User Modeling in Search Logs via a Nonparametric Bayesian Approach

Hongning Wang[1], ChengXiang Zhai[1], Feng Liang[2]
[1]Department of Computer Science
[2]Department of Statistics
University of Illinois at Urbana-Champaign
Urbana IL, 61801 USA
{wang296,czhai,liangf}@illinois.edu

Anlei Dong, Yi Chang
Yahoo! Labs
701 First Avenue, Sunnyvale CA, 94089 USA
{anlei,yichang}@yahoo-inc.com

ABSTRACT

Searchers' information needs are diverse and cover a broad range of topics; hence, it is important for search engines to accurately understand each individual user's search intents in order to provide optimal search results. Search log data, which records users' search behaviors when interacting with search engines, provides a valuable source of information about users' search intents. Therefore, properly characterizing the heterogeneity among the users' observed search behaviors is the key to accurately understanding their search intents and to further predicting their behaviors.

In this work, we study the problem of user modeling in the search log data and propose a generative model, dpRank, within a non-parametric Bayesian framework. By postulating generative assumptions about a user's search behaviors, dpRank identifies each individual user's latent search interests and his/her distinct result preferences in a joint manner. Experimental results on a large-scale news search log data set validate the effectiveness of the proposed approach, which not only provides in-depth understanding of a user's search intents but also benefits a variety of personalized applications.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

User modeling, search log mining, nonparametric Bayesian

1. INTRODUCTION

Among various ways of mining search engine logs, such as studying the distribution of queries and query categories that users frequently search for [11] and recognizing temporal dynamics of search topics [15], identifying individual user's search intent from search logs is of particular importance. Given the huge diversity of search engine users' information needs, accurate understanding of a user's search interests and preferences is arguably one of the most critical research challenges in information retrieval studies [9, 24, 26]. It is, however, impractical to ask users to give explicit feedback about their search intents; and thus studying how to effectively mine such diverse information needs of users from the search logs becomes a valuable research direction.

Search log mining has drawn a considerable amount of attention in research. White and Drucker investigated the variability in people's search behaviors, and suggested two typical types of searchers: navigators and explorers, whose search interaction pattern is highly consistent or highly variable [26]. The studies in [1, 13] showed that implicit feedback extracted from the aggregated users' clickthroughs is reasonably accurate to interpret the generic ranking preferences over the retrieved documents. Although much effort has been devoted to mining general search behaviors of users, little attention has been paid to identifying individual user's search preferences from the search log data. Liu et al. [16] profiled a user's search interest by classifying his/her queries into predefined categories and tailored the ranking model for each individual user accordingly. Sontag et al. [23] proposed a probabilistic model to estimate topic-based profiles for both documents and users, and used such profiles to calibrate the relevance estimation of retrieved documents for each individual user, i.e., search personalization.

A major limitation of existing work is that the analysis of a user's query distribution and that of his/her associated click behaviors are *isolated*. We should note that both queries and clickthroughs convey useful clues about a user's search intent. Taking news search as an example, a user who frequently issues celebrity queries has distinct search interest from another user who often searches for the past political news events. Such difference can be captured by modeling the query distributions of different users and be used to adjust the search results. However, it is insufficient to only model a user's query distribution. For example, even for the same celebrity query, some users tend to click on the most recent news exhibiting his/her unique preference on the latest report about the celebrities, while other users might only click on the reports from authority sources. Therefore, in order to provide a comprehensive understanding of a user's search intent, we must capitalize on the following two aspects simultaneously: 1) user's search interest, i.e., the distribution of queries a user frequently enters; 2) user's unique result ranking preference, i.e., the probability of observing a particular click pattern from the user. Such a joint analysis, however, is missing in the previous work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WSDM'14, February 24–28, 2014, New York, New York, USA.
Copyright 2014 ACM 978-1-4503-2351-2/14/02 ...$15.00.
http://dx.doi.org/10.1145/2556195.2556262.

In this paper, we study the problem of user modeling in search logs, and develop a generative model, dpRank, in the non-parametric Bayesian framework to enable *joint* modeling of query distributions and clickthrough patterns in each individual user. To capture the heterogeneity among all search users, we introduce a novel concept of *latent user groups*. In detail, each latent user group possesses a unique distribution over the queries: for instance, in news search, one group of users may frequently issue celebrity queries, while another group may only look for breaking news events. Meanwhile, the click preference of each latent user group is portrayed by the corresponding relative importance of the ranking features, which leads to distinct click patterns over the returned documents. For example, for the same input query, some group of users may want high authority websites (i.e., larger weight on the pagerank score), while the other groups may prefer the documents better matched with their queries (e.g., larger weight on the relevance features such as BM25). In order to characterize the heterogeneity among the queries and clickthroughs in a single user's search history, we model each user's search intent as a mixture over those latent user groups, where the mixing proportion governs each user's unique search behavior pattern.

By postulating such generative assumptions, homogeneity of users' aggregated search behaviors is captured by the latent user groups, while the variability of individual user's interaction patterns is captured by modeling each user as a mixture over the latent user groups. As a mixture model, the number of mixture components, which is usually manually set, determines the granularity of the model. However, due to the dynamic nature and scale of search logs, it is infeasible for us to manually exhaust the optimal number of the latent user groups. An appealing alternative is to adopt a fully data-driven approach, i.e., postulating Dirichlet process priors [8] over the mixture components, to exploit the clustering property of users' search behaviors. In dpRank, we assume the parameters characterizing each latent user group are drawn from a shared Dirichlet process; accordingly, the mixing proportion in each user over the latent groups is drawn from another Dirichlet process to accommodate such infinite number of components. As a result, the number of latent user groups can be automatically determined based on the data characteristics.

Knowledge discovered by dpRank from the search logs can be useful for many applications. Characteristics of latent user groups, such as those different criteria for document ranking, provide an in-depth understanding of search engine users. The estimated mixing proportion of latent user groups in each user can be used as a proxy to measure similarity between users for collaborative filtering based tasks, or for many other personalized applications.

To investigate the effectiveness of the proposed dpRank model for user modeling in search logs, we collected a large collection of search logs from a commercial news search engine. Experimental results validated our basic hypothesis that joint modeling of users' search behaviors, i.e., their queries and clicks, in search logs provides a more comprehensive and accurate understanding of users' search intents than those isolated modeling approaches. And the mined knowledge about users can benefit a variety of applications, e.g., search result ranking, collaborative query recommendation, and document re-ranking, to improve task performance. In addition, the proposed dpRank model is general; and thus it can be directly applied to many other types of search logs, e.g., web search and blog search logs.

2. RELATED WORK

A great amount of effort has been devoted to turning the massive search usage data in search engine logs into actionable knowledge: Jansen et al. [11] explored search logs to analyze the types of information users search for and how people do search within a web search engine; Kulkarni et al. studied the dynamics of search topics over time [15]; White and Drucker investigated variability in people's search interaction behaviors [26]. A comprehensive survey of research about search log mining can be found in [22].

While many studies have been performed on mining general interests of users from search logs, little work has been done on discovering knowledge about individual users. The related work in this direction can be summarized into two major categories.

One type of studies focuses on the semantic interpretation of queries. Liu et al. [16] categorized a user's queries into a set of ODP categories and tailored the ranking results for each individual user according to such identified search interest. Rose and Levinson [19] grouped the queries from the same user into search tasks, in which the user exhibits consistent information need.

