates rather than individual query transitions. Kato *et al.* [4] proposed an approach to present query suggestions to the user and the method was designed to help query reformulation actions such as specialization and parallel movement. Markov random walk on the click graph was studied in [15][17] for ranking documents and discovering search tasks. Deng *et al.* [18] proposed a new framework for modeling the click graph, in which the various query-URL pairs are treated differently. Besides viewing query log as click graph, probabilistic topic modeling technique is gaining momentum in text mining [19][20] and researchers have developed different topic models[21][22] to analyze search engine query log.

Recently researchers have been aware of the limitations of relevance-oriented query suggestion and explored several strategies to incorporate either diversification or personalization into query suggestions. Ma *et al.* [6] proposed a method based on Markov random walk and hitting time to diversify the query suggestion results. Song *et al.* [12] introduced a query suggestion framework which works in a post-ranking fashion. This framework is essentially different from those proposed in [6][14] because it relies on external knowledge such as the search results from the search engine. Leung *et al.* [13] introduced an approach that captures the user's conceptual preferences in order to provide personalized query suggestions. Mei *et al.* [14] also proposed a personalized query suggestion method by employing hitting time and creating pseudo query nodes in the click graph.

The differences between our work and the previous ones lie in two aspects: First, rather than introducing external knowledge bases, we aim to take full advantage of the rich information in query log and design a new representation to comprehensively capture the knowledge in the query log. Compared with the conventional click graph, the proposed multi-bipartite representation is more suitable for the task of query suggestion. Second, we propose a new query suggestion paradigm which takes both *diversification* and *personalization* into consideration. To the best of our knowledge, the two concepts have only been studied independently so far in the field of query suggestion and the present work is the first one seamlessly integrating both of them in a unified framework.

III. Multi-Bipartite Query Log Representation

TABLE I: Example of Search Engine Query Log

ID	UserID	Query	Clicked URL	Timestamp
q_1	u_1	sun	www.java.com	2012-12-12 11:12:41
q_2	u_1	sun java	java.sun.com	2012-12-12 11:13:01
q_3	u_1	jvm download		2012-12-12 11:14:21
q_4	u_2	sun	www.suncellular.com	2012-12-13 07:13:01
q_5	u_2	solar cell	en.wikipedia.org/.../	2012-12-13 07:14:21
q_6	u_3	sun oracle	www.oracle.com	2012-12-14 14:35:14
q_7	u_3	java	www.java.com	2012-12-14 14:36:26

As shown in Table I, a typical query log entry includes the query identifier, the user identifier, the search query, the clicked URL (if any) and the timestamp. Query log representation usually works as the basis of the downstream query suggestion. In existing work, the *de facto* representation of query log is the click graph (e.g., Fig. 2(a)). However, since two random queries rarely share the same clicked URLs [23], the click graph only captures a small portion of the rich information in query log. The narrow information coverage of the click graph heavily limits the downstream query suggestion's performance in capturing the different facets of the original input query and diminishes the chance of obtaining highly relevant candidates. Furthermore, clickthrough information is

inherently noisy [15] and may also be biased by users or robots with malicious intents [18]. In order to alleviate the aforementioned problem, we propose a multi-bipartite representation that models the relations between search queries via three different bipartites: the query-URL bipartite (e.g., Fig. 2(a)), the query-session bipartite (e.g., Fig. 2(b)) and the query-term bipartite (e.g., Fig. 2(c)). The query-URL bipartite and the query-term bipartite can be straightforwardly constructed based upon the raw query log. Constructing the query-session bipartite is more complicated and we present the definition of the *session* in *Definition 1*. Deriving sessions from raw query log is well studied in literature [24][25] and we employ the method in [25] to deriving sessions in the present work.

Definition 1: The ***session*** *is a series of search queries that are submitted to satisfy a single information need. For example, in the example of Table I,* $\{q_1, q_2, q_3\}, \{q_4, q_5\}$ *and* $\{q_6, q_7\}$ *are three different sessions.*

The superiority of the multi-bipartite representation is illustrated as follows. Consider the query "sun" in Table I. By using the query-URL bipartite of Fig. 2 (a), "sun" can only reach the query "java". By using the query-session bipartite of Fig. 2 (b), "sun" can reach the queries such as "sun java", "jvm download" and "solar cell". Through the query-term bipartite of Fig. 2 (c), "sun" can reach the queries such as "sun java", "sun oracle" and "java". By employing the three bipartites collectively, we can obtain more suggestion candidates for the query "sun". More importantly, we have a better chance to cover different facets of the input query "sun". The example above illustrates the structural superiority of the multi-bipartite representation. The remaining issue of applying the multi-bipartite representation to real-life usage is how to determine the weights for the edges of the three bipartites. Although we can simply utilize the raw relation frequency to assign weights to the edges, it causes loss of important information, since different relation pairs of the three bipartites are not sufficiently distinguished. Therefore, it is more reasonable to weigh the edges differently by considering their distinguishing capability. For instance, in the query-URL bipartite, it is intuitive that a heavily clicked URL with a high query frequency is less discriminative. This motivates us to propose an important concept, referred to as the *inverse query frequency of URL* (iqf^U), to measure the discriminative ability of the URLs. Suppose $|Q|$ is the total number of search queries in the query log, the inverse query frequency for the URL u_j is defined as follows:

$$iqf^U(u_j) = \log|Q| - \log n^U(u_j) = \log\frac{|Q|}{n^U(u_j)}, \quad (1)$$

where $n^U(u_j)$ is the total number of queries that are connected with the URL u_j and it can be calculated by $n^U(u_j) = \sum_{i\in Q} 1_{int(q_i,u_j)}$.

Similarly, a session containing many different queries is less discriminative in terms of its underlying information need and a frequently used query term is less discriminative in terms of its semantic meaning. We further define the *inverse query frequency of session* (iqf^S) for a session s_j and the *inverse query frequency of term* (iqf^T) for a term t_j as follows:

$$iqf^S(s_j) = \log|Q| - \log n^S(s_j) = \log\frac{|Q|}{n^S(s_j)}, \quad (2)$$

$$iqf^T(t_j) = \log|Q| - \log n^T(t_j) = \log\frac{|Q|}{n^T(t_j)}, \quad (3)$$

where $n^S(s_j)$ is the total number of queries that are connected with the session s_j, $n^S(s_j) = \sum_{i\in Q} 1_{int(q_i,s_j)}$, $n^T(t_j)$ is the total number of queries that are connected with the term t_j and $n^T(t_j) = \sum_{i\in Q} 1_{int(q_i,t_j)}$.

In the multi-bipartite representation, we calculate the weights of the edges by multiplying the inverse query frequencies iqf^U, iqf^S and iqf^T with the raw frequencies c^U, c^S and c^T in a unified way, namely,

$$cfiqf^U(q_i, u_j) = c^U_{ij} \times iqf^U(u_j). \quad (4)$$

$$cfiqf^S(q_i, s_j) = c^S_{ij} \times iqf^S(s_j). \quad (5)$$

$$cfiqf^T(q_i, t_j) = c^T_{ij} \times iqf^T(t_j). \quad (6)$$

The intuition behind the aforementioned edge weighing mechanism is that different edges are treated differently according to their corresponding inverse query frequencies, so that the common relations with less frequent yet more specific URLs, sessions or terms are of greater value than the common relations on frequent ones.

