Active Exploration for Learning Rankings from Clickthrough Data

Filip Radlinski
Department of Computer Science
Cornell University
Ithaca, NY, USA
filip@cs.cornell.edu

Thorsten Joachims
Department of Computer Science
Cornell University
Ithaca, NY, USA
tj@cs.cornell.edu

ABSTRACT

We address the task of learning rankings of documents from search engine logs of user behavior. Previous work on this problem has relied on passively collected clickthrough data. In contrast, we show that an active exploration strategy can provide data that leads to much faster learning. Specifically, we develop a Bayesian approach for selecting rankings to present users so that interations result in more informative training data. Our results using the TREC-10 Web corpus, as well as synthetic data, demonstrate that a directed exploration strategy quickly leads to users being presented improved rankings in an online learning setting. We find that active exploration substantially outperforms passive observation and random exploration.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Measurement, Performance

Keywords

Clickthrough data, Web search, Active exploration, Learning to rank

1. INTRODUCTION

There has recently been an interest in training search engines automatically using machine learning (e.g. [20, 4, 24]). The ideal training data would be rankings of documents ordered by relevance for some set of queries. In some cases it is practical to hire experts to manually provide relevance information for particular queries (as in [4]), but usually data provided by experts is too expensive and is not guaranteed to agree with the judgments of regular users. This is a particular problem when typical user queries are short and ambiguous, as is often the case in web search.

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. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
KDD'07, August 12–15, 2007, San Jose, California, USA.
Copyright 2007 ACM 978-1-59593-609-7/07/0008 ...$5.00.

As an alternative source of training data, previous work has used clickthrough logs that record user interactions with search engines. However, as far as we are aware, all previous work has only used logs collected passively, simply using the recorded interactions that take place anyway. We instead propose techniques to guide users so as to provide more useful training data for a learning search engine.

To see the limitation of passively collected data, consider the typical interactions of search engine users. Usually, a user executes a query then perhaps considers the first two or three results presented [14, 15]. The feedback (clicks) on these results is recorded and used to infer relevance judgments. These judgments are then used to train a learning algorithm such as a ranking support vector machine [18]. In particular, users very rarely evaluate results beyond the first page, so the data obtained is strongly biased toward documents already ranked highly. Highly relevant results that are not initially ranked highly may never be observed and evaluated, usually leading to the learned ranking never converging to an optimal ranking.

To avoid this presentation effect, we propose that the ranking presented to users be optimized to obtain useful data, rather than strictly in terms of estimated document relevance. For example, one possiblility would be to intentionally present unevaluated results in the top few positions, aiming to collect more feedback on them. However, such an ad-hoc approach is unlikely to be useful in the long run and would hurt user satisfaction. We instead introduce principled modifications that can be made to the rankings presented. These changes, which do not substantially reduce the quality of the ranking shown to users, produce much more informative training data and quickly lead to higher quality rankings being shown to users.

The primary contribution of this paper is to present a principled approach to efficiently obtaining training data that leads to rankings of higher quality. We start by presenting a summary of observations about user behavior in Section 2, as how real users behave guides our approach. Next, in Section 3 we formalize the learning problem as an optimization task, present a suitable Bayesian probabilistic model and discuss inference and learning. Following this we present strategies to modify the rankings shown to users so that performance of learned rankings improves rapidly over time in Section 4. We describe our evaluation method in Section 5 and present results on synthetic data and TREC-10 Web data in Section 6. In particular, we see the improvements using our exploration strategies are much faster than with passive or random data collection.

2. USER BEHAVIOR

Learning rankings relies on training data collected from users. We will now examine the specific properties of user behavior as they will guide our approach for the remainder of this work. In particular we will see what sort of data can reasonably be collected from clickthrough logs and how we should include user behavior in our learning task.

A number of studies have shown that users tend to click on results ranked highly by search engines much more often than those ranked lower. For example, in recent work Agichtein et al.[1] present a summary distribution of the relative click frequency on web search results for a large search engine as a function of rank for 120,000 searches for 3,500 queries. They show that the relative number of clicks rapidly drops with the rank – compared with the top ranked result, they observe approximately 60% as many clicks on the second result, 50% as many clicks on the third, and 30% as many clicks on the fourth. While we may hypothesize that this is simply because better results tend to be presented higher, Joachims et al.[21] showed that there is an inherent bias to rank in user behavior. They showed that users still click more often on higher ranked results even if presented with rankings reversing the top ten results. In fact, eye tracking studies performed by Granka et al.[15] show that the probability that users even *look at* a search results decays very quickly with rank.

On the one hand, this explains why most common performance measures in information retrieval place greater emphasis on highly ranked results (such as Mean Average Precision, Normalized Discriminative Cumulative Gain and Mean Reciprocal Rank). On the other hand, this observation means that rank strongly influences how many times a document is evaluated by users, and hence how much training data can be collected from clickthrough logs.