Another type of work aims at accurately modeling and interpreting a user's ranking preferences from their click feedback. Joachims et al. [13] and Agichtein et al. [1] analyzed the reliability of implicit feedback generated from clickthrough data and designed a set of click heuristics to extract relative search result preferences from clicks in the search logs. Probabilistic click models [5, 7, 28] have been proposed for modeling user click behaviors and extracting intrinsic relevance information from the clickthroughs.

However, the aforementioned work isolated the analysis of query distribution in a user and that of associated click behaviors. Our approach extends the previous work to enable *joint* modeling of the queries and click behaviors of individual users. By postulating generative assumptions about a user's search behaviors, the proposed dpRank model identifies each individual user's latent search interests and his/her distinct ranking preferences in a unified way, which is necessary to capture subtle preference variations of user interests.

In terms of methodology, the most similar work to ours is Bian et al.'s "divide-and-conquer" approach for web search [3] and Giannopoulos et al.'s method for learning to rank user intent [10], although they isolated the analysis of users' query distribution and click patterns. In [3], Bian et al. represented the queries by the aggregated ranking features from pseudo-feedback documents, and clustered the queries into groups by the k-means algorithm. Within each identified query group, a separate RankSVM [12] model was trained to satisfy the group-specific ranking requirement. Giannopoulos et al. [10] took a similar procedure to cluster queries and estimate independent RankSVM models for each query group. In particular, they represented the queries by the weight vectors from RankSVM models learned in each query. However, in both of these two methods, query is the major focus of study, and they grouped the queries either by the derived query features or click patterns. But neither of them model the users, who generated the queries and clicks.

3. METHOD

In this work, we propose a generative model, dpRank, to enable *joint* modeling of users' queries and clicks in search logs. Given a collection of search logs, including the queries from users, documents returned by a search engine, and the

corresponding clicks from the users, dpRank aims to identify each individual user's latent search interests and their distinct ranking preferences in a unified way. An efficient Markov chain Monte Carlo sampling algorithm is developed to estimate the posterior distribution of latent variables in the proposed dpRank model.

3.1 Problem Definition

Denote a collection of users as $U = \{u_1, u_2, \ldots, u_n\}$ in the given search log data, in which each user u_i is associated with a set of queries $Q^{u_i} = \{q_1^{u_i}, q_2^{u_i}, \ldots, q_m^{u_i}\}$. Each query $q_j^{u_i}$ is represented by a T-dimensional vector of query features, e.g., query length and query frequency in the collection; and $q_j^{u_i}$ is associated with a list of retrieved documents together with user clicks $D_j^{u_i} = \{(d_{j1}^{u_i}, y_{j1}^{u_i}), (d_{j2}^{u_i}, y_{j2}^{u_i}), \ldots, (d_{jl}^{u_i}, y_{jl}^{u_i})\}$, where $d_{jl}^{u_i}$ is a V-dimensional vector of ranking features describing the relevance quality of document $d_{jl}^{u_i}$ to query $q_j^{u_i}$, and $y_{jl}^{u_i}$ indicates if the document $d_{jl}^{u_i}$ is clicked by user u_i (e.g., 1 for the clicked documents and 0 for the skipped ones).

The problem of user modeling in search log data is to characterize different users' search intents by analyzing the association patterns of their issued queries and corresponding clicks. Using the language of probabilistic models, one needs to estimate the joint distribution of queries and click preferences in user u, i.e., $p(Q, D, u)$. In-depth knowledge about individual users can then be discovered by posterior inference on various conditional distributions. For example, $p(Q|u)$ reveals a user's search interest via his/her query distribution; and $p(D|Q, u)$ reflects a user's unique ranking preferences over the returned documents.

3.2 Generative Story in dpRank

The basic idea behind the proposed dpRank model is to define the joint probability $p(Q, D, u)$ in a generative way. To model the diversity among different users' search interests (i.e., heterogeneity among the distribution of queries) and ranking preferences (i.e., heterogeneity among users' clicks in search engine returned documents), we introduce the concept of latent user groups. We assume that to issue a query to the search engine, the user would first select a particular user group he/she belongs to, from which he/she wound then pick a query according to its generation probability within this group. Once the search engine returns a list of documents for the input query, the user will carefully examine those documents, and selectively click the ones that match his/her result preferences for this type of queries specified in the latent user group. As a result, by modeling each individual user's search intents as a mixture over those latent user groups, patterns of users' search behaviors can be automatically identified.

3.3 User Modeling via dpRank

In dpRank, each latent user group indexed by c refers to a homogeneous generative model for $p(q, D)$. Within the user group c, we employ a T-dimensional multivariate Normal distribution to model the generation of queries, i.e., $p(q_j^{u_i}) \sim N(\mu_c, \sigma_c^2 \mathbf{I})$, where we assume a diagonal covariance matrix. It is known that users' clicks may be biased by the displayed ranking positions [5, 7, 28], but the relative preferences of clicked documents over skipped ones are reasonably accurate [1, 13]. Therefore, to learn a user's result ranking preference reflected in his/her clicks, we employ a pairwise preference model on the returned document list, i.e., $p(D_j^{u_i}|q_j^{u_i}) = \prod_{y_{js}^{u_i} > y_{jt}^{u_i}} p(y_{js}^{u_i} > y_{jt}^{u_i}|d_{js}^{u_i}, d_{jt}^{u_i})$. Specifically, we use a logistic function to model such pairwise ranking preferences as,

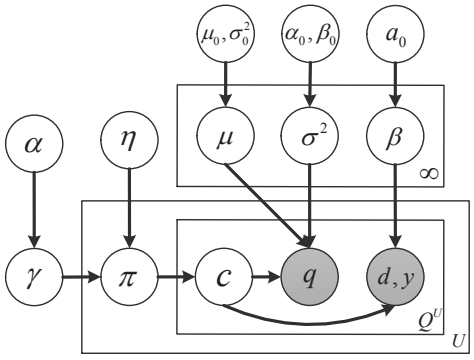

Figure 1: Graphical model representation of dpRank. Light circles denote the latent random variables, and shadow circles denote the observed random variables. The outer plate indexed by U denotes the users in the given search log, the inner plated indexed by Q^U denotes the queries and the associated documents and clicks from user u, and the upper plate denotes the parameters for the countably infinite number of latent user groups in the collection.

$$p(y_{js}^{u_i} > y_{jt}^{u_i}|d_{js}^{u_i}, d_{jt}^{u_i}, \beta_c) = \frac{1}{1 + \exp\big(-\beta_c^{\mathsf{T}}(d_{js}^{u_i} - d_{jt}^{u_i})\big)} \quad (1)$$

where β_c is a V-dimensional weight vector indicating the relative importance of ranking features in latent group c.

Based on these specifications, each latent user group can be fully characterized by a set of parameters, $\theta_c = (\mu_c, \sigma_c^2, \beta_c)$. The joint distribution of query $q_j^{u_i}$ and corresponding clickthroughs in $D_j^{u_i}$ is thus modeled as,

$$p(q_j^{u_i}, D_j^{u_i}|\theta_c) = p(q_j^{u_i}|\mu_c, \sigma_c^2) \prod_{y_{js}^{u_i} > y_{jt}^{u_i}} p(y_{js}^{u_i} > y_{jt}^{u_i}|d_{js}^{u_i}, d_{jt}^{u_i}, \beta_c) \quad (2)$$

given that they are generated from a particular user group c. Note that we do not model the generation of returned documents for query $q_j^{u_i}$, but just the corresponding click preferences conditioned on them, since the documents are returned by the search engine and thus are independent of the users given the query (i.e., without considering the personalization feature in search engines).

