Clickthrough Log Analysis by Collaborative Ranking

Bin Cao[1], **Dou Shen**[2], **Kuansan Wang**[3], **Qiang Yang**[1]
[1] Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
[2] Microsoft, One Microsoft Way, Redmond WA 98052
[3] Microsoft Research, One Microsoft Way, Redmond WA 98052
[1]{caobin,qyang}@cse.ust.hk
[2,3]{doushen,Kuansan.Wang}@microsoft.com

Abstract

Analyzing clickthrough log data is important for improving search performance as well as understanding user behaviors. In this paper, we propose a novel collaborative ranking model to tackle two difficulties in analyzing clickthrough log. First, previous studies have shown that users tend to click top-ranked results even they are less relevant. Therefore, we use pairwise ranking relation to avoid the position bias in clicks. Second, since click data are extremely sparse with respect to each query or user, we construct a collaboration model to eliminate the sparseness problem. We also find that the proposed model and previous popular used click-based models address different aspects of clickthrough log data. We further propose a hybrid model that can achieve significant improvement compared to the baselines on a large-scale real world dataset.

Introduction

Search Engines retrieve relevant information for a given query and return the results in a ranked list. Users will examine the list and click on the "perceived" relevant results, which generates click logs as implicit relevance feedback to search engines. A popular search engine can collect a huge amount of clickthrough log every day, which contains a large amount of valuable information on relevance feedback. Analyzing such clickthrough log data is important for improving search performance as well as understanding user behaviors (Kelly and Teevan 2003).

However, we are facing two challenges for utilizing clickthrough log effectively.

- *Bias problem:* As shown in (Joachims et al. 2007a), clicks are informative yet biased. The ranked list representations of search results cause the bias towards top-ranked results. It is better to use them as relative judgment rather than absolute judgment. Most of previous work on modeling clickthrough data (Craswell and Szummer 2007; Xue et al. 2004) utilized co-click information directly. As a result, they cannot avoid the bias contained in the clicks. In this paper, we propose to exploit pairwise preference instead when analyzing clickthrough log.
- *Sparseness problem:* Clickthrough log is very sparse at the long tail. We cannot get click information for most of the urls ranked behind the first page. Models treating queries individually and independently would seriously suffer from the sparseness problem. Therefore, we would like to find the connection between queries and leverage the information between similar queries.

Copyright © 2010, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

In this paper, we develop a novel collaborative ranking framework to tackle the above issues. The proposed model unifies learning to rank models and latent factor models seamlessly. On the one hand, we consider the pairwise preference information rather than click information to remove the position bias in clickthrough log. On the other hand, we utilize the idea from collaborative filtering (Liu and Yang 2008) to solve the sparseness problem by leveraging information of similar queries. Following the consideration of preference relation, the basic assumption is that similar queries are the ones have many common preference observations and they could share more preference information of their own. We consider the preference observations are affected by some latent factors of queries and urls and we uncover the latent factors that generate the preference observations using a matrix factorization model. The matrix factorization model is optimized with respect to the observed preference rather than classical click information as in random walk models. Despite its powerful modeling capability, the computational complexity of our method is linear to the number of preference pairs in the clickthrough log data rather than the size of the matrix formed by <query, url> pairs, which guarantees that our method is applicable on large-scale data sets. We test our proposed method using a huge set of real world data, which consists of more than 1 million queries and 9 million urls. Experimental results validate the effectiveness of our method. From our experiments, We find that our proposed collaborative ranking model and the random walk models address different aspects of the information contained in clickthrough log data. Our model emphasizes the skip information and ranking relation between urls while the random walk models stress the click information. We further propose a hybrid model to combine the power of both and achieve a significant improvement of the performance.