Given the recorded clicks, the question also remains how best to interpret the log entries to infer relevance information about a document collection. Two alternative approaches to interpreting clickthrough logs are (1) consider a click on a document in a result set as an absolute sign of relevance, or (2) consider a click on a document as a pairwise judgment comparing the relevance of that document to some other document considered earlier. Specifically, the most common approach is to assume that a clicked on result is more relevant than a non-clicked higher ranked result. Joachims et al.[21] showed that interpreting clicks as relative relevance judgments is generally reliable, while absolute judgments are not. Such pairwise relevance judgments have been successfully used to learn improved rankings [20, 24].

A final observation of user behavior is that clickthrough logs are inherently very noisy. Users often click on search results without carefully considering them [15]. In particular this means that any single judgment that states that one document is more relevant than another has a significant probability of being incorrect. However, previous work has shown that if the difference in relevance between documents is larger, relative relevance judgments are less likely to be noisy [25]. This work also showed that if adjacent pairs of documents in the ranking shown to users are randomly swapped to compensate for presentation bias, provably reliable pairwise relevance judgments about adjacent pairs of documents can be collected.

We summarize that (1) Clickthrough data is best interpreted as relative relevance judgments; (2) The relevance judgments are noisy; (3) The top ranked documents are the most important to estimate correctly. Guided by these properties, we now turn to formalizing the learning problem.

3. LEARNING PROBLEM

Assume we have a document corpus $\mathcal{C} = \{d_1, \ldots, d_{|\mathcal{C}|}\}$ and some fixed user query q. For this query, we want to estimate the average relevance $\mu_i^* \in \Re$ of each document d_i (for some user population). From the previous section, we know that users can provide us with noisy judgments of the form $\mu_i^* > \mu_j^*$. We assume that some ranking function can provide initial estimates of μ_i^*. The goal is accurately estimate μ_i^* with as little training data as possible.

The estimation task involves a three step interative process: First, given relevance estimates, we must select a ranking to display to users. Second, given the ranking displayed users provide relevance feedback. Third, using the relevance feedback, we update the relevance estimates and repeat the process for the next user. In this paper, we focus most on the first step, namely selecting rankings of documents to show users so that the collected judgments allow the relevance estimates to be improved quickly, while at the same time maximizing the quality of the rankings.

3.1 Probabilistic Model

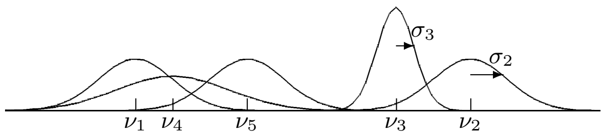
Let $M^* = (\mu_1^*, \ldots, \mu_{|\mathcal{C}|}^*) \in \mathcal{M}$ be the true relevance values of the documents in $\mathcal{C}$. Modeling the problem of finding M^* given training data $\mathcal{D}$ in a Bayesian framework, we want to maintain our knowledge about M^* in the distribution

$$P(M|\mathcal{D}) \quad = \quad \frac{P(\mathcal{D}|M)P(M)}{P(\mathcal{D})}$$

We assume that $P(M|\mathcal{D})$ is a multivariate normal with zero covariance:

$$P(M|\mathcal{D}) = \mathcal{N}(\nu_1, \ldots, \nu_{|\mathcal{C}|}; \sigma_1^2, \ldots, \sigma_{|\mathcal{C}|}^2) \qquad (1)$$

Graphically, we can draw $P(M|\mathcal{D})$ as a set of Gaussians centered at ν_i with variance σ_i^2. For example:



Relevance estimate

This model is motivated by ability estimates maintained for chess players [11]. In the most closely related previous work, Chu and Ghahramani [8] address a similar problem using Guassian Processes. However, instead maintaining the distribution $P(M|\mathcal{D})$, they directly estimate M^* given $\mathcal{D}$. This is also true of other related prior work [9, 10, 22]. The key difference in our approach is that we are not simply finding the optimizing ranking. Rather, maintaining $P(M|\mathcal{D})$ is key as it allows us to optimize for collected training data.

3.2 Inference

We measure the difference between relevance assignments using a loss function $\mathcal{L} : \mathcal{M} \times \mathcal{M} \rightarrow \Re$. To find good relevance estimates, we want to find an $M = (\mu_1, \ldots, \mu_{|\mathcal{C}|}) \in \mathcal{M}$ such that $\mathcal{L}(M, M^*)$ is small. Noting that M^* is unknown, we want to find the ranking that minimizes the expected loss given what we know about M^*, namely $P(M|\mathcal{D})$:

$$\underset{M}{\operatorname{argmin}}\, E_{M^* \sim P(M|\mathcal{D})}\left[\mathcal{L}(M, M^*)\right] \qquad (2)$$

where M^* is drawn from the probability distribution $P(M|\mathcal{D})$.