To construct a fully generative model, we need to specify the generation of the parameter θ_c for each latent user group. In dpRank, in order to avoid manually specifying the configuration of θ_c, e.g., how many unique θ_c should be used for a given collection, we assume θ_c itself is also a random variable and is drawn from a Dirichlet process (DP). A Dirichlet process $DP(G_0, \alpha)$ with a base distribution G_0 and a scaling parameter α is a distribution over distributions [8]. An important feature of the DP is that draws from a DP often share some common values, and therefore naturally form clusters. The number of unique draws, i.e., the number of clusters, varies with respect to the data and therefore is random, instead of being pre-specified.

As a result, in dpRank, the global distribution of queries and corresponding clicks in a given collection of search logs distributes as a DP, and it can then be expressed using a

stick-breaking representation [21]:

$$p(Q^U, D^U) = \sum_{k=1}^{\infty} \gamma_k \delta_{\theta_k} \quad (3)$$

where $\theta_k \sim G_0$, δ_θ is a distribution of (Q^U, D^U) concentrated at θ and $\gamma = (\gamma_k)_{k=1}^{\infty} \sim Stick(\alpha)$ represents the global distribution of mixture components in the whole collection. The stick-breaking process $Stick(\alpha)$ for γ is defined as: $\gamma'_k \sim Beta(1, \alpha), \gamma_k = \gamma'_k \prod_{t=1}^{k-1}(1 - \gamma'_t)$. Specifically, we further assume that each dimension of parameters for generating the query features, i.e., (μ_k, σ_k^2), and each dimension of the linear weights for the ranking features, i.e., β_k, are independently drawn from their corresponding prior distributions defined in G_0: for $t = 1 \dots T$, $\mu_{kt} \sim N(\mu_0, \sigma_0^2)$, $1/\sigma_{kt}^2 \sim Gamma(\alpha_0, \beta_0)$ and for $v = 1 \dots V$, $\beta_{kv} \sim N(0, a_0^2)$.

The global distribution defined in Eq (3) captures the homogeneity of users' search behaviors for the whole population (characterized by the group parameter $\boldsymbol{\theta}$), but tells us little about each individual user. Therefore, we need to inject the information about the variabilities of individual users to the dpRank model.

Because the global distribution defined in Eq (3) only has finite supports at the points of $\theta = \{\theta_k\}_{k=1}^{|Q^U|}$, the distribution of mixture components in each user will only have support at those points as well. Based on Eq (2), the joint probability of queries and clicks in user u_i can be naturally modeled as,

$$p(q_j^{u_i}, D_j^{u_i}) = \sum_{k=1}^{\infty} \pi_{u_i k} p(q_j^{u_i}, D_j^{u_i} | \theta_k) \quad (4)$$

where $\pi_{u_i} = (\pi_{u_i k})_{k=1}^{\infty} \sim DP(\gamma, \eta)$, i.e., we introduce another layer of DP with the global proportion γ as the base distribution for modeling the mixing of latent user groups in each user.

The benefit of such an approach is two-fold. From a modeling aspect, this modeling approach reveals the information of users both at the aggregated level (i.e., the shared latent user groups) and also at the individual level (i.e., the user-specific mixing proportions). From a learning aspect, it allows information sharing across different users, and therefore makes learning more efficient.

Based on the above discussions, we can formalize the generation process of queries and clickthroughs in search logs defined in dpRank using the language of probabilistic graphical models in Figure 1.

3.4 Posterior Inference

To apply the proposed model for user modeling in search logs, we need to calculate the posterior distribution of latent variables. In dpRank, the latent variables of interest are: (μ, σ^2) that characterize the generation of queries in a latent user group; β that depicts a user's relative emphasis over the ranking features in a latent user group; and π that profiles an individual user's search intent over the global latent user groups.

Following the sampling scheme proposed in [17, 25], we develop an efficient Markov chain Monte Carlo (MCMC) algorithm, or more precisely, a Gibbs sampler, to perform the posterior inference for dpRank. To facilitate later description of the employed sampling scheme, we assume that at a particular sampling step, there are in total K active components (i.e., components that associate with data instances), and by permuting the indices, we can index them from 1 to K. We define $\gamma_e = 1 - \sum_{k=1}^{K} \gamma_k$ to be the total mixing proportion for the remaining inactive components.

The key to perform Gibbs sampling is to derive the full conditional distribution for each latent variable. In the following, we will discuss about detailed derivation of the conditional posterior distributions of interest.

• **Sampling c_{uj}:** Given user u, the conditional distribution of $\mathbf{c}_u$ and π_u is given by

$$p(\pi_u, \mathbf{c}_u | \gamma, \boldsymbol{\theta}, q^u, D^u, G_0, \alpha, \eta) \quad (5)$$
$$\propto \; p(\pi_u | \gamma, \eta) \prod_j p(q_j^u, D_j^u | \theta_{c_{uj}}) p(c_{uj} | \pi_u)$$

To sample the latent user group assignment c_{uj} for a particular query q_j^u from user u, we can integrate over π_u in Eq (5) analytically because of the conjugacy between the Dirichlet distribution $p(\pi_u | \gamma, \eta)$ and the multinomial distribution $p(c_{uj} | \pi_u)$, and get the conditional probability of c_{uj} given γ and η as:

$$p(c_{uj} | \gamma, \eta) = \frac{\Gamma(\eta)}{\Gamma(\eta + m_u)} \prod_{k=1}^{K} \frac{\Gamma(\eta\gamma_k + m_{uk})}{\Gamma(\eta\gamma_k)} \quad (6)$$

where m_u is the number of queries user u has and m_{uk} is the number of queries in user u assigned to group k.

As a result, the conditional probability of $c_{uj} = k$ given query q_j^u, corresponding clicks in D_j^u and other latent variables is,

$$p(c_{uj} = k | q_j^u, D_j^u, \boldsymbol{\theta}, \gamma, \eta) \propto (\eta\gamma_k + m_{uk}^{-uj}) p(q_j^u, D_j^u | \theta_k) \quad (7)$$

where k takes values in $\{1, \dots, K, e\}$, m_{uk}^{-uj} is the number of queries from user u assigned to group k except the current query q_j^u, and θ_e is an auxiliary component drawn from the base distribution G_0.

If the sampling result of c_{uj} makes an active component associate with no observations, we need to remove it from the list of active components, set $K = K - 1$ and update γ_e accordingly; and if the sampling result of c_{uj} is e, we need to append θ_e to the list of active components, and draw a new global proportion γ_{K+1} for it. To achieve so, we sample $b \sim Beta(1, \alpha)$ and set $\gamma_{K+1} = b\gamma_e$, $\gamma_e = (1 - b)\gamma_e$.

• **Sampling γ:** Following the sampling scheme introduced in [25], we employ the same auxiliary variable method for sampling the global mixture proportion γ.

