TECHNICAL REPORT

YL-2009-006

A COLLABORATIVE FILTERING APPROACH TO SPONSORED SEARCH

Tasos Anastasakos Dustin Hillard Sanjay Kshetramade Hema Raghavan

Yahoo! Labs
4402 Great America Parkway
Santa Clara, CA, 95054
{tasos,dhillard,sanjayk,raghavan}@yahoo-inc.com

14 August 2009

Yahoo! Labs Technical Report No. YL-2009-006

Yahoo! Labs Technical Report No. YL-2009-006

A COLLABORATIVE FILTERING APPROACH TO SPONSORED SEARCH

Tasos Anastasakos, Dustin Hillard, Sanjay Kshetramade, Hema Raghavan

Yahoo! Labs
4402 Great America Parkway
Santa Clara, CA, 95054
{tasos,dhillard,sanjayk,raghavan}@yahoo-inc.com

14 August 2009

ABSTRACT:

Search engine logs contain a large amount of click-through data that can be leveraged as soft indicators of relevance. In this paper we address the sponsored search retrieval problem which is to find and rank relevant ads to a search query. We propose a new technique to determine the relevance of an ad document for a search query using click-through data. The method builds on a collaborative filtering approach to discover new ads related to a query using a click graph. It is implemented on a graph with several million edges and scales to larger sizes easily. The proposed method is compared to three different baselines that are state-of-the-art for a commercial search engine. Evaluations on editorial data as well as online traffic data indicate that the model discovers many new ads not retrieved by the baseline methods. The ads from the new approach are on average of better quality than the baselines. In addition to the proposed approach our experimental methodology of evaluation on live traffic is a novel contribution to the academic literature.

1. Introduction

Major search engines typically see traffic which amounts to several million queries and a correspondingly large number of user-clicks on documents as a response to those queries. While clicks are noisy due to issues of click fraud, accidental or exploratory clicks and position-bias, they can often be interpreted as a weak indicator of relevance. Several recent works have used click information to improve performance on various tasks [20, 2, 13, 35]. We propose an algorithm that operates on a large bipartite graph of queries and documents (see Figure 1), where an edge between a query and a document is determined by whether users have clicked on the given document for the query. We apply an approach from collaborative filtering to first determine query-query similarity on this bipartite graph. The proposed method uses these query similarity scores to discover new documents that have not been shown for a given query before. Using an artificial example in Figure 1 to demonstrate the intuition behind the method, it is easy to see that the queries "cheap shoes" and "discount shoes" are highly related by virtue of the documents that are commonly clicked for them. The query "whole sale shoes" exhibits a high correlation to these other two queries and therefore one might extrapolate that the document $\mathcal{D}_N$ is also relevant to this query.

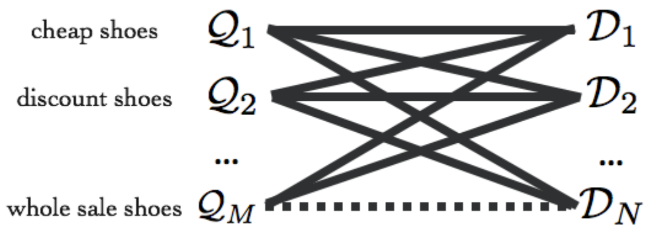

Figure 1: Bipartite graph of queries and documents. A solid line between a query and a document indicates that we have *observed CTR* of this pair. A dotted line indicates the *inferred CTR*. In this paper documents are search engine advertisements.

In this paper we illustrate the effectiveness of our proposed algorithm in the specific domain of sponsored search, where we aim to retrieve and rank textual ads that are relevant to a search query. Our work treats these textual ads as documents and in the remainder of the paper we use the terms ads and documents interchangeably. In Section 2, we introduce the domain and its terminology. Our collaborative filtering framework is presented in Section 3 and then we review related work in Section 4. In Section 5 we describe the experimental setup for offline and online evaluation of the system using the metric of normalized click-through rate. We conclude with results in Section 6 and a discussion in Section 7. We find that our proposed method is better than several baseline systems in terms of the quality of retrieved ads as measured offline by editorial assessments and online by the impact on user clicks. We find 2-18% relative improvements in click-through rates, and 5-37% relative improvements in editorial judgments compared to our baseline systems. Along with the algorithmic contributions, we believe our result analysis is also a novel contribution of the paper. We evaluate our results using not only human annotators/editors but also on live traffic data and measure the improvements in click-through-rate (CTR) brought about by our models. Finally our proposed techniques are general enough so that they can be applied to other domains with click feedback such as web search or image search.

The chief contributions of this paper are: (1) A new query-ad similarity approach based on the query-ad click-graph that can be implemented efficiently and effectively on a map-reduce framework such as Hadoop[1] to operate on several months or more of click-through data, (2) experimental evaluations on state-of-the-art baselines using offline editorial evaluations as well as online techniques that evaluate the proposed method on millions of events from live user-traffic. Our experimental evaluations show that our proposed retrieval method achieves better performance in both evaluation settings.

2. Overview of Sponsored Search

The primary source of revenue for major search engines is through advertising. The online ad spend of advertisers has been growing significantly over the past few years [1]. A search engine typically displays sponsored listings on the top and the right hand side of the web-search results, in response to a user query. The revenue model for these listings is "pay-per-click" where the advertiser pays the search engine only if the advertisement is clicked. The advertiser "targets" specific keyword markets by bidding on search queries. For example, an advertiser selling "shoes" may bid on user queries such as "cheap shoes", "running shoes" and so on. Sponsored search offers a more targeted and less expensive way of marketing for most advertisers as compared to media like TV and newspapers, and has therefore gained momentum in the recent few years.

We now describe the search engine monetization (SEM) terminology used in this paper. The terminology is similar across most of the major search engines. An advertising **campaign** consists of many **ad groups**; each ad group in turn consists of a set of **bidded phrases** or keywords that the advertiser bids on, e.g., *sports shoes*, *stilettos*, *canvas shoes* etc. A **creative** is associated with an ad group and is composed of a **title**, a **description** and a **display URL**. The title is typically 2-3 words in length and the description has about 10-15 words. Clicking on an ad leads the user to the **landing page** of the advertiser.

An advertiser can choose to use **standard** or **advanced** match for the keywords in an ad group. For example, enabling only standard match for the keyword "sports shoes", will result in the corresponding creative being shown only for that exact query. Whereas, if the keyword is enabled for advance match, the search engine can show the same ad for the related queries "running shoes" or "track shoes". A **bid** is associated with each keyword and a second price auction model determines how much the advertiser pays the search engine for the click [15].