IV. Diversification of Query Suggestion Candidates

In this section, we discuss how to obtain diversified query suggestion candidates through the diversification component of PQS-DA. In Section IV-A, we discuss the method of building compact multi-bipartite representation to reduce the overall computational cost of the downstream query suggestion. In Section IV-B, we discuss the approach of identifying the most relevant suggestion candidate. In Section IV-C, we discuss how to iteratively identify the remaining query suggestion candidates while achieving diversification.

A. Build Compact Multi-Bipartite Representation

In terms of the computational cost of query suggestion, two main concerns arise when we use the entire query log. First, the multi-bipartite representation can be very large. Most queries are actually irrelevant to the original input query but they increase the computational cost. Second, the mathematical calculation can be very time-consuming when the number of variables that need to handle with is in the millions. Thus, we now present a method of building a compact multi-bipartite representation. We consider the original input query and those in its *search context* (*Definition 2*) as the starting compact representation and iteratively expand this

representation by Markov random walk via the full multi-bipartite representation, until the total number of queries in the compact one reaches a desired size $\mathbb{Q}$. Then the downstream query suggestion algorithm is performed on the compact representation rather than the full one. Similar approaches have been successfully applied in previous work such as [26][14]. Based on the edge weights of the compact representation, we apply normalization and derive the query-URL matrix W^U, the query-session matrix W^S and the query-term matrix W^T to represent the query relations.

Definition 2: The ***search context*** *of a query q is defined as the set of the previously submitted queries within the same search session. For example, in the session composed of q_1, q_2 and q_3 in Table I, $\{q_1\}$ is the search context of q_2 and $\{q_1, q_2\}$ is the search context of q_3.*

B. Find the First Query Suggestion Candidate

We adopt an iterative approach to identify the suggestion candidates, i.e, selecting one candidate at each time and then finding the next one based on those that have been selected so far. In the iterative approach, identifying the first candidate is the cornerstone of the whole candidate selection process. Therefore, the goal of the technique here is to find the most relevant suggestion candidate with respect to the input query. To better capture the underlying information need of the input query, we take both the input query and its search context into consideration. Given the input query q, we form a $1 \times \mathbb{Q}$ vector F^0, with the entry of q equal to 1, the entries of the queries in the search context of q are assigned with values according to a backward decay function [27], and all the other entries are equal to 0. The intuition of the decay function is that the semantic similarity between q and those in its search context decays with respect to the elapsed time. Assume that the timestamp of the input query is t_q, then the entry value of a query q' in search context is initialized as follows:

$$F^0_{q'} = e^{\lambda(t_{q'} - t_q)}, \tag{7}$$

where $t_{q'}$ is the timestamp of query q' and λ is a scaling parameter.

In order to identify the most relevant suggestion candidate, we propagate the information stored in F^0 via the multi-bipartite representation and identify suggestion candidate that demonstrates the highest similarity with q. More specifically, we utilize a regularization framework to take into account both the fitting constraint and the smoothness constraint, which are defined based on the multi-bipartite representation. The fitting constraint is defined by Equation (8), which constrains that the estimated entry values for q and those in its search context should not differ too much from those stored in F^0.

$$\frac{1}{2}\sum_{i}^{\mathbb{Q}} ||F_i - F^0_i||^2. \tag{8}$$

The smoothness constraint is defined by Equation (9), which constrains that the queries in close relations tend to have similar values in F.

$$\frac{1}{2}\sum_{i,j}^{\mathbb{Q}}\sum_{k}^{\mathbb{X}} W^X_{ik} W^X_{jk} || \frac{F_i}{\sqrt{D^X_{ii}}} - \frac{F_j}{\sqrt{D^X_{jj}}} ||^2, \tag{9}$$

where $X \in \{U, S, T\}$ and D^X is a diagonal matrix in which the element $D^X_{i,i}$ equals the sum of all elements in the ith row of $W^X {W^X}^\top$. By incorporating the two constraints by a positive trade-off parameter μ, we formally define a minimization problem to estimate the relevance of the queries in the multi-bipartite representation:

$$\min_{F,\gamma} \frac{1}{2}\sum_{i}^{\mathbb{Q}} ||F_i - F^0_i||^2 + \mu\gamma, \tag{10}$$

such that the following expressions hold, $\forall X \in \{U, S, T\}$

$$\frac{1}{2}\sum_{i,j}^{\mathbb{Q}}\sum_{k}^{\mathbb{X}} W^X_{ik} W^X_{jk} || \frac{F_i}{\sqrt{D_{ii}}} - \frac{F_j}{\sqrt{D_{jj}}} ||^2 \leq \gamma. \tag{11}$$

This procedure amounts to taking the upper bound of the smoothness constraint over all the three bipartite and applying it for regularization. Then, we derive the dual problem and rewrite the convex optimization problem with the following equation:

$$\max_{\alpha,\pi} \min_{F,\gamma} \frac{1}{2}\sum_{i}^{\mathbb{Q}} ||F_i - F^0_i||^2 + \mu\gamma + \sum_{X\in\{U,S,T\}} \alpha^X (\frac{1}{2}\sum_{i,j}^{\mathbb{Q}}\sum_{k}^{\mathbb{X}} W^X_{ik} W^X_{jk} || \frac{F_i}{\sqrt{D_{ii}}} - \frac{F_j}{\sqrt{D_{jj}}} ||^2 - \gamma) - \pi\gamma, \tag{12}$$

where the Lagrange multipliers satisfy $\alpha_X \geq 0$ and $\pi \geq 0$. The above formula can be simplified as follows:

$$\max_{\alpha,\pi} \min_{F,\gamma} \frac{1}{2}||F - F^0||^2_F + \mu\gamma + \sum_{X\in\{U,S,T\}} \alpha^X (\frac{1}{2} tr(F^\top (I - \mathcal{L}^X) F) - \gamma) - \pi\gamma. \tag{13}$$

where I is the identity matrix, $\mathcal{L}^X = D^{X\frac{1}{2}} W^X {W^X}^\top D^{X\frac{1}{2}}$ and $||\cdot||_F$ is the Frobenius norm[1]. In order to solve the inner optimization problem, we set the derivative with respect to γ to zero and obtain the following equation:

$$\mu - \sum_{X\in\{U,S,T\}} \alpha^X = \pi. \tag{14}$$

By substituting Equation (14) into Equation (13) and setting the derivative with respect to F to zero, we obtain a large sparse linear system as follows:

$$((1 + \sum_{X\in\{U,S,T\}} \alpha^X) I - \sum_{X\in\{U,S,T\}} \alpha^X \mathcal{L}^X) F^* = F^0. \tag{15}$$

[1] http://mathworld.wolfram.com/FrobeniusNorm.html

The numerical problem of Equation (15) has been intensively studied and there exist efficient algorithms, whose computational time is linear in the number of the non-zero entries in the coefficient matrix [28]. Moreover, when the linear system solver is parallelized and distributed on a computing cluster, the proposed algorithm easily scales to much larger datasets. Equation (15) is not very sensitive to α^X and can be empirically tuned in a similar approach as the previous work [26]. After getting F^* and excluding the entries that correspond to the input query and those in the search context of the input query, the most relevant candidate can be directly obtained by choosing the query that has the largest entry value in F^*.