By expanding the ratios of Gamma functions in Eq (6) and including the Stick-breaking prior for γ, we get the posterior distribution for γ conditioned on the auxiliary variable $\mathbf{h}$ as:

$$p(\gamma | \alpha, \mathbf{h}) \propto \gamma_e^{\alpha - 1} \prod_{k=1}^{K} \gamma_k^{\sum_{i=1}^{n} h_{u_i k} - 1} \quad (8)$$

where $h_{u_i k}$ is a random variable based on the occurrence of group k in user u_i, which takes value from 1 to $m_{u_i k}$. And the corresponding distribution of $\mathbf{h}$ given $\mathbf{c}$ and γ is,

$$p(h_{u_i k} = h | \mathbf{c}, \gamma) \propto s(m_{u_i k}, h) (\eta\gamma_k)^h \quad (9)$$

where $s(m, h)$ is the unsigned Stirling numbers of the first kind.

Due to the conjugacy between γ and $\mathbf{c}$, the posterior distribution of γ given $\mathbf{h}$ still follows a Dirichlet distribution with parameters of α and $\mathbf{h}$ in Eq (8); and $h_{u_i k}$ acts as an estimated number of occurrence of group k in user u_i in Eq (9) (more explanation of $\mathbf{h}$ can be found in [25]). Based on Eq (8) and Eq (9), iterative sampling between γ and $\mathbf{h}$ is performed to get the posterior samples for γ.

• **Sampling $\boldsymbol{\theta}$:** Conditioned on the group assignments $\mathbf{c}$ of all the users in the collection, the posterior distributions of $\boldsymbol{\theta}$ are determined by the queries (from all users) assigned to each group, namely,

$$p(\theta_k|\mathbf{c},\mathbf{Q},\mathbf{D},G_0) \propto p(\theta_k|G_0) \prod_{c_j^u=k} p(q_j^u, D_j^u|\theta_k) \quad (10)$$

Because of the Normal-Normal and Normal-Gamma conjugacy, the posterior distributions for μ_k and σ_k^2 can be analytically calculated. Since we have assumed each dimension of μ_k and σ_k is independently drawn from the priors, we can calculate the posteriors of them separately. To simplify the notations, we denote n_k as the total number of queries assigned to group k given $\mathbf{c}$, and $\lambda_{kt} = 1/\sigma_{kt}^2$, i.e., the precision of a Normal distribution. As a result, we have,

$$p(\mu_{kt}|\mathbf{c},\mathbf{q}) = N(\mu_{n_kt}, \sigma_{n_kt}^2) \quad (11)$$

where $\sigma_{n_kt}^2 = \frac{1}{\frac{n_k}{\sigma_{kt}^2}+\frac{1}{\sigma_0^2}}$ and $\mu_{n_kt} = \sigma_{n_kt}^2\big(\frac{\mu_0}{\sigma_0^2} + \frac{\sum_{c_j^u=k} q_{jt}^u}{\sigma_{kt}^2}\big)$. And,

$$p(\lambda_{kt}|\mathbf{c},\mathbf{q}) = Gamma(\alpha_{n_kt}, \beta_{n_kt}) \quad (12)$$

where $\alpha_{n_kt} = \alpha_0 + \frac{n_k}{2}$ and $\beta_{n_kt} = \beta_0 + \frac{1}{2}\sum_{c_j^u=k}(q_{jt}^u - \mu_{kt})^2$.

Based on the posterior distributions specified in Eq (11) and Eq (12), iterative sampling is performed for the posterior update of μ_k and σ_k^2 in each user group.

However, because there is no proper conjugate prior for the linear weight β in the click preference modeling module as defined in Eq (1), the posterior distribution of Eq (10) with respect to β cannot be analytically calculated. In this work, we appeal to the Metropolis Hasting sampling algorithm with Hamiltonian dynamics [18] to perform β's posterior update. Briefly, this sampling scheme uses Hamiltonian dynamics to produce proposals for the Metropolis Hasting algorithm. Because the proposal is generated based on the gradient of energy function defined by the target distribution (as in Eq (10)), it provides faster exploration of the state space comparing to simple random-walk proposals.

• **Gibbs sampling procedure:** the Gibbs sampling for the proposed dpRank model can be performed by alternating the sampling steps for $\mathbf{c}$, γ and $\boldsymbol{\theta}$ described above iteratively. We should note that the sampling of $\mathbf{c}$ is performed for every query from the users in the given collection, and the samplings of γ and $\boldsymbol{\theta}$ are performed for the whole collection given the newly updated latent user group assignments $\mathbf{c}$. As an MCMC algorithm, samples from the sampling chain are usually correlated with nearby samples. To collect independent samples, we only kept the samples from every five iterations, i.e., thinning the sampling chain. Besides, samples from the beginning of the sampling chain (i.e., the burn-in period) may not accurately represent the desired distribution. We only kept the posterior samples after the burn-in period (in our experiment, we discarded the first 20% of samples).

3.5 Discussion

Compared to the two-step query-clustering-based ranking methods introduced in [3, 10], the proposed dpRank model is user-centric and for the first time enables the joint modeling of a user's search query distribution and corresponding click behaviors. As explicitly expressed in Eq (7), the assignment of latent user group for a particular query q^u in user u is determined by two factors: 1) the current proportion of the latent user groups in u; and 2) the likelihood of observing the given query and corresponding clicks in a candidate user group. As a result, the chosen user group for query q^u in user u should not only well explain both of the query q^u and clicks in D^u, but also closely align with his/her general search interest, i.e., the proportion of user group assignments in his/her rest queries. In [3, 10], no user-specific information is considered for clustering the queries, and those clusterings are solely determined by either the queries or clickthrough patterns independently. In [3], only the characteristics of queries were modeled based on the derived query features. As a result, users who issued the same query but clicked on different documents will be assumed to have the same search intent and served with the same ranking result. In [10], to capture users' distinct click preferences, independent ranking models are estimated for each query. But the same user's click preference in different queries cannot be modeled, and sparsity becomes a serious issue for estimating ranking models in each individual query.

In dpRank, by inferring the posterior distribution of the latent variables in a given collection of search logs, important knowledge about a user's search intent can be discovered. First, the posterior distribution of parameter $\theta = (\mu, \sigma^2, \beta)$ reveals the distribution of queries and clickthrough patterns in a particular user group. For instance, by analyzing β, we can understand users' relative emphasis of the ranking features in each user group. Based on such knowledge, we can directly influence the general ranking model training (e.g., create new ranking features for each user group) or estimate a separate ranking model if the ranking criterion in this specific group is very different from that in the global ranking model. Second, the posterior distribution of latent user group assignments in user u, i.e., $p(\pi_u|q^u, D^u, \gamma, \eta)$, summarizes an individual user's search intent in history. This distribution can serve as a compact user profile for many personalized applications. For example, we can compute the similarity between users based on it for collaborative query recommendation and document re-ranking [20].

Besides, the proposed user modeling method can be flexibly applied in the real dynamic search environment, where new users or new queries keep emerging. As it is a Bayesian model, we do not need to re-estimate the model for the whole data set, but just to perform the posterior update discussed in Section 3.4 on the newly arrived data. This property renders dpRank a wider application scenario in practice.