Most search engines typically take a two pronged approach to the problem: (1) finding relevant ads for a query, and (2) estimating the click through rate for the retrieved ads and appropriately ranking those ads for display on the search page. In this work we are largely concerned with the former aspect, i.e., discovering new ads for a query. Nevertheless when we report results on the larger system we include a module that predicts CTR for the purpose of re-ranking. The final

[1]`http://hadoop.apache.org`

ranking displayed on the search engine is a product of the bid and the predicted CTR of the ad. Given a set of ads $\{a_1...a_n\}$ shown at ranks $1...n$ for a query q on a search results page, the expected revenue is given as:

$$R = \sum_{i}^{n} P(click|q, a_i) \times cost(q', a_i, i) \tag{2.1}$$

where $cost(q', a_i, i)$ is the cost of a click for the ad a_i at position i for the bidded phrase q'. In the case of standard match $q = q'$. Most search engines rank the ads by the product of the estimated CTR ($P(click|q, a_i)$) and bid in an attempt to maximize revenue for the search engine. Therefore, accurately estimating the CTR for a query-ad pair is a very important problem for search engines. One approach is to use the historical CTR statistics for query ad pairs that have been previously shown to users. However, the ad inventory is continuously changing with advertisers adding, replacing and editing ads. Likewise, many queries and ads have few or zero past occurrences in the logs. These factors make the CTR estimation of rare and new queries a challenging problem.

Finding relevant ads to a query is an information retrieval problem, and the nature of the queries makes the problem quite similar to web-search. Yet, there are some key differences between web search and sponsored search. One of the primary differences is that the collection of web documents is significantly larger than the advertiser database, and retrieving candidate ads for tail queries is a very important area of research for sponsored search. A second major difference is that the user model is different. Many queries do not have commercial intent; displaying ads on a query like “formula for mutual information” may hurt user experience and occupy real-estate on the search results page in a spot where a more relevant web-search result might exist. Therefore, in sponsored search, we prefer not to show any ads when the estimate of click-through-rate and/or relevance of the ad is low. Using the same user experience argument, for a navigational query [6] like “bestbuy.com”, we would rather show only that particular retailer’s website if that ad existed in the advertiser database. We refer the reader to the study of Jansen and Resnick [18] for further details on user perceptions of sponsored search. In web-search, determining how many candidates to retrieve and display is not as much of an issue as the generally accepted user model is one where users read the page in sequence and exit the search session when their information need is satisfied. While all of the above problems are interesting areas of research, we restrict ourselves to the problem of generating candidate ads for the query. However, we do evaluate our algorithm within a larger system that considers all the aforementioned aspects of sponsored search. The next section will introduce our approach and afterwords we discuss related work in sponsored search, collaborative filtering, and information retrieval.

3. Generating Ad Suggestions from the Query-Ad Graph

In this section we describe a recommendation algorithm based on a collaborative filtering framework. Recommender systems and collaborative filtering methods are increasingly common in e-commerce applications where they try to match users with products they may be interested in. A general recommender system matches users with items that are most likely to elicit an optimally positive response such as a purchase or a favorable product rating. Due to the growing volume and variety of products, the ability to offer informed recommendations is key in enhancing user satisfaction and loyalty.

Collaborative filtering (CF) methods use information from users' past transactions in order to predict future responses. Users are considered similar and are grouped together if they have similar preferences over a common set of items. This general concept can be extended to induce similarities between users even when they do not share a common set of items by measuring their behavior over similar items. Conversely, items can be considered similar when associated to a similar set of users. An appealing aspect of CF methods is that they do not rely on explicit user profiles or an underlying feature parametrization of the data associations. In the following, we apply the CF framework to the problem of query-ad recommendation by using queries and ads in place of users and items respectively. The input data consists of the sponsored search traffic logs from a web search engine that contain the user queries and the ads that were displayed and clicked over a specified time period.

It is useful to describe the problem in terms of a bipartite graph $\mathcal{G}(\mathcal{Q}, \mathcal{A}, \mathcal{E})$. The set of queries $\mathcal{Q}, (|Q| = M)$ and the set of ads $\mathcal{A}, (|A| = N)$ comprise the partitions of the graph and $\mathcal{E}$ is the set of edges that connect queries to ads. A query q and an ad a are connected if a user issued the query q and clicked on the URL corresponding to the ad a from the list of sponsored search results. The edge between q and a is weighted by $r_{q,a}$ with $r_{q,a} > 0$. The weight $r_{q,a}$ represents the strength of the association and can be measured as a function of the number of clicks generated for query q and ad a among all users, the CTR of the query ad pair, the number of conversions[2] for the query ad pair (q, a) or some other measure of affinity between query q and ad a. In this work an edge for (q, a) exists only if q and a appear in the sponsored search logs and at least one click has been observed for this pair. The resulting $N \times M$ response matrix $R = [r_{q,a}]$ is sparse, as the search logs reveal power-law distribution properties for the frequency of query ad clicks [13]. Furthermore as described in Section 3, we use two formulations for the corresponding edge weight, a position normalized CTR (nCTR) and a machine-learned estimate of the probability of click.

Unlike several previous applications of the click graph [13, 16, 24] we are not using click counts as the edge weights but functions of the click-through rate. The criteria of selecting ads in sponsored search combine relevance metrics of an ad to the input query with performance metrics and monetary incentives as described in Section 2. As a result, the click count of an ad over a time interval is not an accurate representation of the ad relevance, without taking into account the number of times that the ad was displayed to users. For illustration, in response to query q, ad a_1 is shown 1000 times and clicked 10 times whereas ad a_2 is shown 2000 times and clicked 15 times. By click count a_2 appears more popular and relevant, whereas the CTR metric (ratio of clicks to impressions) indicates that a_1 is more relevant to query q. Using the CTR metrics we can normalize for the selection bias in the data. We use two different formulations for the edge weights that are described in sections 3.2.1 and 3.2.2. The latter can take into account content similarity between queries and ads, so that a similarity measure can be computed even in cases when there is no available click information to associate a query and an ad.

The goal of the CF algorithm is to predict the unknown responses in the matrix R (the dotted line in Figure 1). Based on a common family of CF approaches, termed "neighborhood methods" [5,

[2]number of clicks that ultimately resulted in a completed transaction, such as a purchase

23, 34], our method first computes similarities between queries and then computes a prediction of the response between a query and a new ad based on how similar queries responded to the same ad. Breese et al [5] find that the correlation based formulation performs very well for a wide variety of tasks and our method is based on that. The approach consists of two operations: finding similar queries and then predicting the response of unseen query-ad pairs.