C. Find the Remaining Query Suggestion Candidates

After determining the most relevant suggestion candidate, we need to find the remaining K-1 ones if in total K queries need to be recommended to the user. The remaining candidates need to be relevant to the input query but also be different from each other to cover different facets of the input query. We now propose the cross-bipartite hitting time to achieve this goal. Imagine a random walker on the multi-bipartite representation. At each step, the walker can do one of two following things: it moves to a neighbor query according to the edge weights in the current bipartite or it is teleported to another bipartite and moves to a neighbor query. To represent the cross-bipartite transition, we use a 3×3 matrix N_k to store the cross-bipartite transition probabilities for the query q_k, in which $N_k[i,j]$ is $p(X_j|q_k, X_i)$. Then the cross-view transition probabilities for all queries can be stored in a $\mathbb{Q}\times\mathbb{Q}$ diagonal transition matrix $\mathbf{N}$ in which the kth diagonal element $N[k,k] = N_k$. Without any prior knowledge, we utilize equal weights for the three bipartite. We further utilize a three-dimension row vector $P[b,a]$ to denote the intra-bipartite transition probabilities. In $P[b,a]$, the entry $p^X(q_a|q_b)$ to denote the intra-bipartite transition probabilities in the bipartite X ($X \in \{U,S,T\}$) from q_b to q_a. The value of $p^X(q_a|q_b)$ can be straightforwardly obtained from the compact representation obtained in Section IV-A.

At the starting point, we set $\mathbf{M}^0$ as a diagonal matrix with each element being a uniform three-dimension row vector in which each element is 1/3. Therefore, $M[i,j]$ is also a three-dimensional row vector which can be calculated as follows:

$$\mathbf{M}[i,j] = \sum_{k=1}^{\mathbb{Q}} (\mathbf{M}^0[i,k]\cdot\mathbf{N}_k)\overrightarrow{\times}P(q_j|q_k). \tag{16}$$

After the initialization, we then employ the cross-bipartite hitting time to iteratively identify the remaining K-1 suggestion candidates. Let S be the set of candidates that have been selected, the expected hitting time $h(q_i|S)$ of the random walk is the expected number of steps before the query q_i visits the starting set S. It can be easily verified that the hitting time satisfies the following system of linear equations:

$$\begin{cases} h(q_i|S) = 0 & if \quad q_i \in S \\ h(q_i|S) = 1 + \sum_{q_j\in\mathcal{N}(q_i)}\sum_X M^X(i,j)h(q_j|S) & if \quad q_i \notin S. \end{cases} \tag{17}$$

where $\mathcal{N}(q_i)$ denotes the neighbors of the query q_i. The recurrence relations in Equation (17) can be used to iteratively compute the expected hitting time. We skip the inference of the hitting time equation in this paper due to the space limitation. Interested readers can refer to [14] to find more details about hitting time.

We select the second suggestion candidate q_2 as the one having the largest expected hitting time to the current set S. This naturally inhibits queries closely connected to those in S, hence encourages diversity. If we need the third suggestion, we can simply add q_2 into the subset S and compute the expected hitting time once more. The whole process of obtaining the K query suggestion candidates is presented in Algorithm 1. Based on the discovering sequence of each candidate, we actually obtain a ranked list of K candidates, which is sorted with a descending relevance to the input query and potentially covers different facets of the input query. In the diversification component, the relevance between the input query and the suggested query is evaluated via their affinity in the multi-bipartite representation. In the next section, we discuss how to incorporate the relevance from each individual user's perception.

Algorithm 1 Diversification of Query Suggestion Candidates

Input: The input q, the search context $\mathbf{C}_q$ and the number of iterations l.

Output: A ranked list of K query suggestion candidates.

```
Initialize F^0 according to q and C_q;
Compute F* via Equation (15);
Identify the first candidate q_1 based on F*;
repeat
    Set h_0(k) = 0 for the existing candidate k;
    for t = 0; t < l; t++ do
        h_{t+1}(v) = Σ_{u≠v} Σ_X M^X[u,v] h_t(u) + 1;
    end for
    j = arg max_i h_i;
    Select q_j as the next candidate;
until K-1 candidates are picked.
```

V. Personalization of Query Suggestion List

In this section, we discuss the personalization component and illustrate how it determines the final ranking of the suggestion candidates for each individual user. In Section V-A, we provide the details about *User Profiling Model* (UPM) for offline user profiling. In Section V-B, we discuss the details of online personalization through the user profiles generated by UPM.

A. Offline User Profiling

Profiling search engine users for the purpose of personalization is challenging since query log contains multiple types

of information, such as the words, URLs and timestamps, which collectively reflect the user's preference. Since all these types of information are strictly coupled, an effective model needs to capture the latent relations between different types of information. In the face of this challenge, we propose the *User Profiling Model* (UPM) and it does not only incorporate multiple types of information in a principled way, but also reduces the data dimension of the plain text of query log data and make the user profiles concise enough for offline storage and efficient online personalization.

Algorithm 2 Generative Process of the UPM

for document $d \in 1, ..., D$ **do**
  Draw d 's topic distribution $\theta_d \sim$ Dirichlet(α);
  for topic $k \in 1, ..., K$ **do**
    Draw a word distribution $\phi_{kd} \sim$ Dirichlet(β_k);
    Draw a URL distribution $\Omega_{kd} \sim$ Dirichlet(δ_k);
  end for
  for each session s in d **do**
    Choose a topic $z \sim$ Multinomial(θ_d);
    Generate the words $w \sim$ Multinomial(ϕ_{zd});
    if URL existence indicator $X_{ds} = 1$ **then**
      Generate the URLs $u \sim$ Multinomial(Ω_{zd});
    end if
    Generate the timestamps $t \sim$ Beta(τ_z);
  end for
end for

The generative process of the UPM is presented in Algorithm 2. We organize the query log entries of each user as a document and assume that each document has a Multinomial distribution over a set of latent topics. For personalized query suggestion, some unique features of web search need to be incorporated in the generative process of the UPM. (1) As discussed in Section III, the session is the basic unit to satisfy a single information need. The words and URLs in the same session have coherent meanings. As a result, in the UPM, we constrain that the words and URLs in the same session are generated from the same topic. (2) Web search is essentially dynamic and the prominence of a topic changes over time. Hence, similar to [29], we introduce the Beta distribution to capture the temporal prominence of each topic in the UPM. (3) For a specific topic, each user has his or her own preferences in terms of word usage and URL clicking. For example, two users U_a and U_b are both interested in a topic about cars. U_a prefers Japanese cars, and she frequently submits search queries containing the word "Toyota" and clicks URLs referring to websites about Toyota cars. In contrast, U_b prefers American cars, and she frequently submits search queries containing the word "Ford" and clicks URLs referring to websites about Ford cars. In this case, for the same topic *Cars*, U_a and U_b have different perceptions of the importance of the words "Toyota" and "Ford". Therefore, in the generative process of UPM, we assume that each user has customized Multinomial distributions over words and URLs for a specific topic, in order to capture different users' intra-topic preference of the words and the URLs, which are essential for query suggestion personalization.