Related Work

Implicit Feedback

Efforts have been made to utilize clickthrough log to improve search performance. Shen et al. in (Shen, Tan, and Zhai 2005) utilize previous queries and clickthrough information to reduce the ambiguity caused by sparseness and use it to find the relevant documents. Craswell and Szummer try to solve the sparseness problem of clickthrough by using a random walk model on a bipartite graph constructed by clicked <query, url> pairs (Craswell and Szummer 2007). Similar co-click based models are also used in (Xue et al. 2004). However, such co-click based models are constructed based on the click information which is usually noisy and biased. Besides that, co-click models only care about clicked urls and discard the information implied by the skipped urls, which is also very important for inferring relevance scores.

To reduce the noise and bias in clickthrough log, a class of approaches utilize additional information such as dwell time, users' behaviors after clicks, etc (Agichtein, Brill, and Dumais 2006). An alternative way is to use preference information contained in clicks (Joachims et al. 2005). We adopt the latter approach in this paper.

Learning to Rank

Many learning to rank algorithms have been proposed recently. Most of them try to solve the ranking problems by constructing proper cost functions for ranking problems (Taylor et al. 2008; Xu et al. 2008) or adapting algorithms from classification or regression problems (Xu and Li 2007). These algorithms obtain promising results on some open test datasets. However, the need of heuristic feature extraction and manually labeled training data hinders them in many real world applications.

There are some previous work (Agichtein, Brill, and Dumais 2006; Joachims et al. 2007b) combining the implicit feedback information extracted from the clickthrough log data into the learning to rank framework to avoid data labeling. Agichtein et al. improve the performance of learning to rank algorithms on the top rankings by utilizing many features extracted from clickthrough log. They compare two alternatives of incorporating implicit feedback into the search process, namely reranking with implicit feedback and incorporating implicit feedback features directly to train ranking functions. Joachims et al. use the extracted preference pairs as training data to learn a ranking function (Joachims et al. 2007b), which shares the same spirit with our work. But they rely on predefined features. In this paper, we consider a problem of learning with latent features, which is similar to the problem of collaborative filtering.

Collaborative Filtering/Ranking

Collaborative ranking (CR) has recently been exploited in (Weimer et al. 2007) and (Liu and Yang 2008) to improve the performance of collaborative filtering (CF). Liu et al. in (Liu and Yang 2008) extend the traditional memory based methods while (Weimer et al. 2007) extends the model based approach from rating based models to ranking based models. In collaborative filtering problems, the ranking list is easy to obtain by sorting the items according to their ratings. Therefore the cost functions based on ranking list are widely adopted (Weimer et al. 2007) and (Liu and Yang 2008). However, for Web search, it is almost impossible to obtain such a ranking list since the users do not assign scores to the returned Web pages explicitly. Therefore, we can only infer the pairwise preference information based on users' clicks and develop collaborative ranking algorithms with pairwise ranking cost functions.

Collaborative Ranking for Clickthrough Log Analysis

In this section, we present our solution of using collaborative ranking for clickthrough log analysis. We first introduce the cost function based on pairwise preference information and then present the model for discovering latent factors. After that, we show how to integrate the ranking cost function and the latent factor models into a unified objective function. Finally, we present the approach of optimizing the objective function.

Pairwise Based Ranking Objective

In (Joachims 2002), Joachims proposes to generate pairwise preference from clickthrough log as training data for learning to rank algorithms. In (Joachims et al. 2005; Radlinski and Joachims 2005), they further investigate different ways to generate pairwise preference from clickthrough log. The experiments show that as interpreted as relative preference, clickthrough is consistent with explicit judgment. In this paper, we use the rule in (Joachims 2002), "clicked url $\succ$ skipped url above" ("$\succ$" mean "better than", or "more relevant than"), to extract pairwise preference and leave the comparison of different extraction methods as our future work.

In the literature of learning to rank, several different cost functions have been proposed. In this paper, we adopt the likelihood cost used in (Burges et al. 2005). Let r_{ij} be the relevance score for url j to the query i. For the observed pairwise preference that url j is better than url k, we use the Bradley-Terry model (Burges et al. 2005) to define its likelihood.

$$P(r_{ij} \succ r_{ik}) = \varphi(r_{ij} - r_{ik}) = \frac{1}{1 + e^{-(r_{ij} - r_{ik})}}$$

where $\varphi(x) = \frac{1}{1+e^{-x}}$ and $r_{ij} \succ r_{ik}$ means that result j is better than result k for the query i. For each query i, we may have a set of pairwise preference $< j, k >_i$.