4. EXPERIMENTAL RESULTS

To evaluate the proposed method for user modeling in search logs, we conducted experiments on a large-scale search log data set. First, we qualitatively demonstrated the knowledge discovered by dpRank from the search logs, e.g., distribution of queries in the identified user groups and the corresponding result ranking preferences. Then, we performed a series of quantitative evaluations to validate our main hypothesis in this work that joint modeling of users' search behaviors could provide a more comprehensive and accurate understanding of users' search intents than those isolated approaches in a variety of scenarios.

4.1 Data Set

A large set of search logs are collected from Yahoo! news search engine[1] in a two-months period, from late May to late July 2011. During this period, a subset of users are randomly selected and all their search activities are collected, including the anonymized user ID, query string, timestamp,

[1] `http://news.search.yahoo.com`

top 10 returned documents and the corresponding user clicks. In this data set, each document is represented by 65 basic ranking features, such as document age, website authority, and query matching in document title and body.

Since we are modeling users in a per-user basis, we must ensure every user has his/her own training and testing queries in our evaluation. To achieve so, we ordered the queries in each user by their timestamps, and used the first 60% queries for training and the rest 40% for testing in each user.

Simple pre-processing is applied on this data set: 1) filter out the queries without clicks; 2) discard the users with less than two queries; 3) normalize the ranking features by their mean and standard deviation estimated in the training queries, i.e., z-score. After the pre-processing steps, there are 42,642 users with 288,786 queries selected in this collection, among which 189,757 queries are used for training and 99,029 queries for testing. The basic statistics about this evaluation search log set are summarized in Table 1.

Table 1: Basic statistics of evaluation search log set.

#user	42,642	#query	288,786
#document	2,483,486	query/user	6.77±6.21
click/query	1.22±0.57	uni-user/query	17.9±121.7

However, in this data set, there is no predefined query feature. We appealed to the method proposed in [3] to create query features from the search engine returned documents. In detail, we computed the mean vector of ranking features in the documents associated with the given query as its query feature representation.

In dpRank, there is a set of hyper-parameters to be determined in the prior distribution G_0. *Uninformative prior* is often used in Bayesian models when the number of observations is large. In our experiments, we found that the performance of dpRank was not sensitive to the setting of such hyper-parameters due to the large volume of observations we have. Therefore, in the following experiments, we fixed the hyper-parameters in G_0: $\mu_0 = 0$, $\sigma_0^2 = 0.5$, $\alpha_0 = 1.0$, $\beta_0 = 0.5$, $a_0 = 1.0$, $\alpha = 1.0$, $\eta = 0.1$. And we set the total number of Gibbs sampling iterations to be 1000.

4.2 Qualitative Evaluation

The posterior distribution of latent variables in dpRank reveals in-depth knowledge about a user's information need buried in the search log data. In this experiment, we qualitatively illustrate some interesting findings from dpRank in our news search log data set.

By averaging the group assignment c for each training query q in the posterior samples, we can estimate the posterior distribution of queries within each user group as,

$$p(q|c = k) \propto \sum_{s \in S'} \sum_{u \in U} \delta(c_{uq}^s = k) \tag{13}$$

where S' is a set of effective posterior samples and c_{uq}^s is the sampling result of group assignment for query q from user u in s-th effective sample. Because in dpRank the number of user groups might change during the sampling iterations (i.e., generating/removing groups), we only collected samples after the sampling chain is stabilized in this estimation.

From one run of Gibbs sampling, 47 different user groups were created for our training set. Due to space limit, in Table 2, we only list the top 5 most probable queries from 10 selected user groups. We reassigned the IDs of these selected groups in the table for illustration purpose.

Table 2: Top 5 most probable queries under 10 selected latent user groups.

Group	Top Ranked Queries
1	iran, china, libya, vietnam, syria
2	selena gomez, lady gaga, britney spears, jennifer aniston, taylor swift
3	fake tupac story, pbs hackers, alaska earthquake, southwest pilot, arizona wildfires
4	joplin missing, apple icloud, sony hackers, google subpoena, ford transmission
5	casey anthony trial, casey anthony jurors, casey anthony, crude oil prices, air france flight 447
6	tree of life, game of thrones, sonic the hedgehog, world of warcraft, mtv awards 2011
7	the titanic, the bachelorette, cars 2, hangover 2, the voice
8	los angeles lakers, arsenal football, the dark knight rises, transformers 3, manchester united
9	miami heat, los angeles lakers, liverpool football club, arsenal football, nfl lockout
10	today in history, nascar 2011 schedule, today history, this day in history

As shown in Table 2, the query distributions in the automatically identified user groups are quite meaningful. The queries in groups 1 and 2 are mostly the names of countries and celebrities. Those queries account for the users' constant interest of obtaining the latest update about these countries and celebrities over time. The queries in groups 3 to 5 are mainly for breaking news events in the summer of 2011, e.g., "alaska earthquake" and "casey anthony trial." Queries in groups 6 and 7 are entertainment-focused, and those in groups 8 and 9 are sports related. We should note that we did not utilize any external knowledge, e.g., entity lexicon or taxonomy, about the queries in our query feature representation, but the proposed model automatically discovered such meaningful association among the queries by exploring the users' search behaviors.

One thing we want to emphasize is that the query distribution within each user group as illustrated in Table 2 is not solely determined by the query features, but also governed by the users' click preferences. Therefore, it is interesting to further investigate how different the users' ranking preferences are in those seemly distinct groups. In dpRank, the posterior distribution of linear weight β reflects a user's relative emphasis over the ranking features, and thus can be used to analyze users' ranking preferences in each group. In Figure 2, we visualized the corresponding posterior mean of βs from the same user groups as shown in Table 2. For comparison purpose, we also included the linear weights estimated by a globally trained RankSVM model and named it as Group 0. To make the scale of those linear weights comparable across different groups, we followed the approach used in [10] to normalize the weight vectors by l_2 norm. After normalization, we computed the variance in the linear weights of those features across different groups and selected the top 20 features with the largest variance for illustration. In particular, the first 4 features in the figure are *document age*, *query match in title*, *proximity in title*, and *site authority*.

Comparing to the globally estimated feature weights, users in different groups exhibit distinct ranking requirements. The users belonging to groups 1, 2, 8 and 9 highly emphasize the freshness quality of the returned documents, i.e., document age. And for those in groups 4, 5 and 6, they not

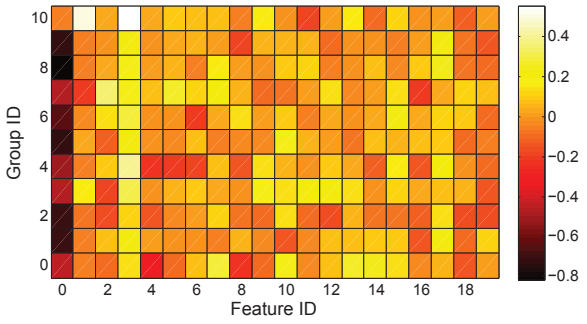

Figure 2: Heat map visualization of 20 selected feature weights in 11 different user groups (including a global RankSVM as group 0). Features are ordered by the variance of the estimated weights across different groups in descending order.

only prefer the latest news reports, but also emphasize the relevance quality of the documents, i.e., higher weight on query document matching. One interesting group of users identified by dpRank is group 10: they repeatedly issued the queries of "today in history." They did not mind the freshness of results, but stressed the query-document matching quality and site authority. In this sense, their search intent is more like navigational web search, rather than news search.