Given a set of documents prepared from query log, we can learn the appropriate values for the latent variables by alternating between optimizing the topic parameters given the hyperparameters, and optimizing the hyperparameters given the topic parameters. We adopt a sampling based approach for parameter estimation and start by factoring the complete likelihood as follows:

$$P(\mathbf{w}, \mathbf{u}, \mathbf{t}, \mathbf{z}|\alpha, \beta, \delta, \tau) = P(\mathbf{u}|\mathbf{z}, \delta)P(\mathbf{w}|\mathbf{z}, \beta)P(\mathbf{z}|\alpha)P(\mathbf{t}|\tau). \tag{18}$$

The probability $P(\mathbf{z}|\alpha)$ is given as:

$$P(\mathbf{z}|\alpha) = \left(\frac{\Gamma(\sum_{k=1}^{K}\alpha_k)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\right)^{D} \prod_{d=1}^{D} \frac{\prod_{k=1}^{K}\Gamma(C_{dk}^{DK}+\alpha_k)}{\Gamma(\sum_{k=1}^{K}(C_{dk}^{DK}+\alpha_k))}, \tag{19}$$

where C_{dk}^{DK} is the number of sessions that are assigned to topic k in document d. The probability $P(\mathbf{w}|\mathbf{z}, \beta)$ and $P(\mathbf{u}|\mathbf{z}, \delta)$ are as follows:

$$P(\mathbf{w}|\mathbf{z}, \beta) = \prod_{d=1}^{D}\prod_{k=1}^{K}\left(\frac{\Gamma(\sum_{w=1}^{W}\beta_{kw})}{\prod_{w=1}^{W}\Gamma(\beta_{kw})} \frac{\prod_{w=1}^{W}\Gamma(C_{kwd}^{KWD}+\beta_{kw})}{\Gamma(\sum_{w=1}^{W}(C_{kwd}^{KWD}+\beta_{kw}))}\right), \tag{20}$$

$$P(\mathbf{u}|\mathbf{z}, \delta) = \prod_{d=1}^{D}\prod_{k=1}^{K}\left(\frac{\Gamma(\sum_{u=1}^{U}\delta_{ku})}{\prod_{u=1}^{U}\Gamma(\delta_{ku})} \frac{\prod_{u=1}^{U}\Gamma(C_{kud}^{KUD}+\delta_{ku})}{\Gamma(\sum_{u=1}^{U}(C_{kud}^{KUD}+\delta_{ku}))}\right), \tag{21}$$

where C_{kwd}^{KWD} is the number of times that the word w are assigned to topic k in document d and C_{kud}^{KUD} is the number of times that the URL u are assigned to topic k in document d. The probability $P(\mathbf{t}|\tau)$ is as follows:

$$P(\mathbf{t}|\tau) = \prod_{d=1}^{D}\prod_{i=1}^{T_d}\left(\frac{(1-t_{di})^{\tau_{z_{di}1}-1}t_{di}^{\tau_{z_{di}2}-1}}{B(\tau_{z_{di}1}, \tau_{z_{di}2})}\right). \tag{22}$$

After combining the formula terms, applying Bayes rule and folding terms into the proportionality constant, we get the following conditional probability for Gibbs sampling:

$$\begin{aligned} P(z_i = k|\mathbf{z_{-i}}, \mathbf{w}, \mathbf{u}, \mathbf{t}, \alpha, \beta, \delta, \tau) \propto \\ \frac{C_{dk}^{DK}+\alpha_k}{\sum_{k'=1}^{K}(C_{dk'}^{DK}+\alpha_{k'})} \prod_{j=1}^{T_i}\left(\frac{(1-t_j)^{\tau_{z_j1}-1}t_j^{\tau_{z_j2}-1}}{B(\tau_{z_j1}, \tau_{z_j2})}\right) \\ \frac{\Gamma(\sum_{w=1}^{W}(C_{kwd}^{KWD}+\beta_{wk}))}{\Gamma(\sum_{w=1}^{W}(C_{kwd}^{KWD}+\beta_w+N_{iw}))} \prod_{w=1}^{W}\frac{\Gamma(C_{kwd}^{KWD}+\beta_{wk}+N_{iw})}{\Gamma(C_{kwd}^{KWD}+\beta_{wk})} \\ \left(\frac{\Gamma(\sum_{u=1}^{U}(C_{kud}^{KUD}+\delta_{uk}))}{\Gamma(\sum_{u=1}^{U}(C_{kud}^{KUD}+\delta_{uk}+N_{iu}))} \prod_{u=1}^{U}\frac{\Gamma(C_{kud}^{KUD}+\delta_{uk}+N_{iu})}{\Gamma(C_{kud}^{KUD}+\delta_{uk})}\right)^{I(X_i=1)}. \end{aligned} \tag{23}$$

Different from convention topic models such as Latent Dirichlet Allocation (LDA) [19], it is imperative to learn the hyperparameters of UPM. The information contained in the ϕ values of LDA is captured by the β and δ values of UPM. The complete likelihood $P(\mathbf{w}, \mathbf{u}, \mathbf{t}, \mathbf{z}|\alpha, \beta, \delta, \tau)$ is presented as follows:

$$P(\mathbf{w},\mathbf{u},\mathbf{t},\mathbf{z}|\alpha,\beta,\delta,\tau)=\prod_{d}((\frac{\Gamma(\sum_{k=1}^{K}\alpha_k)}{\prod_{k=1}^{K}\Gamma(\alpha_k)})\frac{\prod_{k=1}^{T}\Gamma(C_{dk}^{DK}+\alpha_k)}{\Gamma(\sum_{k=1}^{K}(C_{dk}^{DK}+\alpha_k))}$$
$$\prod_{d,k}\Big((\frac{\Gamma(\sum_{w=1}^{W}\beta_{wk})}{\prod_{w=1}^{W}\Gamma(\beta_{wk})})\frac{\prod_{w=1}^{W}\Gamma(C_{kwd}^{KWD}+\beta_{wk})}{\Gamma(\sum_{w=1}^{W}(C_{kwd}^{KWD}+\beta_{wk}))}$$
$$(\frac{\Gamma(\sum_{u=1}^{U}\delta_{uk})}{\prod_{u=1}^{U}\Gamma(\delta_{uk})})\frac{\prod_{u=1}^{U}\Gamma(C_{kud}^{KUD}+\delta_{uk})}{\Gamma(\sum_{u=1}^{U}(C_{kud}^{KUD}+\delta_{uk}))}\Big)\prod_{d,s,i}\big(p(t_{dsi}|\tau_{k_{dsi}})\big)\big). \tag{24}$$