Suppose we have obtained the relevant score matrix, we are able to calculate the log likelihood of all preference observations.

$$l = \log \prod_{i, <j,k>_i} P(r_{ij} \succ r_{ik}) = \sum_i \sum_{<j,k>_i} \log(\varphi(r_{ij} - r_{ik}))$$

Let $\mathbf{d}_{j \succ k}$ be a vector with element j being 1 and k being -1, others being 0. Then we can rewrite the cost function as

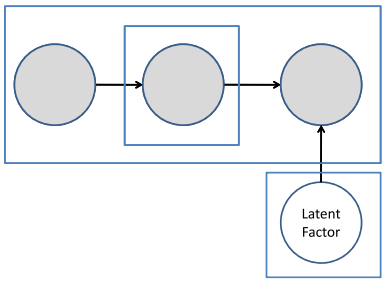

Figure 1: Graph representation of the latent factor model with preference observation.

$$l = \sum_i \sum_{<j,k>_i} \log(P(r_{ij} \succ r_{ik})) = \sum_i \sum_{<j,k>_i} \log \varphi(\mathbf{r}_i \mathbf{d}_{j \succ k}^{\mathrm{T}}) \tag{1}$$

where $\mathbf{r}_i$ is the vector of relevance scores of query i to the urls.

It is hard to estimate the relevant scores by directly maximizing the likelihood due to its huge dimension. A general solution is to reduce the number of variables we need to learn. Learning to rank uses a function of predefined feature set for this propose. Our method, from the other perspective, solves this problem by integrating latent factor models, which can automatically extract implicit features from the clickthrough log data.

Matrix Factorization Model

Latent factor models are able to uncover the underlying factors that determine the generation of observed data. For example, the PLSA model could find the latent topics in texts (Hofmann 1999).

When we are considering the ranking problem, what only matters is the relative values of the scores rather than the absolute values. Therefore, any scaling factor with respect to each query does not change the ranking results. Therefore, it is unnecessary to constrain the value of relevance score to be a probability that is between zero and one. Hence, we simplify the model by just considering the following matrix factorization problem.

$$\mathbf{R} = \mathbf{Q}\mathbf{U}^{\mathrm{T}} \tag{2}$$

where $\mathbf{R}$ is the relevance score matrix with rows representing queries and columns representing urls, $\mathbf{Q}$ is the latent representation for queries and $\mathbf{U}$ is the latent representation for urls. Usually, we will assume there are only a relative small number of latent factors that are influencing the relevance scores. So both $\mathbf{Q}$ and $\mathbf{U}$ are low-rank matrices.

Ranking Objective with Latent Factors

We would like to learn the latent factors that can best fit the observations of preference. Therefore, our objective would be the likelihood of

$$p(\{r_{ij} \succ r_{ik}\}|\mathbf{R}) \tag{3}$$

where $\{r_{ij} \succ r_{ik}\}$ is the set of preference observations. Unifying the Bradley-Terry model and the matrix factorization model, we can obtain the log-likelihood function by plugging Eq. 2 into Eq. 1,

$$l = \sum_i \sum_{<j,k>_i} \log \varphi(\mathbf{q}_i \mathbf{U}^{\mathrm{T}} \mathbf{d}_{j \succ k}^{\mathrm{T}})$$

where $\mathbf{q}_i$ is the ith row of $\mathbf{Q}$.