3.1. Finding similar queries

We estimate the similarity s_{ij} between two queries i and j with the Pearson correlation. If we represent queries by the corresponding rows in the response matrix R, and set non-observed query-ad pair responses to zero, the correlation similarity is computed as:

$$s_{i,j} = s(\boldsymbol{q}_i, \boldsymbol{q}_j) = \frac{\sum\limits_{k \in \mathcal{S}(i,j)} (r_{i,k} - \overline{r}_i)(r_{j,k} - \overline{r}_j)}{\sqrt{\sum\limits_{k \in \mathcal{S}(i,j)} (r_{i,k} - \overline{r}_i)^2} \sqrt{\sum\limits_{k \in \mathcal{S}(i,j)} (r_{j,k} - \overline{r}_j)^2}} \tag{3.1}$$

where $\overline{r}_i$ ($\overline{r}_j$) is the mean response of query q_i (q_j) over all advertisements. The correlation measure is normalized for global mean and variance effects.

The summations are computed over the set of common ads between the two queries, the support set $\mathcal{S}(i, j)$. Similarity measures computed over larger support sets are likely to be more robust, so we adjust the similarity metric by an overlap factor to discount similarity scores for query pairs with a small support set. The overlap factor is defined as:

$$J(\boldsymbol{q}_i, \boldsymbol{q}_j) = \frac{\sum\limits_{k \in \mathcal{S}(i,j)} (r_{i,k} + r_{j,k})}{\sum\limits_{l} r_{i,l} + \sum\limits_{l} r_{j,l}}$$

The similarity score between two queries q_i and q_j becomes:

$$\widehat{s}(\boldsymbol{q}_i, \boldsymbol{q}_j) = J(\boldsymbol{q}_i, \boldsymbol{q}_j) \cdot s(\boldsymbol{q}_i, \boldsymbol{q}_j) \tag{3.2}$$

Each query is characterized by a vector $\boldsymbol{q}_i = [r_{i,1}, \ldots, r_{i,N}]$ where $r_{i,j}$ is the edge weight between query q_i and ad a_j. The semantics of the query q_i is captured in the ads that are associated with the query with varying measures of intensity represented by $r_{i,k}$ for each ad a_k. Similarly the queries that are associated with an ad in the bipartite graph provide a description of the ad. The similarity score sums over all ads in the support set. A potential disadvantage in this formulation is that ads that are linked to a very large number of queries have diffused semantics and are not a good indicator of similarity. We define the inverse frequency for advertisement a_k as:

$$f_k = \log \frac{M}{(\text{\# of queries linked to } a_k)}$$

and weigh the responses in $\boldsymbol{q}_i$ as:

$$\widehat{\boldsymbol{q}}_i = [(f_1 r_{i,1}), \cdots, (f_N r_{i,N})] \tag{3.3}$$

The similarity score as defined in equation 3.1 can now use the new edge weights $f_k r_{i,k}$ instead of $r_{i,k}$ in order to discount the effect of advertisers that associate the same ad with a large number of bidded phrases. These advertisers typically aim to display their ads to a large number of users to make their brand known and do not mind a low CTR, whereas our algorithm goal is to find relevant ads and queries. This approach is commonly used in information retrieval where word frequencies are modified by their inverse document frequency in order to offset the importance of words that occur in multiple documents. The hypothesis is that words that appear in multiple documents are not very discriminative of documents and topics and therefore their feature weight should be discounted

3.2. Predicting query-ad responses

Using the similarity measures derived in equations (3.1–3.3), we compute the set $Q^K(t)$ of K nearest neighbor queries to q_t. An ad a_i that is not adjacent to query q_t in the bipartite graph $\mathcal{G}$ has an unobserved response. If a_i is adjacent to any of the queries in $Q^K(t)$, we derive the predicted response $r_{t,i}$ between query q_t and ad a_i as the weighted average of the responses of these neighboring queries:

$$r_{t,i} = \frac{\displaystyle\sum_{k \in Q^K(t)} \widehat{s}_{t,k}\, r_{k,i}}{\displaystyle\sum_{k \in Q^K(t)} \widehat{s}_{t,k}} \tag{3.4}$$

The observed responses $r_{k,i}$ are weighted by the query similarity score for each query rewrite q_t to q_k. If the response matrix R holds the observed CTR, the computed quantity is the predicted CTR for a new query-ad pair. For the remainder of the paper we refer to the response prediction following equation 3.4 as the **QuAd-Click-Graph (query-ad-click-graph)** approach.

3.2.1. Estimating $r_{i,k}$ using normalized CTR The edges on the query ad graph can be weighted by the click-through-rate - a metric that measures the *fraction* of times an advertisement was clicked. A simple way to calculate CTR for an advertisement a for query q is to divide the number of $clicks$ by the number of $impressions$ (or views) of the query and ad pair : $CTR(q,a) =$ clicks$(q,a)/$imp(q,a). The main issue with the raw CTR metric defined in this way is the problem of *position-bias* in clicks [30, 20]. It is well known that users tend to click on items at higher positions in a ranked list regardless of the relevance of the item. This is in part due to the top-down order in which users navigate the ranked list and also in part due to the inherent bias in their belief of a search engine's ability. Therefore, methods that use clicks for implicit feedback or for evaluation need to factor out this position bias. In search engine advertising there is the additional problem due to the "golden triangle" of users' attention on a search results page [30]. Most of the visual attention is focused just below the search-box, with attention on the right hand side of the page being very low, leading to very low CTR for those ads. We use a metric (nCTR) that computes the number of clicks divided by the expected clicks of an item as a rank normalized version of CTR [36, 9].

Let clicks(q, a, d) be the number of clicks at rank d for an ad a that has been retrieved for a query q. $nCTR(q, a)$ is defined as

$$\begin{aligned} nCTR(q,a) &= \frac{\sum_d \text{clicks}(q,a,d)}{\text{ec}(q,a)} && (3.5) \\ \text{ec(q,a)} &= \sum_d \text{imp}(q,a,d) P(\text{click}|d) && (3.6) \end{aligned}$$

The quantity ec(q, a) is the expected number of clicks computed over all possible rank positions. The quantity $P(\text{click}|d)$ is estimated by observing the per-position click-through rate on a size-able portion of the traffic for several days.