Now we convert the formula above to its log-likelihood and get the following optimization formulas for each hyperparameter:

$$\alpha_{\cdot}^{'}=\arg\max\sum_{d,k}(\log\Gamma(C_{dk}^{DK}+\alpha_k)-\log\Gamma(\alpha_k))$$
$$+\sum_{d}(\log\Gamma(\sum_{k}\alpha_k)-\log\Gamma(\sum_{k}C_{dk}^{DK}+\alpha_k)). \tag{25}$$

$$\beta_{\cdot k}^{'}=\arg\max\sum_{d,k,w}(\log\Gamma(C_{kwd}^{KWD}+\beta_{wk})-\log\Gamma(\beta_{wk}))$$
$$+\sum_{d,k}(\log\Gamma(\sum_{w}\beta_{wk})-\log\Gamma(\sum_{w}C_{kwd}^{KWD}+\beta_{wk})). \tag{26}$$

$$\delta_{\cdot k}^{'}=\arg\max\sum_{d,k,u}(\log\Gamma(C_{kud}^{KUD}+\delta_{uk})-\log\Gamma(\delta_{uk}))$$
$$+\sum_{d,k}(\log\Gamma(\sum_{u}\delta_{uk})-\log\Gamma(\sum_{u}C_{kud}^{KUD}+\delta_{uk})). \tag{27}$$

Each equation above defines a vector, either $\alpha_{\cdot}^{'}$, $\beta_{\cdot k}^{'}$ or $\delta_{\cdot k}^{'}$. We use limited memory BFGS [30] to perform the maximization. We run Gibbs sampling to steady-state and then choose $\alpha_{\cdot}$, $\beta_{\cdot\cdot}$ and $\delta_{\cdot\cdot}$ to maximize complete likelihood $p(\mathbf{w},\mathbf{u},\mathbf{t},\mathbf{z}|\alpha,\beta,\delta,\tau)$. After the sampling on the user u_i's query log, the temporal parameters of topic k are updated as follows:

$$\tau_{k1}=\bar{t_k}(\frac{\bar{t_k}(1-\bar{t_k})}{s_k^2}-1), \tag{28}$$

$$\tau_{k2}=(1-\bar{t_k})(\frac{\bar{t_k}(1-\bar{t_k})}{s_k^2}-1), \tag{29}$$

where $\bar{t_k}$ and s_k^2 denote the sample mean and biased sample variance of topic k's timestamps.

The process above is repeated until the convergence of $\alpha_{\cdot}$, $\beta_{\cdot\cdot}$ and $\delta_{\cdot\cdot}$. Note that the UPM can take advantage of parallel Gibbs sampling paradigms such as the one proposed in [31] and it can scale to very large datasets. After processing each user's search history by the UPM, the dth user's search interests are represented by a topic vector $(\theta_{d1},\theta_{d2},...,\theta_{dk})$ where θ_{dk} is a real number that indicates the user's endorsement for the kth topic. The value of θ_{dk} is calculated based on the final states of the Markov chain after Gibbs sampling and the formula is given as follows:

$$\theta_{dk}=\frac{C_{dk}^{DK}+\alpha_k}{\sum_{k'=1}^{K}(C_{dk'}^{DK}+\alpha_{k'})}. \tag{30}$$

B. Online Personalization

According to the user d's profile, we can easily obtain the user's preference score of a suggestion candidate q via the following equation:

$$P(q|d)=\frac{\sum_{w\in q}\sum_{k}\frac{B(n_{wkq}+\beta_{wk})}{B(\beta_{wk})}\cdot\theta_{dk}}{\sum_{w\in q}1}, \tag{31}$$

where $B(\cdot)$ is the multidimensional Beta function. Based on the personalized preference score $P(q|d)$, we introduce a strategy to personalize the query suggestion list. We first rank the query candidates according to their personalized preference scores. Then, we aggregate this ranking list with the ranking list from the diversification component via Borda's method [32]. Note that the original ranking of the candidates from the diversification component is ranked via their relevance to the input query from the perspective of query affinity in the multi-bipartite representation while the ranking of the personalization component organizes the queries according to their relevance to a specific user's search preference. Therefore, we utilize ranking aggregation to combine these two perspectives of relevance and generate the final personalized query suggestion lists.

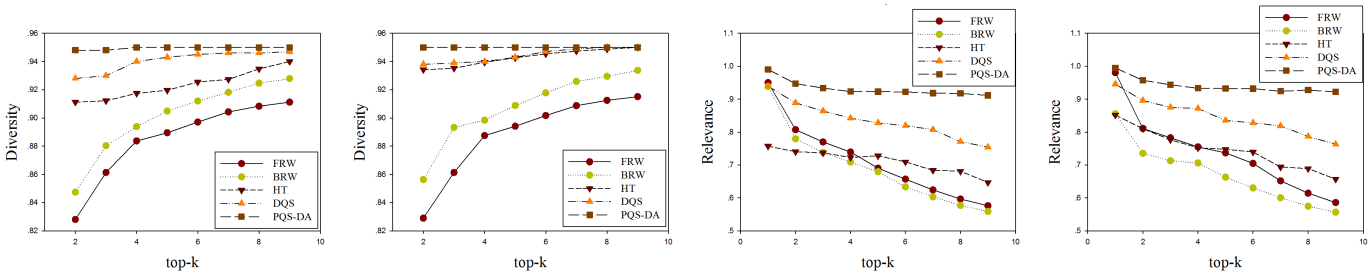
VI. Experiments

In this section, we compare the performance of PQS-DA with several the state-of-the-art query suggestion methods. In Section VI-A, we describe the experimental setup. Then we evaluate the effectiveness of the diversification component in Section VI-B and the performance of the personalization component in Section VI-C. Finally, in Section VI-D, we provide an efficiency analysis of PQS-DA.

A. Experimental Setup

We prepare the experimental dataset by a real-life query log obtained from a major commercial search engine. The query log contains search queries that were submitted by a total of 12,085 users. The raw query log data contain a lot of noises which will potentially affect the effectiveness of the query suggestion algorithms. Therefore, we conduct cleaning in a similar way as [33]. In order to enhance the reproducibility of the experimental results, we implement several well-known query suggestion methods as the baselines to gauge the performance of PQS-DA.

As shown in Fig. 1, the query suggestions of PQS-DA are generated by sequentially applying diversification and personalization. In order to reveal the details about the internal mechanism of PQS-DA, we will present the query suggestion results in a step-by-step fashion. In Section VI-B, we present the evaluation of the intermediate query suggestion results that are obtained by applying the diversification component. Then, in Section VI-C, we present the evaluation of the final query suggestion results that are generated by sequentially applying personalization for the intermediate query suggestion results obtained from the diversification component.