For the matrices $\mathbf{Q}$ and $\mathbf{U}$, we also assign them prior distributions with Gaussian distributions,

$$p(q_{ij}) \sim \mathcal{N}(0, \sigma_q^2), \quad p(u_{ij}) \sim \mathcal{N}(0, \sigma_u^2)$$

where q_{ij} and u_{ij} are the elements of $\mathbf{Q}$ and $\mathbf{U}$. σ_q and σ_u are parameters controlling the confidence of prior. The prior distributions have the same effect as regularization terms in maximum margin learning (Weimer et al. 2007), which could prevent overfitting. Then we obtain the posterior distribution as our objective function,

$$l = \sum_i \sum_{<j,k>_i} \log \varphi(\mathbf{q}_i \mathbf{U}^{\mathrm{T}} \mathbf{d}_{j \succ k}^{\mathrm{T}}) + \log(p(\mathbf{Q})) + \log(p(\mathbf{U})) \tag{4}$$

where

$$\log(p(\mathbf{Q})) = -\frac{1}{2} \sum_i \mathbf{q}_i^{\mathrm{T}} \mathbf{q}_i / \sigma_q^2 + C_q$$

and

$$\log(p(\mathbf{U})) = -\frac{1}{2} \mathrm{vec}(\mathbf{U})^{\mathrm{T}} \mathrm{vec}(\mathbf{U}) / \sigma_u^2 + C_u$$

$\mathrm{vec}(\mathbf{U})$ is the vectorization of the matrix $\mathbf{U}$, and C_q and C_u are constants irrelevant to $\mathbf{Q}$ and $\mathbf{U}$ respectively.

The above model is similar to the probabilistic matrix factorization model (Salakhutdinov and Mnih 2008). However, we consider a ranking loss instead of the square loss in our problem.

Optimization Algorithm

Gradient based optimization algorithms can be used to optimize the above objective function. The gradients of the cost function with respect to $\mathbf{Q}$ and $\mathbf{U}$ are shown in Equation (5) and (6).

$$\frac{\partial l}{\partial \mathbf{q}_i} = \sum_{<j,k>_i} \frac{\varphi'(\mathbf{q}_i \mathbf{U}^{\mathrm{T}} \mathbf{d}_{j \succ k}^{\mathrm{T}})}{\varphi(\mathbf{q}_i \mathbf{U}^{\mathrm{T}} \mathbf{d}_{j \succ k}^{\mathrm{T}})} \mathbf{d}_{j \succ k} \mathbf{U} - \mathbf{q}_i / \sigma_q^2 \tag{5}$$

$$\frac{\partial l}{\partial \mathbf{U}} = \sum_i \sum_{<j,k>_i} \frac{\varphi'(\mathbf{q}_i \mathbf{U}^{\mathrm{T}} \mathbf{d}_{j \succ k}^{\mathrm{T}})}{\varphi(\mathbf{q}_i \mathbf{U}^{\mathrm{T}} \mathbf{d}_{j \succ k}^{\mathrm{T}})} \mathbf{d}_{j \succ k}^{\mathrm{T}} \mathbf{q}_i - \mathbf{U} / \sigma_u^2 \tag{6}$$

where

$$\varphi'(x) = \frac{e^{-x}}{(1 + e^{-x})^2}$$

In computation, when $-x$ is so large that e^{-x} overflows, the result of above equation would be ∞/∞. To avoid the numeric issue, the following equivalent equation is used.

$$\varphi'(x) = \frac{1}{(1 + e^{x})(1 + e^{-x})}$$

When $|x|$ is large, $\varphi'(x)$ would be equal to 0.

There are different gradient-based algorithms available for such a non-constraint optimization problem. We use a gradient descent algorithm to optimize the objective function, as shown in Algorithm 1. The complexity of calculating the gradient is in the order of number of preference pairs in the training data rather than the size of queries urls matrix. Therefore, our algorithm can handle large-scale dataset with millions of queries and urls.

Algorithm 1 Calculating collaborative ranking model.