3.2.2. Estimating $r_{i,k}$ using $P(click|query, ad)$

The smoothed response $r_{k,i}$ can be estimated using a machine learned model that predicts the probability that the user is likely to click on an ad for a query $P(click|q, ad)$. We find that learning a maximum entropy model for this task is quite effective. The model has the following functional form:

$$p(click|query, ad) = \frac{1}{1 + exp(\sum_{i=1}^{N} w_i f_i)} \tag{3.7}$$

where f_i denotes a feature based on either the query, the ad or both. w_i is the weight associated with the feature and N is the total number of features. The model is described in the work of Shaparenko et al[32] and is learned from the query logs of a major search engine. Each line in the query log contains a query and an ad along with other information such as the time-stamp and the position on the page that the ad was shown to a particular user. The search engine also records whether the user clicked on the particular query ad pair (1 for a click and 0 for no click). This data is used to train a binary classifier using the maximum entropy model as described above. While any supervised classification algorithm may have been used (eg., [30] and [11]) the learning framework for maximum entropy can be easily parallelized to handle the millions of training samples that can be obtained from the query logs. Maximum entropy models can also handle sparse and mutually correlated feature-sets reasonably well. We have also experimented with Gradient Boosted Decision Trees and trained on the same set of features but used nCTR as a target. The model we tried was similar to that of Richardson et al [30]. In various experiments we have found both GBDT and Maximum Entropy models to be equally effective for click prediction.

The features for this model include textual overlap features, historical click-through-rate features and other features such as time-stamp, time-of-day, page position etc. The textual overlap features measure the proportion of words and characters common between various parts of the ad: title, description, display URL. Historical click-through-rate is aggregated at various levels: query-ad, ad, query. An ad is also associated with a campaign and an account so we can therefore aggregate the click-through-rates for campaigns and accounts and additionally use them as features in the model. Several conjunctions of these features are also used. The model is trained on 3 months of query logs.

3.2.3. Implementation Details The click-graph in our implementation is constructed from a portion of sponsored search traffic for a commercial search engine. The graph data is collected over a period of two weeks and consists of user queries and the associated ads that are displayed and clicked. A typical graph consists of 27 million unique queries, 20 million unique ads and 51 million query-ad edges. Each edge of this bipartite graph represents a click response between the associated query and ad, weighted by the number of clicks or a predicted smoothed response that incorporates lexical features as described in Section 3.2. The graph is regenerated every week in order to include recent queries and capture recent changes in user activity. The algorithms can easily be implemented in a map-reduce framework and new query-query and query-ad associations can be detected in a few hours. Empirically we found that using nCTR for the query-query similarity scores and $p(click|query, ad)$ for the query-ad similarity scores was the best weighting scheme.

4. Related Work

As discussed previously we divide related work into two parts based on the two major sub-components of a typical SEM system.

4.1. Finding relevant ads

Past work on finding relevant ads for a query has typically used one of two different approaches: (a) query rewriting methods [21, 26] or (b) direct query-ad matching approaches [7, 28]. In query rewriting, the goal is to generate a relevant rewrite q_j for a given query q_i. Then ads associated with the bidded phrase q_j are retrieved in response to input query q_i. In direct query-ad matching the ads are treated as documents with multiple sections that correspond to the creative and landing page. Following standard information retrieval and web-search techniques, an input query is used to retrieve and rank candidate ads.

Typical query re-writing approaches learn from user query transformations extracted from web-search logs [21, 37]. These transformations include similar queries and sub-phrases in query re-formulations that are obtained from user sessions in the logs. Different models replace the original query, or sub-phrases and compute the similarity of the new query to the original query taking into account lexical constraints and the frequency of the proposed transform in the logs. These methods have shown to work well in practice. We use the session-based query rewrite approach [21] as a baseline and show that our proposed method performs well in comparison.

While there have been few papers on traditional information retrieval approaches for sponsored search [7, 28], much work has been done in web-search to determine relevance of a web-page to a query [12]. Many of the successful models typically combine query-document overlap features with document attributes such as host-trust using machine learning techniques [8, 10]. We further describe one such system that we use as our baseline in Section 5. There is also a size-able body of work that uses IR methods for contextual advertising (the problem of finding ads relevant to a web-page) [33, 29, 25]. However, in contextual advertising, the web-page is a rich source of features, while in sponsored search a query is quite sparse, often only 2-3 words in length, posing a greater challenge for retrieval. In addition, there is extensive research that uses click data to improve the effectiveness of document retrieval in web-search in the recent years [2, 14, 27]. Often click-features used in addition to lexical overlap features like BM25 in a machine learning ranking model improve performance over models that only consider lexical overlap.

Recent work on query recommendation and query clustering has considered the input data in web search as a bipartite click graph of queries and documents with edges that correspond to click information [4, 35, 13, 16, 3]. This abstraction naturally leads to random-walk methods that vary in the definition of the transition matrix and the stationary probability distribution that they compute over the graph nodes. In [13, 16] the edge weights in the graph represent click counts between queries and documents and the random walk is defined so that for an input query it computes the personalized PageRank [19] of all other queries and then selects the top ranking ones as recommendations. The self-transition probabilities and the absolute limit threshold on the random walk length are used to keep the results close to the original query and avoid concept drift. A similar approach is followed in [16] where the random walk is formulated in terms of a Markov random field model. Craswell and Szummer [13] find that a large amount of recall is obtained at a walk length of 3, after which precision deteriorates slightly, with little increase in recall. The approaches in [4, 35] use the click graph to obtain query recommendations where the similarity measure is based on the overlap of adjacent nodes between two queries without taking into account the volume of clicks. In [24] the random walk aims to estimate the probability of hitting time for each query node. The hitting time algorithm computes how soon the original query can be reached from any other query in the graph with an average of all possible paths. The query suggestions are obtained as the queries with the top ranking hitting time probabilities. This method aims to mitigate the concept drift discussed earlier without tuning parameters.

Our method viewed as a random walk on a bipartite graph enables walks of 3-step transitions for query suggestions and 4-step transitions for query-ad response prediction. As found in prior work, limits on the maximum transition length lead to higher precision. Unlike most methods we utilize a function of CTR as the weight on the graph edges that avoids following popular paths obtained due to selection bias. The transition matrix in our formulation is not normalized to unity, and is not considered a probability measure. The work in [3] extends the random walk to n-step transitions using the CTR graph.

Finally, our work primarily follows prior research on collaborative filtering [5, 23, 34] that have been extensively applied to user-item data sets where the edge weights are either binary-valued or reveal ratings and preferences. The goals are to recommend new items to users preferences, cluster similar users or similar items based on associations derived from historical preferences. The use of Pearson correlation and the extensions described in Section 3.1 has been detailed in [5] and more recently used in recommendation tasks for large data sets [23, 34].