From the above results, it is evident that users' search intents are distinct, and dpRank captures such diversity and properly groups them by jointly modeling the users' issued queries and corresponding clicks in the search logs.

4.3 Quantitative Evaluation

In this section, we performed quantitative evaluation to test the main hypothesis in our work: the joint modeling of users' queries and clicks is better than those isolated modeling approaches. To test this, we compared our dpRank model with several state-of-the-art methods, which take separated steps for modeling the queries and clicks in the search logs, on the tasks of document ranking, collaborative query recommendation and document re-ranking.

Due to the randomness of sampling-based inference, the performance of dpRank varies across different runs of Gibbs sampler. In the following experiments, we report the average performance of dpRank over 5 different runs of sampling.

4.3.1 Baselines

As we discussed in Section 2, the two query-clustering-based ranking approaches [3, 10] analyzed the queries and clicks in an isolated manner. To test our hypothesis, we employed them as our major baselines for comparison.

Bian et al. proposed Topical RankSVM (TRSVM) for separating the training corpus into query-specific groups [3]. RankSVM models are trained on each of the identified groups. In TRSVM, k-means algorithm is used to cluster queries into groups based on the aggregated ranking features from pseudo-feedback documents for each query. In addition, to incorporate the knowledge of feature's importance in ranking, they weighted each dimension of query features by the weight vector from a global RankSVM model. To predict the ranking scores of documents in a new query, they assembled the independent RankSVM models' output by,

$$f(q,d) = \sum_{k=1}^{K} p(c=k|q) f_k(d) \tag{14}$$

where $f_k(d)$ is the output of RankSVM in the k-th group. The ensemble weight $p(c=k|q)$ is calculated by $p(c=k|q) \propto \exp(-||q-\bar{q}_k||^2)$, in which $\bar{q}_k$ is the mean feature vector of cluster k given by the k-means algorithm.

Giannopoulos et al. performed similar procedures to cluster queries and estimate RankSVM models for each query group [10]. But they represented queries by the linear weights from RankSVM models learned in each query, and assumed it reflected users' search intents. We will refer to this method as Intent RankSVM (IRSVM). In the original IRSVM, the ensemble weight $p(c=k|q)$ for a new testing query is estimated by the assignments of query groups in its similar training queries. But the authors did not clearly explain how to define such similarity between queries, and whether the model's performance is sensitive to such similarity metric. In our experiment, we estimated this ensemble weight by averaging the query group assignments in the same user's training queries,

$$p(c=k|u) \propto \sum_{q} \delta(c_{uq}=k) \tag{15}$$

We randomly sampled 20% training queries and performed 5-fold cross-validation to select the best performing cluster size K in TRSVM and IRSVM (according to MAP metric for document ranking). As a result, we fixed K in TRSVM and IRSVM to be 30 and 20 in the following experiments.

4.3.2 Document Ranking

In dpRank, a document's ranking score $\mathbf{s}$ for a given query in user group c can be calculated by the inner product between the linear weight vector β_c and document's ranking features, i.e., $\mathbf{s}(q_j^u, d_{jt}^u|c) = \beta_c^{\mathsf{T}} d_{jt}^u$. To rank documents for a testing query, we can compute the expectation of document d_{jt}^u's ranking score $\mathbf{s}$ for query q_j^u in user u, and order the documents by the descending order of such scores,

$$\mathbf{s}(d_{jt}^u, q_j^u) = \frac{1}{|S|} \sum_{s\in S} \sum_{k} p(c^{(s)}=k|q^u) \beta_k^{(s)\mathsf{T}} d_{jt}^u \tag{16}$$

where S is a set of posterior samples for the testing query q_j^u after burn-in period; the group index k takes value in $\{1,\ldots,K^{(s)},e\}$, and $K^{(s)}$ is the number of active groups in the s-th sample; and $p(c^{(s)}=k|q^u) \propto p(q^u|\mu_k^{(s)},\sigma_k^{(s)2}) p(c^{(s)}=k|\pi_u^{(s)})$ is the posterior group assignment for the new query q_j^u estimated by the s-th sample.

We also included two standard baselines for comparison: 1) a global RankSVM model is estimated based on all the training queries; 2) a set of independent RankSVM models are estimated for each user based on his/her own training queries, and applied to their testing queries accordingly. We named these two baselines as GRSVM and URSVM.

To quantitatively compare different models' ranking performance, we employed a set of standard IR evaluation metrics: by treating all the clicked documents as relevant, we calculated Mean Average Precision (MAP), Precision at 1 (P@1), Precision at 3 (P@3) and Mean Reciprocal Rank (MRR). Definitions of these metrics can be found in [2].

The ranking performance of different models is listed in Table 3, in which the proposed dpRank outperformed all the baseline methods. As we have demonstrated in Section 4.2, users' search intents are diverse, the global ranking model, i.e., GRSVM, failed to satisfy such divergent ranking requirements by using the one-size-fits-all ranking strategy. On the other hand, estimating separate ranking models for each individual user did not work as well. Because of the

Table 3: Comparison of ranking performance.

	MAP	P@1	P@3	MRR
URSVM	0.4865	0.2979	0.2195	0.5010
GRSVM	0.6161	0.4464	0.2826	0.6315
TRSVM	0.6223	0.4585	0.2825	0.6378
IRSVM	0.6165	0.4492	0.2814	0.6318
dpRank	**0.6424***	**0.4854***	**0.2897***	**0.6578***

$^*p$-value<0.05 under paired t-test against TRSVM

sparsity of training queries in each user (on average each user only has 4 training queries in this data set), URSVM cannot get accurate estimation of model parameters. For the same reason, IRSVM did not perform well either, since it depends on the ranking models estimated on each single query to perform the grouping. TRSVM and dpRank addressed the problems of diversity and sparsity by clustering the queries/users. But in TRSVM no user-related information is explored in query clustering, and thus the same queries from users with distinct ranking preferences will be ranked in the same way. dpRank solves this limitation by jointly exploring user-specific heterogeneity in their search behaviors (as shown in Eq (7)), which provides more accurate ranking for each individual user.

As having been demonstrated in Table 3, dpRank alone can already provide promising ranking performance. We want to further explore the merit of such joint modeling approach in helping general ranking model training. In this experiment, we chose LambdaMART [27] as our base ranking model. LambdaMART is one of the best performing learning-to-rank algorithms [4], which estimates a set of boosted regression trees to directly optimize IR-related metrics, e.g., MAP or MRR.

A set of additional ranking features are created based on dpRank's output for all the query-document pairs in the collection. In detail, beside the dpRank's output ranking score for a particular query-document pair (as defined in Eq (16)), we also treated the product of a document's predicted ranking score $\mathbf{s}(q_j^u, d_{jt}^u|c)$ in each user group and the corresponding posterior weight of this group inferred in the given query as a new ranking feature:

$$\tilde{f}_k(d_{jt}^u, q_j^u) \triangleq \frac{1}{|S'|} \sum_{s \in S'} p(c^{(s)} = k|q^u) \beta_k^{(s)\mathsf{T}} d_{jt}^u \qquad (17)$$

where $p(c^{(s)} = k|q^u)$ is the group assignment solely estimated on query q^u (the same as in Eq (16)) and S' is a set of effective posterior samples collected in the same way as in Eq (13) to avoid the change of number of user groups during Gibbs sampling. As a result, $K+1$ additional ranking features are introduced by dpRank. We did not include the auxiliary user group θ_e, because its prediction is random across different documents.