Require: Initializing learning rate α, iteration number T, query feature $\mathbf{q}_i$ and url feature $\mathbf{U}$.
Ensure: $\mathbf{R} = \mathbf{QU}$
  for $t = 0$ to T **do**
    for $i = 1$ to $|\text{Query}|$ **do**
      $\mathbf{q}_i \leftarrow \mathbf{q}_i + \alpha \frac{\partial l}{\partial \mathbf{q}_i}$
    end for
    $\mathbf{U} \leftarrow \mathbf{U} + \alpha \frac{\partial l}{\partial \mathbf{U}}$
  end for

Discussion

In this section, we discuss the relation between the proposed collaborative ranking model and the popular used random walk models. We refer to the matrix of click frequency from queries to urls as a click matrix C. We denote the preference matrix of queries to preference pair by P. An example is shown in Figure 2. In the figure, the above table shows a click matrix C of four related queries ($q_a - q_d$) and three urls(url 1-3). Assuming the returned url lists are all (url 1, url 2, url 3), we can obtain the preference matrix P as shown in the below figure. The question marks in the preference matrix represent the unknown values. A simple analysis is able to show the differences between the two matrices. Considering the similarity between the queries using the co-click information and co-preference information, the most similar pairs in C would be $< q_a, q_b >, < q_a, q_c >, < q_a, q_d >, < q_b, q_d >, < q_c, q_d >$ since they all share one common click. However, the most similar pairs in P would be $< q_a, q_c >, < q_c, q_d >$, which both have shared click on url 3. If two queries have co-click urls at the tail of the list, they tend to have more co-preference pairs. Therefore, the preference matrix put more emphasis on the clicks in the tails. This would alleviate the position bias problem in clickthrough data.

We further consider the random walk model in (Craswell and Szummer 2007),

$$P_{0|t}(k|j) = [\frac{1}{Z} A(\dots(A(Aq_j)))]_k \quad (7)$$

where A is transition probability matrix from queries to urls calculated by normalizing click matrix C, and q_j is the click probabilistic distribution of query j. Z is a normalization factor to make the distribution valid.

On the one hand, the random walk models are essentially the power method to compute the eigenvectors of the click matrix (Golub and Loan 1996). On the other hand, matrix factorization is another way to compute the eigenvectors of the matrix (Golub and Loan 1996). They are equivalent in some sense. Therefore, random walk models can be regarded as approximating the click matrix using the eigenvectors. While the collaborative ranking model aim at approximating the preference matrix using the matrix factorization model. They are similar in the approximation models but different in the information addressed. Based on the analysis, we can further propose a hybrid model to combine the two types of information together.

	url 1	url 2	url 3
Query A	√	X	√
Query B	√	√	X
Query C	X	X	√
Query D	X	√	√

	url 3 >url 2	url 3>url 1	url 2>url 1
Query A	√	?	?
Query B	?	?	?
Query C	√	√	?
Query D	?	√	√

Figure 2: An example of the click matrix and preference matrix.

A Hybrid Model

The collaborative ranking model has the advantage of utilizing the skip information. However, it ignores the fact that most of the un-clicked urls are irrelevant, which is captured by the random walk models. Therefore, it would be better to unify them into a hybrid model for a full consideration. We propose to use a simple linear combination to unify them together. We will show in the experiments that this method works well.

$$r_{q,u} = (1-\theta) r^r_{q,u} + \theta r^c_{q,u}$$

where the $r^r_{q,u}$ is the relevance score obtained by the collaborative ranking model and the $r^c_{q,u}$ is the relevance score obtained by the random walk model. θ is a tradeoff value in $[0, 1]$.

Since the two models address different aspects of information in clickthrough log. We will show in the experiments part that the hybrid model could achieve significant improvement.

Experiments

Datasets and Evaluation Metrics

We also use a search log dataset sampled from a major commercial search engine for experiment. The dataset contains 1,604,517 unique queries, 9,999,873 unique urls and 13,947,439 clicks. In terms of evaluation, we use accuracy on predicted pairwise preference as our evaluation metric. We first extract all the preference pairs from the data. Then we randomly split it into two sets. We use one for training