4.2. Predicting Click Through Rate

As discussed previously, the goal of the ranking model is to predict the CTR of a given query-ad pair. Ciaramita et al [11] use a logistic regression model to predict user clicks. Richardson et al [30] use a regression model trained on CTR as a target. These models typically use a rich set of features that include historical CTR and lexical overlap features. Dupret and Piwowarski [14] recently proposed such a model for web-search. While these techniques typically do not find new ads, our proposed algorithm has the advantage of discovering new query-ad pairs and estimating their click-through-rate at the same time.

5. Experimental Setup

In this section we describe our baselines and offline and online evaluation strategies.

5.1. Baselines

We compare the new **QuAd-Click-Graph** ad suggestions described in the previous section (3.2) to three baselines. The first baseline uses an information retrieval approach that matches queries directly to the ad and the remaining two baselines are based on query rewriting techniques.

Our first baseline is the direct query-ad system based on a commercial search index of the ads. Given a query, a search is performed on the index of landing pages and ad descriptions. The system uses a machine learned ranking function trained on query-document pairs which are labeled on a five point scale similar to many recent works [17, 10]. The model used is very similar to the Gradient Boosted Tree method which is the basis of the approach of Chen et al for web-search [10]. The model uses several features that broadly fall into three classes: (a) query features, such as query length (b) document level features, such as host-trust, anchor-text, category information and (c) features that model the query document relationship, such as word-overlap in various sections of the landing page. For each query 100 ads are returned, and these are further filtered by comparing with the web-search results. Ads with relevance significantly worse than the quality of the web-search results are not shown, since bad ads can significantly hurt user experience [18]. We refer to this as the **IRB (information retrieval baseline)** in the rest of the paper.

The first query rewriting baseline is the query log based system [21] introduced in Section 4. We refer to this system as **LBQSB (log based query substitution baseline)**. Taking an example from the original paper, given a query *"catholic baby names"*, the method considers rewrites of the original query such as *"baby names", "catholic names"*. In addition rewrites are also generated by segmenting the original query as *(catholic) (baby names)* and considering insertions, deletions and substitutions of the individual segments in the query log giving alternative rewrites like *"(christian) (baby names)", "(baby names)"* etc. A log-likelihood ratio threshold is used to filter out query pairs that co-occur frequently by chance (due to the high query frequency of the individual queries).

The second query rewriting baseline is the collaborative filtering approach described in Section 3.1. We refer to this set of nearest neighbor queries as the **CFB (collaborative filtering baseline)**. We take up to five query rewrites per query, ranked by query similarity score.

The above three baseline methods are a subset of the techniques that contribute to edges on the click-graph used for our proposed methods. The Query-Ad Click Graph system builds on top of this graph and therefore, the method discovers not only new documents/ads not retrieved by the CFB method, but also new documents that were not discovered by any of the other baselines.

Score	Point Value
Perfect	3.0
Certainly Attractive	1.0
Attractive	0.5
Somewhat Attractive	0.4
Probably Not Attractive	0.2
Certainly Not Attractive	0.0
Offensive/Risky	0.0

Table 1: Relevance Judgment Labels for Offline Evaluation

5.2. Offline Testing

We provide an offline evaluation of the **QuAd-Click-Graph** ad suggestions for a sample set of 1,000 randomly selected queries that are representative of typical search engine traffic. We generate the **QuAd-Click-Graph** ad suggestions as well as ads from each of the three baseline systems for this query set. The pooled set of query-ad pairs retrieved by all systems were labeled by trained editors who gave each ad a rating from one of the seven categories listed in Table 1. The score *Perfect* is reserved for ads that are a perfect match to a query where there is only one correct answer. *Certainly Attractive* results meet or are strongly related to the likely commercial intent or explicit need of the query. *Probably attractive* and *Somewhat attractive* labels were assigned to ads that had a shift in scope/specificity from the original intent of the query. Judges marked listing as *Probably Not Attractive* when they can see the relationship between the listing and the query, but the listing is not likely to meet a commercial need of the user. Ads for which a user was never likely to click on were marked *Certainly Not Attractive*. The editors were experienced professionals, trained in the task, which had strict and very detailed guidelines with several examples for each label. Each judgment is associated with a point value which enables us to compute a weighted **average-score** of the listings returned by each of the systems.

5.3. Online Testing

After offline experiments validate the effectiveness of our technique we ran our system live on a fraction (or “bucket”) of traffic for a real commercial search engine. Given the large user base of the commercial search engine that we had access to, it was possible to expose our algorithm to several million users and reliably collect statistics over a sufficiently long period of time. Using a large number of page-views containing candidates generated from our system and the baselines, we can reliably estimate measures that indicate the quality of the individual systems. Additionally, because in “pay-per-click” advertising the desired goal of the search engine is user-clicks on ads, metrics from such “bucket-testing” can help evaluate the monetization capability of the new algorithm. For an introduction to bucket testing the reader is referred to the paper by Kohavi et al [22].

The online traffic is divided so that a random subset of the user population (or a "bucket") receives candidates retrieved by the new algorithm. We want to test whether the proposed algorithm adds value to a system that is as good as the current production system, so we run a version of the production system in a "baseline bucket" that contains among many other candidate generation techniques, the three baseline systems (LBQSB, CFB and IRB). In the "experimental bucket" candidates are generated using all the techniques in the baseline, in addition to the proposed query-ad click-graph technique.

An important point to note is that the CFB was generated on the entire query-ad click graph obtained from the system in the baseline bucket following the description in Section 5.1. We restricted ad suggestions from our proposed method to only include those ads that were "new" or had not been retrieved by any of the systems in the baseline bucket.

In both buckets all candidates retrieved are de-duplicated and re-ranked using a model trained to predict the probability that the ad is likely to be clicked for the query. This model is the same as the one described in Section 3.2.2.

Given such a system, we can then measure the effectiveness of our method by computing click-based metrics of each of the individual candidate generation technologies as well as aggregate statistics comparing the two buckets. The web-logs allow us to analyze the data to determine which candidate generation technology was the source for each candidate retrieved. The chief metric we use for comparing algorithms in an online setting is the nCTR described in section 3.2.1

One of the caveats of the metrics reported on online testing is our ability to report only relative performances with respect to the baseline for most metrics due to confidentiality reasons. In addition, the online click metrics discussed above are also only reliable at the query level for more popular queries (with sufficient impressions), so editorial judgments are necessary for infrequent queries that appear only once or twice a month. A search engine sees a huge volume of a diverse set of queries every day, hence it is not manually possible to have humans editorially judge a large enough set to compare to the online click evaluations. A combination of online and offline metrics are necessary to understand the quality and impact of any algorithm.

6. Results

In this section we discuss our results in both the offline and online testing frameworks described above.