Similar procedures can be applied to TRSVM and IRSVM to extract corresponding ranking features. We computed the new query-group-based ranking features by,

$$\tilde{f}_k(d_{jt}^u, q_j^u) \triangleq p(c = k|q_j^u) f_k(d_{jt}) \qquad (18)$$

where $f_k(d_{jt})$ and $p(c = k|q_j^u)$ are RankSVM's output and the ensemble weight as defined in TRSVM and IRSVM.

These newly introduced ranking features are appended to the original 65 relevance-driven ranking features, and fed into LambdaMART together. We used the implementation in RankLib [6] with the default parameter settings (500 regression trees with 20 nodes in each tree), and MAP was chosen to be optimized during training. To avoid overfitting, we held out 25% training queries as validation set, and applied the best performing model from validation set to the testing queries. The ranking performance of LambdaMART with different features is shown in Table 4.

Table 4: Comparison of LambdaMART performance with ranking features derived by different methods.

Features	MAP	P@1	P@3	MRR
original	0.6643	0.5142	0.2975	0.6797
+TRSVM	0.6620	0.5108	0.2974	0.6774
+IRSVM	0.6361	0.4767	0.2872	0.6516
+dpRank	**0.6696***	**0.5212**	**0.2989***	**0.6833***

$^*p$-value<0.05 under paired t-test against original

As we can notice in the results, only the features generated by dpRank improved LambdaMART's ranking performance against the original features. To investigate the contribution of the new features introduced by dpRank, we computed the relative feature importance in the estimated LambdaMART model according to the method used in [14]. The top 10 important features are listed in Table 5. The most important feature selected by LambdaMART is dpRank's predicted document ranking score, which has already been proved to be highly correlated with users' click preferences by the results shown in Table 3. In addition, many features newly derived by dpRank were also considered as important, i.e., predicted document ranking score in different clusters, which correspond to coherent user groups, e.g., those who keep searching for celebrity or breaking news. As a result, the features from dpRank provide informative signals about each user's search intents and unique ranking preferences that enhance LambdaMART.

Table 5: Relative feature importance determined by LambdaMART.

Feature	Relative Importance
dpRank-score	1.000
c_{12}-score	0.464
c_0-score	0.250
query match in body	0.235
document age	0.228
perplexity in body	0.221
c_{13}-score	0.220
c_4-score	0.163
query match in title	0.141
site authority	0.135

4.3.3 Collaborative Filtering

The distribution of user group assignment π_u in each user naturally serves as a profile of user's search intent. It can be used as a proxy to measure similarity between the users. To study the quality of such user profile generated by our join modeling approach, we test it in the applications of collaborative query recommendation and document re-ranking.

The basic idea of collaborative filtering based method is to promote the items from users who share similar interest as the target user [20]. Therefore, the quality of employed similarity metric between users is crucial for this type of algorithms. In dpRank, we defined user similarity based on the posterior distribution of π_u as,

$$sim(u_i, u_j) = 1 - \frac{1}{2}\big[KL(\pi_{u_i}||\pi_{u_j}) + KL(\pi_{u_j}||\pi_{u_i})\big] \qquad (19)$$

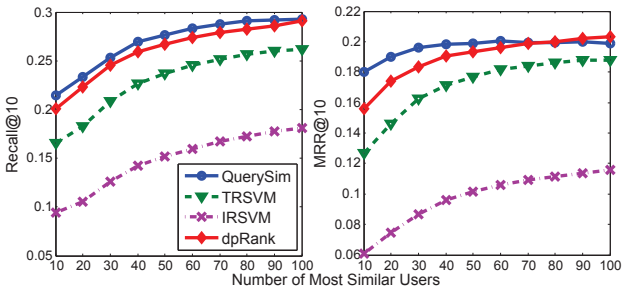

Figure 3: Comparison of collaborative query recommendation performance.

No user profiles can be directly extracted from TRSVM or IRSVM, because neither of them explicitly model the individual users. To make them applicable in this experiment, we used the same approach defined in Eq (15) to estimate the proportion of query groups in each user as his/her profile. In addition, we smoothed the estimation in Eq (15) by the global query group distributions in TRSVM and IRSVM to avoid zero probabilities. Based on this derived user profile, similarity between users is computed in the same way as defined in Eq (19) for TRSVM and IRSVM.

• **Collaborative Query Recommendation.** A good user profile for query recommendation should capture a user's specific search interest. To perform collaborative query recommendation, we calculated the weighted frequency of a candidate query in the M most similar users of the target user u, and selected the top 10 queries as recommendation. In detail, we only used the testing queries in each user for recommendation, since the training queries have already been used for creating the user's profile. Each candidate query q's recommendation score in user u is computed as,

$$r(q,u) = \sum_{u_i \in N(u,M)} sim(u,u_i)\delta(q,u_i) \qquad (20)$$

where $N(u,M)$ is the set of M most similar users to the target user u, and $\delta(q,u_i) = 1$ if the candidate query q occurs in u_i's testing query set.

In addition, we included a baseline method that only uses the vector of queries from a user's training query set as user profile. The similarity between users was measured by the cosine similarity between the query vectors of two users. We named this baseline as QuerySim.

To evaluate the recommendation performance, we treated the testing queries in the target user as relevant, and computed Recall (percentage of relevant queries occurred in the recommendation query set) and MRR as the performance metrics. We did not measure the precision of recommended queries, because in our setting it is uncertain whether the users did not like the recommended query or they just did not know about it when they did the search.

We varied the setting of M from 10 to 100 and summarized different methods' performance in Figure 3.

The user profile generated by dpRank outperformed those from TRSVM and IRSVM, because neither of these baselines explicitly model the distribution of queries in different users; but it was worse than QuerySim. We examined the recommendation results and found that a large portion of users tend to repeat their queries over time, e.g., celebrity queries. QuerySim captured such property by directly exploring a user's issued queries in history; while the granularity of profiles generated by dpRank is too coarse to capture such details of a user's search interest (user-group level v.s. query level). However, because QuerySim relies on strictly query matching for measuring user similarities, its coverage is limited: it cannot align the users with too few queries or those who shared the same search interest but issued different queries (e.g., vocabulary gap).

Table 6: Query recommendation performance with combined user profile (M=100).

	Recall@10	MRR@10
QuerySim	0.2929	0.1991
QuerySim+TRSVM	0.3133	0.2262
QuerySim+IRSVM	0.2763	0.2001
QuerySim+dpRank	**0.3280***	**0.2365***

$^*p$-value<0.05 under paired t-test against QuerySim

Since dpRank does not suffer from such a coverage issue as QuerySim, it would be beneficial to take advantage of both types of profiles for better query recommendation performance. In Table 6, we list the recommendation performance given by the combined user profiles with $M = 100$. In particular, we linearly interpolated the similarity scores given by dpRank and QuerySim, and gave QuerySim a higher weight (0.8 versus 0.2) due to the scale difference between cosine similarity and KL distance. We applied similar procedures on TRSVM and IRSVM for comparison purpose.

We can find that by combining the profiles from QuerySim all the model-based profiles, i.e., those generated by TRSVM, IRSVM and dpRank, achieved improved recommendation performance. And the combined profiles from TRSVM and dpRank further outperformed QuerySim profiles, and between them dpRank got better improvement.