(a) Diversity (Raw) (b) Diversity (Weighted) (c) Relevance (Raw) (d) Relevance (Weighted)

Fig. 3: Evaluation of Query Suggestion After Diversification

B. Query Suggestion After Diversification

We first compare the quality of the intermediate query suggestion results obtained from the diversification component with those generated by four state-of-the-art query suggestion methods: the Forward Random Walk (FRW) [15], the Backward Random Walk (BRW) [15], the Hitting Time (HT) [14] and the Diversifying Query Suggestion (DQS) [6]. We utilize the *Diversity* and *Relevance* metrics proposed in [6] to gauge the performance of the methods under study. The two metrics can be automatically calculated by using query log and the Open Directory Project[2] (ODP). To make the paper self-contained, we detail the definitions of the two metrics in the following two paragraphs.

We first discuss the *Diversity* metric. Assume that q_i and q_j are two suggested queries. Let $\mathbf{P}(q_i)$ and $\mathbf{P}(q_j)$ denote the corresponding clicked Web page sets of q_i and q_j. The diversity of q_i and q_j, $d(q_i, q_j)$, is defined as:

$$d(q_i, q_j) = 1 - \frac{\sum_{m=1}^{M} \sum_{n=1}^{N} sim(p_{im}, p_{jn})}{M \times N}, \tag{32}$$

where $p_{im} \in \mathbf{P}(q_i)$, $p_{jn} \in \mathbf{P}(q_j)$, M and N are the sizes of $\mathbf{P}(q_i)$ and $\mathbf{P}(q_j)$ and $sim(p_{im}, p_{jn})$ measures the similarity between p_{im} and p_{jn}. Then the diversity of a query suggestion list L is further defined as:

$$D(L) = \frac{\sum_{i=1}^{|L|} \sum_{j=1, i \neq j}^{|L|} d(q_i, q_j)}{|L| \times (|L| - 1)}, \tag{33}$$

We proceed to discuss the *Relevance* metric. To measure the relevance of two queries, we can use a notion of similarity between the corresponding categories provided by ODP. Assume that q_i and q_j are two search queries and A_i and A_j are their corresponding ODP categories. We can measure the relevance of the two queries as the length of their longest common prefix $PF(A_i, A_j)$ divided by the length of the longest path between A_i and A_j. More precisely, the relevance metric is defined as:

$$R(q_i, q_j) = \frac{|PF(A_i, A_j)|}{\max(|A_i|, |A_j|)}. \tag{34}$$

In order to see the extent to which the edge weighing mechanism in Section III affects the performance, we conduct experiments on both the raw and the weighted multi-bipartite representations for PQS-DA while conduct experiments on both the raw and the weighted click graphs for the four baselines. To objectively reflect each method's performance in real-life scenario, we randomly select 10,000 testing queries from the search engine query log and report the average values of the results of all the testing queries.

The diversity evaluation of the top-k suggested queries are shown in Fig. 3(a) and (b). From these two figures, we can see that PQS-DA generates more diverse suggestions than FRW, BRW, HT and DQS. This result verifies the effectiveness of the diversification component in identifying diversified query suggestion candidates. The superiority of our diversification component is achieved by integrally applying the multi-bipartite representation and the cross-bipartite hitting time, which make much more search queries accessible and thus the result can cover much more facets of the original input query. We further show the relevance of the top-k suggested queries in Fig. 3(c) and (d). By comparing the top-1 relevance of the query suggestion candidate generated by these methods, we can see that PQS-DA is better at identifying the most relevant suggestion candidate than all the four baselines. The advantage of PQS-DA in finding the most relevant candidate verifies the effectiveness of our regularization framework which utilizes two constraints to integrate the information of the multi-bipartite representation and the search context. Besides the top-1 relevance, the relevance of PQS-DA still outperforms those of the baselines at other ranks, since the proposed multi-bipartite representation can effectively reduce the bias from a single bipartite and thus improves the overall relevance. The experimental results unequivocally show that using the multi-bipartite representation and the cross-bipartite hitting time can effectively discover highly relevant suggestion candidates. More importantly, the degradation of the relevance of PQS-DA is modest as k increases.

Through comparing the results of the raw multi-bipartite representation (i.e., Fig. 3(a) and (c)) with those of the weighted multi-bipartite representation (i.e., Fig. 3(b) and (d)), we observe that the weighted multi-bipartite representation is effective to improve the overall performance of PQS-DA. The insights gained from this comparison verify our assumption that differentiating the importance of different relations by

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

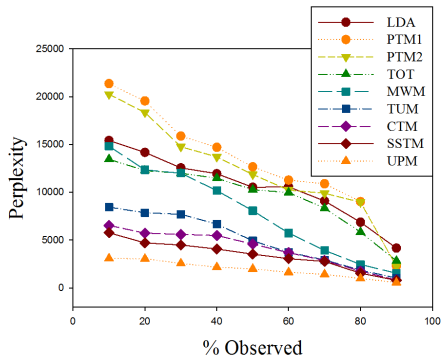


Fig. 4: Perplexity of Search Engine Query Log

their discriminative capabilities can better model the latent relations between queries in query log. Similarly, using the weighted click graph can also improve the performance of FRW, BRW, HT and DQS. In the rest of this paper, we utilize the weighted multi-bipartite representation and the weighted click graph to conduct the experimental evaluations. Astute reader may wonder why we do not apply the baselines on the multi-bipartite representation. The reason is that these baselines are primarily proposed for the click graph. Hence, transferring them to a multi-bipartite scenario is not trivial and a lot of tricky issues which are beyond the scope of this paper need to be solved. Therefore, we utilize the original methods described in literature as the baselines and this choice also contributes to the reproducibility of our experimental results.

C. Query Suggestion After Diversification and Personalization

In this subsection, we first quantitatively gauge the performance of the proposed UPM against several existing generative models. Then we present the experimental results of comparing PQS-DA against a wide range of existing query suggestion methods..

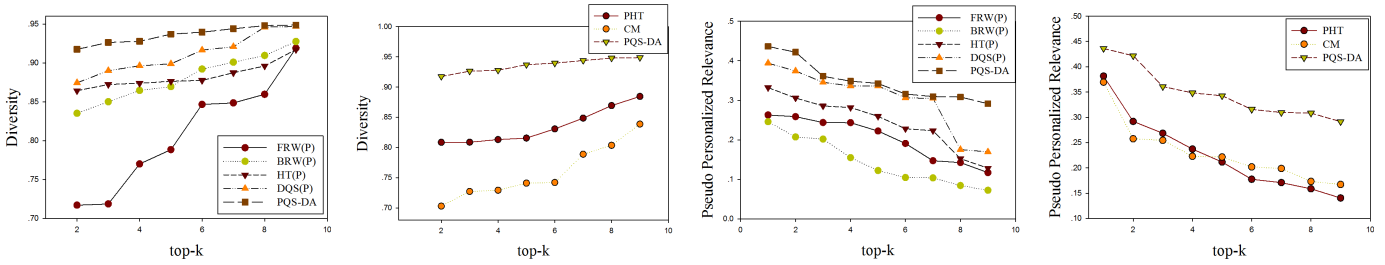
1) Quantitative Evaluation of UPM: We compare UPM with a variety of existing generative models. The first baseline is the widely used Latent Dirichlet Allocation (LDA) [19], the second one and third one, PTM1 and PTM2 are designed for search engine query log analysis [21], the fourth one is the Topic-Over-Time (TOT) model [29]. We also compare UPM with the Meta-word Model (MWM), the Term-URL Model (TUM), the Clickthrough Model (CTM) [34] and the SSTM model [35]. We compare the strength of the baseline models with the UPM in terms of how well the models can predict the remaining query words after observing a portion of the user's web search history. Suppose we observe the query words $w_{1:P}$ from a user's query log , this experiment aims to find out which model can provide better predictive distribution $p(w|w_{1:P})$ of the remaining query words. We use Equation (35) to calculate the perplexity of the remaining unseen data. The experimental results are presented in Fig. 4. UPM demonstrates the best performance with an average perplexity of 1933. The result shows the UPM is better at predicting the user's future search activity based on his or her search history. Compared with the other models, the UPM effectively integrates the query words, the URLs, the sessions, the timestamps as well as the users' preferences in word usage and URL clicking. Therefore, its generative process can better align with the latent structure of query log data and its result demonstrates better predicting performance.