6.1. Offline Experiments

We first describe our experiments on determining the best response ($r_{k,i}$) to use in Equations 3.1 and 3.4 and then proceed to compare our algorithms with the various baselines.

6.1.1. Query-ad click graph weighting experiments We now describe experiments performed to describe the best method of weighting the edges on the query-ad click graph ($r_{k,i}$) for predicting query-query similarity as well as for predicting query-ad similarity.

For the collaborative filtering algorithm we considered the following four configurations:

	t1	t2	t3	t4
P	0.639 (0.10)	0.640 (0.10)	0.660 (0.10)	0.653 (0.10)
R	0.435 (0.06)	0.435 (0.07)	0.416 (0.05)	0.445 (0.07)

Table 2: Micro-averaged (over-query) values of precision and recall. Numbers in bracket indicate variance. None of the methods are statistically significant in their difference from each other.

t1: uses $r_{k,i} = nCTR(query, ad)$

t2: extends t1 with the addition of inverse advertiser frequency (InvAdvFreq) per equation 3.3

t3: uses the t1 configuration and additionally discards graph edges with expected clicks (equation 3.5) below a threshold set to 0.1. The rationale is that these associations have very few impressions and/or have very low rank. They might be the result of infrequent queries, or lower quality advertisement matches and introduce noise in the graph

t4: weights the graph by a model based estimate of the clickability of an ad for a query as described in section 3.2.2

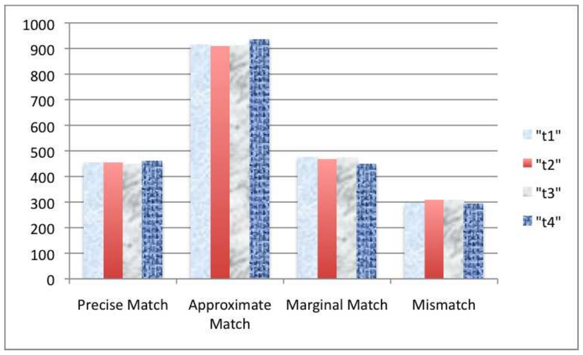

Figure 2: The fraction of judgments in each relevance category retrieved by the different weighting schemes on the edges.

We asked human judges (editors) to label query rewrite pairs generated by the above four methods. 490 queries were run through the system with the above four configurations and each method generated up to five suggestions of "similar queries" along with a score. In total 4684 unique query pairs were given to the judges. The judges were asked to label the suggestions (q') for a query (q) on a 4 point scale:

1. Precise Match: User intent is almost entirely preserved with little allowance for variations in scope. Eg.,: *automotive insurance* and *automobile insurance*

2. Approximate Match: The suggestion (q') has direct relationship to the topic of the query but the scope has narrowed or broadened slightly. Eg., *h*ybrid car and *toyota prius*

3. Marginal Match: A distant but plausible match, say via some categorical relationship to a related topic. Eg., *ibm thinkpad* to *laptop bag*.

4. Clear mismatch: There is no relationship between the two queries.

In terms of the fraction of candidates in each category marked by the editors, all four weighting schemes perform similarly, with t4 being slightly better as is indicated by Figure 2. We also considered precision and recall by marking the first two ratings (Precise and Approximate) above as relevant and the remaining two as non-relevant. The micro-averaged precision and recall is then computed and reported in Table 2. None of the methods are statistically significantly different from each other.

However, in absolute terms, we can observe that the use of inverse advertiser frequency (t2) improves precision a little. Edge filtering (t3), as expected, improves precision at the expense of recall. t4 slightly improves precision and recall over t1 and t2. We can also plot at precision and recall graphs by sweeping through the score thresholds as shown in Figure 3. The use of inverse advertiser frequency (t2) helps improve t1 in the high precision regions. t4 improves over t1 and t2 in the high recall regions. While none of the weighting approaches show significant differences, we chose t2 for logistical and implementation reasons.

Using the t2 configuration for the Collaborative Filtering Algorithm as described previously, we proceed to evaluate the best weight to use for $r_{k,i}$ in equation 3.4. We used the human-labeled data obtained as discussed in section 5.2. We experimented with both normalized CTR and a model based prediction as we did in the first stage. Here we found that using the machine learned model (from Section 3.2.2) was significantly better (p-value = 4.932e-05) than using nCTR (from Section 3.2.1) and therefore we use this weighting approach for our experiments. The average score using each of these two methods was 0.365 for the nCTR weighting and 0.41 for the model based weighting.

Now that we have determined how to weight the edges on the query-ad click-graph we can compare the method with other state-of-the-art baselines. For each query q_t we select up to 30 ads a_i that are ranked by their predicted response $r_{t,i}$. The new query-ad suggestions are then added to the final candidate set for display on the search page.

6.1.2. Comparison with Baselines Table 3 presents results of the editorial evaluation for query-ad pairs generated by our three baseline suggestion systems, as well as the proposed QuAd-click-graph. For each query, editors score all of the proposed suggestions and the scores are averaged to produce an overall quality score per method. Column 2 in table 3 gives the percent of retrieved ads that are relevant (i.e., had a score greater than "somewhat attractive") and column 3 presents the total number of listings retrieved by each method.

The QuAd-click-graph query-ad pairs achieve the highest average editorial score, with a similar quality to the IRB system (although the IRB system produces much fewer overall ad suggestions). The CFB and LBQSB systems have more suggestions, but significantly lower average quality.

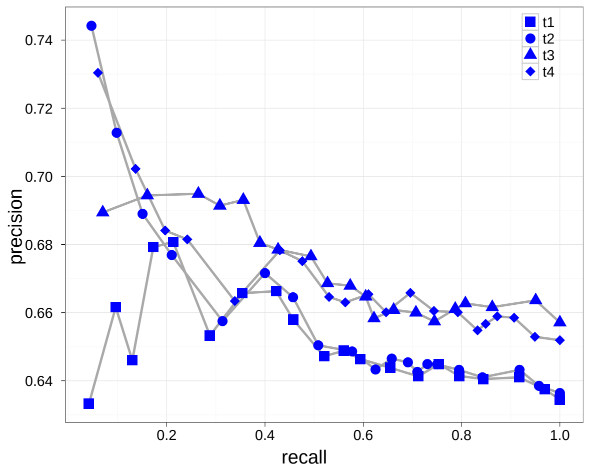

Figure 3: Precision Recall Curve for different weighting schemes

Method	% Relevant Listings	#listings	Average Score
IRB	68	311	0.39
CFB	54	908	**0.30**
LBQSB	45	683	**0.26**
QuAd-click-graph	67	1753	0.41

Table 3: Average relevance scores for offline editorial testing. Bolded numbers indicate the baselines which the QuAd-click-graph method outperformed in statistical significance tests.