• **Collaborative Document Re-ranking.** An effective user profile for collaborative document re-ranking should precisely reflect a user's result ranking preference. Similar procedure as we used in collaborative query recommendation can be applied here: given a list of documents ordered by a global ranking model, we promote the documents that have been clicked by the users who share similar profile as the target user. In particular, we define the collaborative ranking score for document d under query q in user u as,

$$r(d,q,u) = \sum_{u_i \in N(u,M)} sim(u,u_i)\delta(d,q,u_i) \qquad (21)$$

where $\delta(d,q,u_i) = 1$ if user u_i has clicked document d under query q.

The re-ranking procedure is performed as follows: given a list of documents ordered by a global model, we give extra credit to the documents that have been clicked by the target user's most similar peers ($u_i \in N(u,M)$) according to the score defined in Eq (21); for documents that have not been clicked by the target user's peers, we will keep their original ranking. For the same consideration as in the collaborative query recommendation, we only performed such re-ranking in each user's testing queries.

In this experiment, we also included a simple baseline method by re-ranking the documents according to its click-through rate (CTR) in all the users *except* the target user. We named this method as GlobalCTR.

Table 7 lists the re-ranking performance based on different user profiles, where the global ranking order is given by GRSVM and we computed the recommendation score in 100 most similar users, i.e., $M = 100$ in $N(u,M)$.

The profile generated by the user's queries in history (i.e., QuerySim) performed much worse than the simple GlobalCTR baseline. This result validates our hypothesis that

Table 7: Performance comparison of collaborative document re-ranking with different user profiles.

	MAP	P@1	P@3	MRR
GRSVM	0.6161	0.4464	0.2826	0.6315
GlobalCTR	0.6444	0.4832	0.2945	0.6601
QuerySim	0.6360	0.4738	0.2908	0.6521
TRSVM	0.6461	0.4861	0.2953	0.6619
IRSVM	0.6455	0.4855	0.2951	0.6614
dpRank	**0.6539***	**0.5016***	**0.2957**	**0.6711***

$^*p$-value<0.05 under paired t-test against GlobalCTR

only modeling a user's queries alone cannot well explain his/her search intent, and one also needs to capture the factors that affect the user's clicks on the retrieved documents. Comparing to TRSVM and IRSVM, which performed very similarly as the GlobalCTR baseline, profile generated by dpRank performed significantly better than GlobalCTR. Because we are evaluating in the news search log data, where new documents keep emerging. Only a small number of users who issued the same query will have overlap on the returned documents. As a result, the number of effective peers where we can collect click statistics for the target user is very limited; and thus the final performance is very sensitive to the quality of similarity metric between the users. In dpRank, the identified latent user groups explicitly encode different users' click preferences, so that when we weight the clicks from similar users by such profiles, we can get more accurate click predictions for the target user.

5. ACKNOWLEDGMENTS

Hongning Wang is supported by a Google Ph.D. Fellowship. This work is partly supported by the National Science Foundation under Grant Number CNS-1027965.

6. CONCLUSIONS

In this work, we studied the problem of joint modeling of users' queries and clicks in the search log data, and proposed a generative model, dpRank, to enable such a joint modeling. By introducing the concept of latent user groups and modeling each individual user's search intent as a mixture over such latent user groups, dpRank effectively identifies each individual user's search interests and their distinct ranking preferences in a unified way. Experimental results on a large-scale news search log data set validate our hypothesis that joint modeling of user's search behaviors provides a more comprehensive and accurate understanding of users' search intents, and the mined knowledge about the users is beneficial for a variety of personalized applications, e.g., document ranking and collaborative query recommendation.

The proposed modeling approach is general, and thus can be easily applied to many other types of search logs, e.g., web search and blog search. Besides, the current model only considers the users' logged search behaviors, but it would be interesting to incorporate more information about the users, e.g., gender, location, and social networks, to explore a more comprehensive understanding of users' search intents.

7. REFERENCES

[1] E. Agichtein, E. Brill, S. Dumais, and R. Ragno. Learning user interaction models for predicting web search result preferences. In *SIGIR'06*, pages 3–10. ACM, 2006.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[3] J. Bian, X. Li, F. Li, Z. Zheng, and H. Zha. Ranking specialization for web search: a divide-and-conquer approach by using topical ranksvm. In *WWW'2010*, pages 131–140. ACM, 2010.

[4] O. Chapelle and Y. Chang. Yahoo! learning to rank challenge overview. *Journal of Machine Learning Research-Proceedings Track*, 14:1–24, 2011.

[5] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *WWW'09*, pages 1–10. ACM, 2009.

[6] V. Dang. Ranklib-v2.1. `http://people.cs.umass.edu/~vdang/ranklib.html`.

[7] G. E. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *SIGIR'08*, pages 331–338. ACM, 2008.

[8] T. Ferguson. A bayesian analysis of some nonparametric problems. *The annals of statistics*, pages 209–230, 1973.

[9] R. Fidel and M. Crandall. Users' perception of the performance of a filtering system. In *SIGIR'97*, pages 198–205. ACM, 1997.

[10] G. Giannopoulos, U. Brefeld, T. Dalamagas, and T. Sellis. Learning to rank user intent. In *CIKM'2011*, pages 195–200. ACM, 2011.

[11] B. Jansen, A. Spink, and T. Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information processing & management*, 36(2):207–227, 2000.

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

[13] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR'05*, pages 154–161. ACM, 2005.

[14] A. C. König, M. Gamon, and Q. Wu. Click-through prediction for news queries. In *SIGIR'09*, pages 347–354. ACM, 2009.

[15] A. Kulkarni, J. Teevan, K. Svore, and S. Dumais. Understanding temporal query dynamics. In *WSDM'11*, pages 167–176. ACM, 2011.

[16] F. Liu, C. Yu, and W. Meng. Personalized web search by mapping user queries to categories. In *CIKM'02*, pages 558–565. ACM, 2002.

[17] R. Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000.

[18] R. Neal. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 54:113–162, 2010.

[19] D. Rose and D. Levinson. Understanding user goals in web search. In *WWW'04*, pages 13–19. ACM, 2004.

[20] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW'2001*, pages 285–295. ACM, 2001.

[21] J. Sethuraman. A constructive definition of dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

[22] F. Silvestri. Mining query logs: Turning search usage data into knowledge. *Foundations and Trends in Information Retrieval*, 4(1-2):1–174, 2010.

[23] D. Sontag, K. Collins-Thompson, P. N. Bennett, R. W. White, S. Dumais, and B. Billerbeck. Probabilistic models for personalizing web search. In *WSDM'12*, pages 433–442. ACM, 2012.

[24] J. Teevan, S. Dumais, and D. Liebling. To personalize or not to personalize: modeling queries with variation in user intent. In *SIGIR'08*, pages 163–170. ACM, 2008.

[25] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[26] R. W. White and S. M. Drucker. Investigating behavioral variability in web search. In *WWW'07*, pages 21–30. ACM, 2007.

[27] Q. Wu, C. Burges, K. Svore, and J. Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270, 2010.

[28] Y. Zhang, W. Chen, D. Wang, and Q. Yang. User-click modeling for understanding and predicting search-behavior. In *KDD'2011*, pages 1388–1396. ACM, 2011.