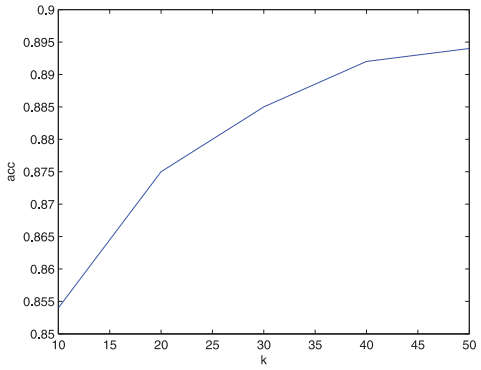

Figure 3: How the performance change with respect to the number of latent factors k.

and the other for test. For the test data, we have the ground truth of preference relation of the pairs (either $i \prec j$ or $i \succ j$ for the pair $< i, j >$) and we can define the accuracy by

$$acc = \frac{N_{acc}}{N},$$

where N_{acc} is the number of correctly predicted preference pairs in the test data. N is the number of all preference pairs in the test data. This evaluation metric avoids the effort of labeling data. Therefore, with such an unsupervised metric we can evaluate our method on large-scale datasets.

Experimental Results

In the following, we want to show that our proposed method can effectively leverage the pairwise preference information in large-scale clickthrough log data. Therefore, we compare our method with the co-click based models in (Craswell and Szummer 2007) on the real world dataset. For the models in (Craswell and Szummer 2007), we use the parameters presented in the paper. The results of both backward and forward random walk models in the paper are reported in Table 1. From the results we can see, the backward model outperforms the forward model, which is consistent with the experiments in (Craswell and Szummer 2007).

For the co-click models, the underlying assumption is "clicked url$\succ$un-clicked url". Although it introduces bias in clicks, it utilizes more information in general. For our current model, since we only utilize the preference that "clicked url$\succ$skipped urls above", much of the information that un-clicked urls are not relevant is not utilized. Nevertheless, the performance of our model still slightly outperforms the state-of-the-art algorithms.

In terms of parameters, the number of latent variables k used in this experiment is 50 and the iteration number T is 50. Figure 3 shows the performance change with respect to k, we can see that the trend is still increasing for $k = 50$. But for speed consideration, we choose $k = 50$ in the experiment.

Figure 4 shows how the performance change with θ varies. It is interesting to notice that the combined performance is very robust to the selection of θ. The reason is

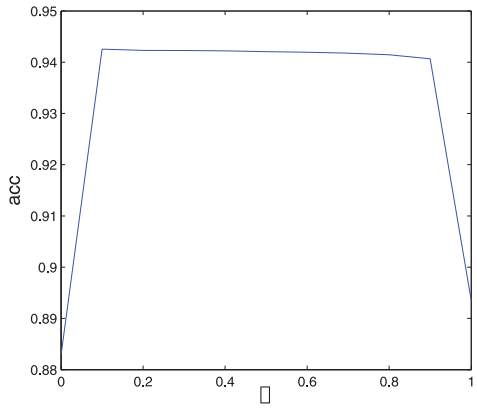

Figure 4: How the performance of combined method change with respect to θ.

Table 1: Performance comparison with random walk models.

	Forward	Backward	CoRank	Hybrid
acc	0.87	0.88	0.89	0.94

that the random walk model ignores the information of being skipped. On the other hand, the collaborative ranking model ignores the information of being un-clicked. These two kinds of information are principally orthogonal so that the simple combination works well. It is also an interesting topic to directly combine these two types of information in a unified model. We plan to investigate it in our future work.

Conclusion and Future Work

In this paper, we propose a novel collaborative ranking model for improving Web search using clickthrough log. Our model integrates the advantage of leveraging pairwise preference information and mining latent factors both from clickthrough log in a principled way. Experiments on real-world data validate the effectiveness of the proposed model.

Theoretically, our model can be easily extended to consider different ranking cost functions and latent factor models. In the future, we plan to investigate how different combinations would influence the performance empirically. Besides that, we will take the confidence of each preference pair into consideration and see whether it can boot the search performance. We also plan to investigate more elegant model to combine the collaborative ranking model and random walk model together. A direction is to use the relevance values obtained by random walk model as prior for the collaborative ranking model.