	$r_{t,i}$ threshold				
	0	.5	1	1.5	2
Total Listings	1,753	1,635	1,180	456	113
Total Queries	366	348	287	178	63
Average Score	0.41	0.41	0.45	0.51	0.62

Table 4: Average relevance scores for offline editorial testing of QuAd-Click-Graph with various thresholds on predicted response ($r_{t,i}$).

The distribution of editorial scores per system is graphed in Figure 4. The bar graph displays the percent of suggestions that fall into each editorial category. Again, QuAd-click-graph and IRB have similar trends, with about 10% *Certainly Attractive* and 30% *Probably Attractive* ads. CFB and LBQSB distributions are heavier at lower quality, with 20% to 30% receiving judgments of *Certainly Not Attractive* compared to about 10% for IRD and QuAd-click-graph.

While the editors reviewed all ads with non-zero predicted response scores, we could filter what ads we present in live online tests. We investigate possible filtering thresholds on the predicted response scores ($r_{t,i}$) by computing the average editorial score for all ads that pass a given threshold. Table 4 presents the average editorial score at various response score thresholds. In addition, we present the number queries with ads (and total number of ads) at each threshold to show the trade-off between number of ads and average quality. We select a threshold of 1.5 in order to achieve an average editorial score of .51, while still maintaining coverage[3] of 178 out of 1,000 queries.

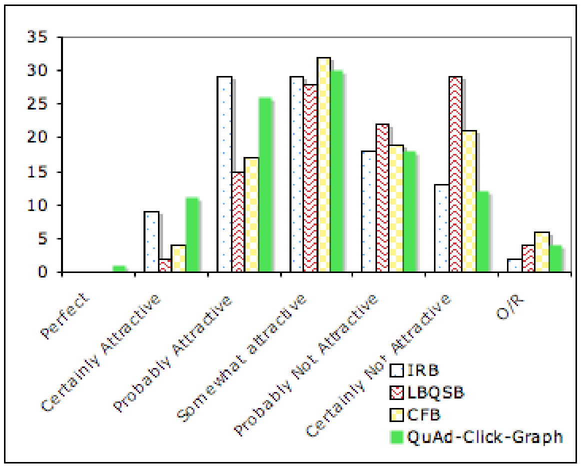

Figure 4: The fraction of judgments in each relevance category retrieved by each method.

[3]coverage is the portion of queries for which ads from a given algorithm are available

6.2. Online Experiments

We ran an online experiment with the "experimental bucket" for almost 3 weeks so that there were sufficient impressions of the new candidate advertisements in order to reliably estimate nCTR. To show why this is necessary, we investigate a set of 20K queries that were randomly sampled from the query logs in the experimental bucket. On the first few days of the bucket-test, the fraction of query-ad pairs retrieved by our method which had collected non-zero click rate history was only 10%. After three weeks, about 55% of query-ad pairs had collected click rate history (which also leads to improved $P(click|query, ad)$ accuracy due to more historical statistics).

We computed the relative performance of the proposed algorithm with respect to the baselines on the entire traffic in the experimental bucket in the fourth week. The relative nCTR is shown in the second column in Table 5. The results show that the proposed method is better than all baselines in terms of improving quality on the online metric of nCTR. The differences were also found to be significant at greater than 95% confidence level.

Baseline	$\triangle\ nCTR$
$\triangle$(QuAd-IRB)	18.36%
$\triangle$(QuAd-LBQSB)	3.7 %
$\triangle$(QuAd-CFB)	2.1%

Table 5: Improvement in Click-through-rates with respect to several baselines observed on a portion of live traffic.

As a whole, the experimental bucket showed a nCTR that was 3.6% better than the baseline. The performance improvement on the aggregate nCTR of the baseline is smaller than the numbers seen in Table 5, because the coverage, is small as there are many diverse algorithms running in parallel and due to the several strict quality filters that are applied on the front-end. Additionally, the coverage of query rewriting techniques like LBQSB and CFB is always bound to be higher because of their more generic nature (query level vs. ad level). Hence, even though the quality of the QuAd click-graph ads are better, they are a small fraction of the total, leading to a smaller improvement in the total nCTR of the experimental bucket.

One must observe here that the CFB and QuAd click-graph algorithms rely on queries having appeared in the logs. This is a common characteristic of collaborative filtering recommender systems termed "cold start" [31]. The IRB on the other hand does not have any click-feedback based features. Hence it is worth examining the sensitivity of our methods and the baselines to query frequency. We sorted the queries that occurred in the fourth week of our bucket by the number of page-views and split them into 10 bins or deciles. The cumulative frequency of all the queries in the bins is equal, with decile 1 having the most frequent queries like "ebay". Decile 1 has less than 1% of the unique queries while decile 10 has about 30% of the unique queries in this analysis. This is due to the power law distribution of query-frequency in web-search.

Figure 5 shows the proportion of ads per decile retrieved by each method. As can be seen in this figure, close to 75% of the ads retrieved by IRB are in deciles 5-10. LBQSB seems to retrieve an equal proportion of candidates across the deciles, while the click based CFB and QuAd-click-graph approaches have more ads in the higher deciles. Figure 6 plots the nCTR of each method in deciles 2-8. We did not include decile 1 because queries in that decile are mostly covered by standard match to the bidded phrases, decile 9 and 10 are also excluded because the rare nature of the queries makes the collected metrics noisy. Decile 10 has very few commercial queries (i.e., queries for which the user is likely to click on an advertisement of a product for) and therefore very few clicks. The nCTR values in Fig 6 are scaled such that the nCTR of IRB in decile 2 is pegged at 1. All methods have a decrease in nCTR across deciles, partly due to the decrease in the number of commercial queries in lower deciles. IRB has little decrease in nCTR in the lower deciles despite retrieving a large relatively larger number of candidates. CFB shows a tendency of improved performance in the middle deciles. Our QuAd Click graph method has the best performance in most deciles.