$$Perplexity_{portion}(\mathcal{M}) = (\prod_{d=1}^{D}\prod_{i=P+1}^{N_d} p(w_i|\mathcal{M}, w_{1:P}))^{\frac{-1}{\sum_{d=1}^{D}(N_d-P)}}. \quad (35)$$

where $\mathcal{M}$ is the model learned from the training process and N_d is the length of document d.

2) Evaluation of Query Suggestion Lists: After quantitatively verifying the effectiveness of the UPM, we now evaluate whether the user profiles generated by UPM can achieve good personalization performance for query suggestion. In Section VI-B, the experimental results already show that the diversification component has achieved better performance against several baselines. A natural question arising here is whether PQS-DA keeps its advantages over the baselines after collectively applying diversification and personalization. To answer this question, we provide a detailed evaluation of the query suggestion results after applying both diversification and personalization. Although we can still use the same *Diversity* metric as that proposed in Section VI-B to evaluate the diversity of the query suggestion lists, the relevance after personalization is primarily based on a specific user's perception. Hence, we need to design a new metric for gauging relevance and propose *Pseudo Personalized Relevance* (PPR) as the metric.

For each user in the query log, we select ten most recent sessions as the *testing sessions* and consider the remaining ones as the *historical sessions*. Based on the *historical sessions*, we build the user profile via the UPM. Then we utilize the first query in each *testing session* as the input query, obtain the query suggestion candidates from the diversification component and personalize the ranking of the suggestion candidates via the personalization component. The PPR value is then calculated as the cosine similarity between the word vectors of the suggested query and the high-quality fields [36] (i.e., the HTML title and document title) of the clicked Web pages in the same session. A higher PPR value suggests a higher correlation between the suggested query and the specific user's information need. Hence, PPR is a valid metric to gauge the relevance of query suggestion from each search engine user's perspective. Another advantage of using PPR as a metric is that no human involvement is required during the evaluation and thus the evaluation can be conducted on a large scale dataset. We present the average of the PPR values over all testing sessions as the evaluation results. We first apply our personalization method to the results of the methods studied in the previous subsection and we add the suffix (P) to them to indicate that their results have been personalized by our personalization approach. We also compare PQS-DA with two existing personalized query suggestion methods: the



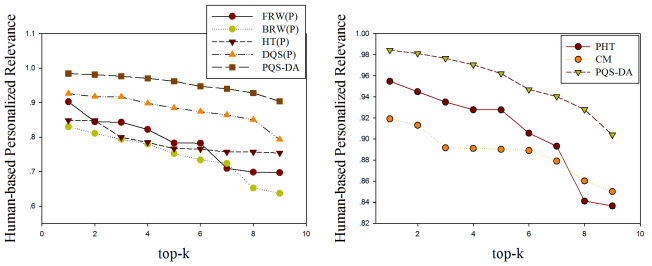
(a) Diversity (b) Diversity Comparison with PHT and CM (c) PPR (d) PPR Comparison with PHT and CM

Fig. 5: Evaluation of Query Suggestion After Diversification and Personalization

Personalized Hitting Time (PHT) [14] and the concept-based method (CM) [13].

We randomly select 5,000 users from the query log and calculate the average diversity and the average PPR value. The diversity after personalization is shown in Fig. 5(a) and (b). We can see that PQS-DA still maintains high diversity for all ranks after personalization. PQS-DA gets higher diversity than the personalized results of the methods studied in Section VI-B and this result again verifies the effectiveness of the diversification component, which can identify much more diversified results than the baselines. More importantly, we find that the personalization does not necessarily degrade the diversity of the query suggestion lists. The diversity of PQS-DA also significantly outperforms the methods that focus on personalized query suggestion such as PHT and CM. The results of the PPR evaluation are presented in Fig. 5(c) and (d). We observe that for the top suggested queries, PHT and CM demonstrate higher PPR than the other baselines, indicating that personalization is generally effective to improve the relevance of query suggestions for a specific user. Notably, the personalized results of PQS-DA outperform those generated by the baselines in terms of PPR. This shows that our personalization strategy can capture each user's preference and boost the overall relevance of query suggestion for each search engine user. The main advantage of PQS-DA is that it can maintain high relevance while achieving diversity, which is not achievable by the other two personalized query suggestion methods, PHT and CM.

Compared with utilizing the historical information stored in query log to evaluate the relevance of query suggestions, human experts can better emulate real web search process and give explicit relevance feedback. Thus, we design a metric named *Human Personalized Relevance* (HPR) to evaluate the effectiveness of the personalized ranking from the users' explicit feedbacks. Similar to [37], this experiment introduces human experts to conduct real web search in a period of four months. A web search middleware is implemented to record the experts' query log and suggest queries through different methods. The human experts are required to submit search queries to middleware and rate the suggested queries on a 6-point scale (0, 0.2, 0.4, 0.6, 0.8, and 1), where 0 means



(a) HPR (b) HPR Comparison with PHT and CM

Fig. 6: Evaluation of Query Suggestion After Diversification and Personalization (Human Personalized Relevance)

"totally irrelevant" and 1 indicates "entirely relevant". We report the average HPR in Fig. 6. We can see that PQS-DA generates query suggestions that better align with the users' latent information needs and it significantly outperforms the baselines with respect to the HPR. The results support our idea that a better query suggestion paradigm should be able to personalize the ranking of the suggested queries according to each individual user's preferences.