As we can see, a major problem of our current model is that it cannot handle new queries, which have never shown up in the clickthrough log data. To solve this problem, we plan to put forward an online learning version for this model.

Acknowledgments

Bin Cao and Qiang Yang thank the support of RGC/NSFC grant N_HKUST624/09.

References

Agichtein, E.; Brill, E.; and Dumais, S. 2006. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual ACM SIGIR conference*, 19–26. Seattle, Washington, USA: ACM.

Burges, C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N.; and Hullender, G. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd ICML*, 89–96. Bonn, Germany: ACM.

Craswell, N., and Szummer, M. 2007. Random walks on the click graph. In *Proceedings of the 30th annual ACM SIGIR conference*, 239–246. Amsterdam, The Netherlands: ACM.

Golub, G., and Loan, C. V. 1996. *Matrix computations*.

Hofmann, T. 1999. Probabilistic latent semantic analysis. In *In Proc. of Uncertainty in Artificial Intelligence, UAI '99*, 289–296.

Joachims, T.; Granka, L.; Pan, B.; Hembrooke, H.; and Gay, G. 2005. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th annual ACM SIGIR conference*, 154–161. Salvador, Brazil: ACM.

Joachims, T.; Granka, L.; Pan, B.; Hembrooke, H.; Radlinski, F.; and Gay, G. 2007a. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Trans. Inf. Syst.* 25(2):7.

Joachims, T.; Li, H.; Liu, T.-Y.; and Zhai, C. 2007b. Learning to rank for information retrieval (lr4ir 2007). *SIGIR Forum* 41(2):58–62.

Joachims, T. 2002. Evaluating retrieval performance using clickthrough data. *In Proceedings of the SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval* 79—96.

Kelly, D., and Teevan, J. 2003. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum* 37(2):18–28.

Liu, N. N., and Yang, Q. 2008. Eigenrank: a ranking-oriented approach to collaborative filtering. In *SIGIR '08: Proceedings of the 31st annual ACM SIGIR conference*, 83–90. New York, NY, USA: ACM.

Radlinski, F., and Joachims, T. 2005. Query chains: learning to rank from implicit feedback. In *Proceedings of the eleventh ACM SIGKDD conference*, 239–248. Chicago, Illinois, USA: ACM.

Salakhutdinov, R., and Mnih, A. 2008. Probabilistic matrix factorization. In Platt, J.; Koller, D.; Singer, Y.; and Roweis, S., eds., *Advances in Neural Information Processing Systems 20*. Cambridge, MA: MIT Press. 1257–1264.

Shen, X.; Tan, B.; and Zhai, C. 2005. Context-sensitive information retrieval using implicit feedback. In *Proceedings of the 28th annual ACM SIGIR conference*, 43–50. Salvador, Brazil: ACM.

Taylor, M.; Guiver, J.; Robertson, S.; and Minka, T. 2008. Softrank: optimizing non-smooth rank metrics. In *Proceedings of the international conference on Web search and web data mining*, 77–86. Palo Alto, California, USA: ACM.

Weimer, M.; Karatzoglou, A.; Le, Q.; and Smola, A. 2007. Cofi rank - maximum margin matrix factorization for collaborative ranking. In *Advances in Neural Information Processing Systems 20*, 1600, 1593. MIT Press.

Xu, J., and Li, H. 2007. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual ACM SIGIR conference*, 391–398. Amsterdam, The Netherlands: ACM.

Xu, J.; Liu, T.-Y.; Lu, M.; Li, H.; and Ma, W.-Y. 2008. Directly optimizing evaluation measures in learning to rank. In *Proceedings of the 31st annual ACM SIGIR conference*, 107–114. Singapore, Singapore: ACM.

Xue, G.-R.; Zeng, H.-J.; Chen, Z.; Yu, Y.; Ma, W.-Y.; Xi, W.; and Fan, W. 2004. Optimizing web search using web click-through data. In *CIKM '04*, 118–126. New York, NY, USA: ACM.