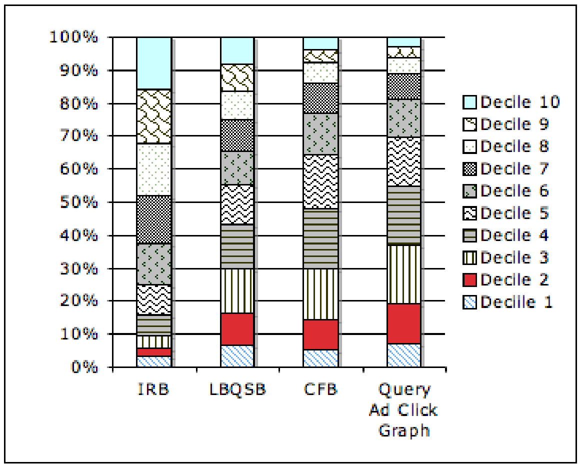

Figure 5: The graph shows the proportion of ads per decile retrieved by each method

7. Discussion and Conclusions

We have proposed a collaborative filtering algorithm for sponsored search ad retrieval that operates on a large bipartite graph of queries and ads to predict new ads likely to be clicked based on click data for related query-ad pairs. Our approach finds related queries based on a correlation measure over the query-ad graph, and then ranks candidate ads based on their average (weighted by query rewrite similarity score) expected click propensity over related queries. We compare our approach to three common baselines and find that we consistently improve performance. Editorial evaluations found good average quality for our rewrites offline, and we verify the algorithm with online bucket tests. We find that ads proposed by our approach have a greater normalized click-through-rate and also perform well in the re-ranking stage of the ad presentation system.

Analysis by query frequency decile shows that our performance declines with search frequency, probably due to increased noise in click graph statistics for lower volume queries. Future work

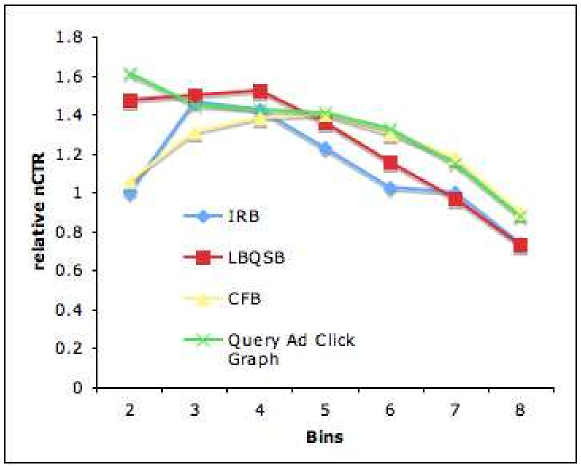

Figure 6: Queries are sorted by frequency and binned into deciles. The graph shows performance of each method in each bin

will evaluate approaches for improving the performance for infrequent queries, using smoothing approaches for click metrics as well as incorporating text-based relevance features such as those used in the information retrieval model. In addition, we are also working on models to consider relevance based filtering of the ads suggested by our proposed method in order to remove suggestions that travel to far in meaning from the original query.

References

[1] `www.emarketer.com/Article.aspx?id=1006319`.

[2] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR*, 2006.

[3] I. Antonellis, H. G. Molina, and C. C. Chang. Simrank++: query rewriting through link analysis of the click graph. *Proc. VLDB Endow.*, 1(1), 2008.

[4] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *KDD*, 2000.

[5] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, 1998.

[6] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2), 2002.

[7] A. Z. Broder, P. Ciccolo, M. Fontoura, E. Gabrilovich, V. Josifovski, and L. Riedel. Search advertising using web relevance feedback. In *CIKM*, 2008.

[8] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML*, 2005.

[9] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *WWW*, 2009.

[10] K. Chen, R. Lu, C. K. Wong, G. Sun, L. Heck, and B. Tseng. Trada: tree based ranking function adaptation. In *CIKM*, 2008.

[11] M. Ciaramita, V. Murdock, and V. Plachouras. Online learning from click data for sponsored search. In *WWW*, 2008.

[12] N. Craswell and D. Hawking. Overview of the trec 2004 web track. In *13th Text REtrieval Conference (TREC 2004)*, November 2004.

[13] N. Craswell and M. Szummer. Random walks on the click graph. In *SIGIR*, 2007.

[14] G. E. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *SIGIR*, 2008.

[15] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1), March 2007.

[16] A. Fuxman, P. Tsaparas, K. Achan, and R. Agrawal. Using the wisdom of the crowds for keyword generation. In *WWW*, 2008.

[17] X. Geng, T.-Y. Liu, T. Qin, A. Arnold, H. Li, and H.-Y. Shum. Query dependent ranking using k-nearest neighbor. In *SIGIR*, 2008.

[18] B. Jansen and M. Resnick. Examining searcher perceptions of and interactions with sponsored results. In *Workshop on Sponsored Search Auctions*, 2005.

[19] G. Jeh and J. Widom. Scaling personalized web search. In *WWW*, 2003.

[20] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting click-through data as implicit feedback. In *SIGIR*, 2005.

[21] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW*, 2006.

[22] R. Kohavi, R. M. Henne, and D. Sommerfield. Practical guide to controlled experiments on the web: listen to your customers not to the hippo. In *KDD*, 2007.

[23] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, 2008.

[24] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *CIKM*, 2008.

[25] V. Murdock, M. Ciaramita, and V. Plachouras. A noisy-channel approach to contextual advertising. In *ADKDD '07: Workshop on Data mining and audience intelligence for advertising*, 2007.

[26] F. Radlinski, A. Broder, P. Ciccolo, E. Gabrilovich, V. Josifovski, and L. Riedel. Optimizing relevance and revenue in ad search:a query substitution approach. In *SIGIR*, 2008.

[27] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *KDD*, 2005.

[28] H. Raghavan and R. Iyer. Evaluating vector-space and probabilistic models for query to ad matching. In *SIGIR '08 Workshop on Information Retrieval in Advertising (IRA)*, 2008.

[29] B. Ribeiro-Neto, M. Cristo, P. B. Golgher, and E. S. de Moura. Impedance coupling in content-targeted advertising. In *SIGIR*, 2005.

[30] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *WWW*, 2007.

[31] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR*, 2002.

[32] B. Shaparenko, O. Cetin, and R. Iyer. Data driven text features for sponsored search click prediction. In *AdKDD Workshop (KDD'09)*, 2009.

[33] W. tau Yih, J. Goodman, and V. R. Carvalho. Finding advertising keywords on web pages. In *WWW*, 2006.

[34] A. Töscher, M. Jahrer, and R. Legenstein. Improved neighborhood-based algorithms for large-scale recommender systems. In *2nd Netflix-KDD Workshop (KDD'08)*, 2008.

[35] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang. Clustering user queries of a search engine. In *WWW*, 2001.

[36] W. V. Zhang and R. Jones. Comparing click logs and editorial labels for training query rewriting. In *WWW 2007 Workshop on Query Log Analysis: Social And Technological Challenges*, 2007.

[37] Z. Zhang and O. Nasraoui. Mining search engine query logs for query recommendation. In *WWW*, 2006.