D. Efficiency Analysis of PQS-DA

Query suggestion is primarily deployed in an online fashion. Thus, an important issue is the efficiency of the suggestion process. We proceed to gauge the amount of time that PQS-DA typically consumes to generate the top-10 query suggestions for an input query. Most of the computational cost of PQS-DA is from the diversification component while the personalization component is very efficient, since we only need to reorganize the top-k suggestions, where k is usually set to a relatively small integer (e.g., 10 in our experiments). Fig. 7 shows the relative consumed time of different methods against the numbers of utilized queries. We can see that the efficiency of PQS-DA is comparable with the state-of-the-art diversification method DQS, and significantly better than that of CM, which relies on a large ontology for personalized query suggestion. More importantly, as the number of queries increases, the consumed time of PQS-DA increases moderately, indicating that PQS-DA can perform well even when the numbers of

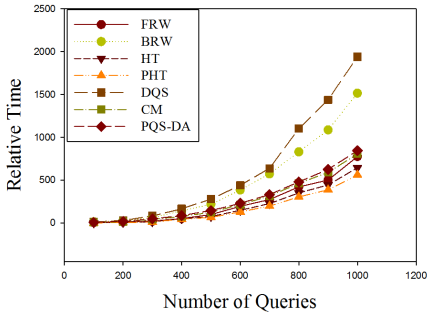


Fig. 7: Efficiency of Different Query Suggestion Methods

queries is large. The good efficiency of PQS-DA enables it to be online deployed and generate better query suggestions while keeping the consumed time comparable with the existing query suggestion methods.

VII. Conclusion

In this paper, we present a new query suggestion paradigm, *Personalized Query Suggestion with Diversity Awareness* (PQS-DA) to effectively integrate *diversification* and *personalization*, which are previously viewed as incompatible concepts, into one unified framework. We first propose a multi-bipartite representation to take full advantage of the rich information in the search engine query log. Based on the multi-bipartite representation, we design a regularization framework, which is effective to identify the most relevant suggestion candidate. Then an algorithm based on cross-bipartite hitting time is proposed to explore the latent structure of the multi-bipartite representation and identify the remaining suggestion candidates with diversity-awareness. Finally, the *User Profiling Model* (UPM) is proposed to personalize the ranking of the query suggestion lists for each individual user. Experimental results based on a large-scale query log clearly show that our approach outperforms several state-of-the-art methods in terms of diversity, relevance and ranking quality of the query suggestions. The insights obtained from this work pave the way for designing more effective query suggestion system in contemporary search engines.

Acknowledgment

This work is partially supported by GRF under grant numbers HKUST 617610. We also wish to thank the anonymous reviewers for their comments.

References

[1] M. Kato, T. Sakai, and K. Tanaka., "When do people use query suggestion?" *Information Retrieval*, 2013.
[2] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li, "Context-aware query suggestion by mining click-through and session data," in *SIGKDD*, 2008.
[3] S. Cucerzan and R. W. White, "Query suggestion based on user landing pages," in *SIGIR*, 2007.
[4] M. Kato, T. Sakai, and K. Tanaka, "Structured query suggestion for specialization and parallel movement: effect on search behaviors," in *WWW*, 2010.
[5] I. Szpektor, A. Gionis, and Y. Maarek, "Improving recommendation for long-tail queries via templates," in *WWW*, 2011.
[6] H. Ma, M. R. Lyu, and I. King, "Diversifying query suggestion results," in *AAAI*, 2010.
[7] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong, "Diversifying search results," in *WSDM*, 2009.
[8] R. L. Santos, J. Peng, C. Macdonald, and I. Ounis, "Explicit search result diversification through sub-queries," in *Advances in information retrieval*, 2010.
[9] D. Vallet and P. Castells, "Personalized diversification of search results," in *SIGIR*, 2012.
[10] C. Yu, L. Lakshmanan, and S. Amer-Yahia, "Recommendation diversification using explanations," in *ICDE*, 2009.
[11] ——, "It takes variety to make a world: diversification in recommender systems," in *EDBT*, 2009.
[12] Y. Song, D. Zhou, and L. He, "Post-ranking query suggestion by diversifying search results," in *SIGIR*, 2011.
[13] K. W. T. Leung, W. Ng, and D. L. Lee, "Personalized concept-based clustering of search engine queries," *TKDE*, 2008.
[14] Q. Mei, D. Zhou, and K. Church, "Query suggestion using hitting time," in *CIKM*, 2008.
[15] N. Craswell and M. Szummer, "Random walks on the click graph," in *SIGIR*, 2007.
[16] Y. Kim, J. Seo, and W. Croft, "Automatic boolean query suggestion for professional search," in *SIGIR*, 2011.
[17] J. Cui, H. Liu, J. Yan, L. Ji, R. Jin, J. He, Y. Gu, Z. Chen, and X. Du, "Multi-view random walk framework for search task discovery from click-through log," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, 2011.
[18] H. Deng, I. King, and M. R. Lyu, "Entropy-biased models for query representation on the click graph," in *SIGIR*, 2009.
[19] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, 2003.
[20] G. Doyle and C. Elkan, "Accounting for burstiness in topic models," in *ICML*. ACM, 2009.
[21] M. Carman, F. Crestani, M. Harvey, and M. Baillie, "Towards query log based personalization using topic models," in *CIKM*, 2010.
[22] D. Jiang, J. Vosecky, K. W.-T. Leung, and W. Ng, "G-wstd: A framework for geographic web search topic discovery," in *CIKM*. ACM, 2012.
[23] D. Beeferman and A. Berger, "Agglomerative clustering of a search engine query log," in *SIGKDD*, 2000.
[24] J. Huang and E. N. Efthimiadis, "Analyzing and evaluating query reformulation strategies in web search logs," in *CIKM*, 2009.
[25] D. Jiang, K. W. T. Leung, and W. Ng, "Context-aware search personalization with concept preference," in *CIKM*, 2011.
[26] X. Li, Y. Wang, and A. Acero, "Learning query intent from regularized click graphs," in *SIGIR*, 2008.
[27] G. Cormode, V. Shkapenyuk, D. Srivastava, and B. Xu, "Forward decay: A practical time decay model for streaming systems," in *ICDE*, 2009.
[28] D. Spielman and S. Teng, "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," in *STC*, 2004.
[29] X. Wang and A. McCallum, "Topics over time: a non-markov continuous-time model of topical trends," in *SIGKDD*, 2006.
[30] C. Zhu, R. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization," *TOMS*, 1997.
[31] D. Newman, P. Smyth, and M. Steyvers, "Scalable parallel topic models," *Journal of Intelligence Community Research and Development*, 2006.
[32] F. Schalekamp and A. van Zuylen, "Rank aggregation: Together were strong," *ALENEX*, 2009.
[33] X. Wang and C. X. Zhai, "Learn from web search logs to organize search results," in *SIGIR*, 2007.
[34] D. Jiang, K. W.-T. Leung, W. Ng, and H. Li, "Beyond click graph: Topic modeling for search engine query log analysis," in *DASFAA*, 2013.
[35] D. Jiang and W. Ng, "Mining web search topics with diverse spatiotemporal patterns," in *SIGIR*, 2013.
[36] Z. Bao, B. Kimelfeld, and Y. Li, "Automatic suggestion of query-rewrite rules for enterprise search," in *SIGIR*, 2012.
[37] K.-T. Leung, D. L. Lee, and W.-C. Lee, "Personalized web search with location preferences," in *ICDE*, 2010.