Suppose the loss function $\mathcal{L}$ can be decomposed over pairs of documents in $\mathcal{C}$. We can then decompose the expected loss into a form easier to work with:

$$\begin{aligned} & E_{P(M^*|\mathcal{D})}\left[\mathcal{L}(M, M^*)\right] \\ = \; & E_{P(M^*|\mathcal{D})}\left[\sum_{i=1}^{|\mathcal{C}|}\sum_{j=i+1}^{|\mathcal{C}|}\mathcal{L}^{pair}(M, M^*, i, j)\right] \\ = \; & \sum_{i=1}^{|\mathcal{C}|}\sum_{j=i+1}^{|\mathcal{C}|} E_{P(M^*|\mathcal{D})}\left[\mathcal{L}^{pair}(M, M^*, i, j)\right] \end{aligned} \quad (3)$$

where $P(M^*|\mathcal{D})$ is shorthand for $M^* \sim P(M|\mathcal{D})$.

We will now show that the mode of $P(M|\mathcal{D})$, namely $\hat{M} = (\nu_1, \ldots, \nu_{|\mathcal{C}|})$, is often the solution to Equation 2. Consider solving Equation 2 for a loss function that counts the number of misordered pairs of documents. The assignment with minimum expected loss is the mode of $P(M|\mathcal{D})$.

LEMMA 1. *Let $P(M|\mathcal{D}) = \mathcal{N}(\nu_1, \ldots, \nu_{|\mathcal{C}|}; \sigma_1, \ldots, \sigma_{|\mathcal{C}|})$ be a distribution over models. Assume $\mathcal{L}(M, M^*)$ counts the number of differently ordered pairs of documents, when they are sorted by μ_i and μ_i^* respectively. A solution of*

$$\operatorname*{argmin}_M E_{M^* \sim P(M|\mathcal{D})}\left[\mathcal{L}(M, M^*)\right]$$

is $\hat{M} = (\nu_1, \ldots, \nu_{|\mathcal{C}|})$.

PROOF. *Assume $M^{opt} = (\mu_1^{opt}, \ldots, \mu_{|\mathcal{C}|}^{opt})$ is the minimizing relevance assignment, and has lower loss than $\hat{M}$. There must exist two documents d_i and d_j that are ranked adjacently when documents are ordered by M^{opt} yet are ordered differently by $\hat{M}$, i.e. $\mu_i^{opt} > \mu_j^{opt}$ and $\nu_i < \nu_j$. Let M^{flip} be the ranking obtained by reversing d_i and d_j. Let these rankings have expected loss E^{flip} and E^{opt}.*

As the documents are adjacent, the loss of M^{opt} and M^{flip} only differs in the contribution of the pair (d_i, d_j). Plugging in the loss function we get

$$\begin{aligned} E^{flip} - E^{opt} &= P(\mu_i^* > \mu_j^*) - P(\mu_j^* > \mu_i^*) \\ &= \Phi\left(\frac{\nu_i - \nu_j}{\sqrt{\sigma_i^2 + \sigma_j^2}}\right) - \Phi\left(\frac{\nu_j - \nu_i}{\sqrt{\sigma_i^2 + \sigma_j^2}}\right) < 0 \end{aligned}$$

where Φ is the cumulative distribution function of the standard normal distribution, since $\nu_i < \nu_j$. Hence we have a contradiction as M^{opt} is not the minimizing ranking. $\square$

We see a similar result for the loss function that penalizes any error in the difference of document relevances, i.e. $\mathcal{L}^{pair}(M, M^*, i, j) = ((\mu_i - \mu_j) - (\mu_i^* - \mu_j^*))^2$.

LEMMA 2. *Let $P(M|\mathcal{D}) = \mathcal{N}(\nu_1, \ldots, \nu_{|\mathcal{C}|}; \sigma_1, \ldots, \sigma_{|\mathcal{C}|})$ be a distribution over models. Assume $\mathcal{L}^{pair}(M, M^*, i, j) = ((\mu_i - \mu_j) - (\mu_i^* - \mu_j^*))^2$. A solution of*

$$\operatorname*{argmin}_M E_{M^* \sim P(M|\mathcal{D})}\left[\mathcal{L}(M, M^*)\right]$$

is $\hat{M} = (\nu_1, \ldots, \nu_{|\mathcal{C}|})$.

PROOF. *Let M^{opt} be the minimizing model. Let $\delta_{ij}^{opt} = \mu_i^{opt} - \mu_j^{opt}$ be the difference in relevance estimates of d_i and d_j according to M^{opt}, and $\hat{\delta}_{ij}$ and δ_{ij}^* be defined equivalently for $\hat{M}$ and M^* respectively. Let $\sigma_{ij} = (\sigma_i^2 + \sigma_j^2)^{1/2}$. The contribution to the expected loss for the pair of documents (d_i, d_j) is*

$$\begin{aligned} & E_{P(M^*|\mathcal{D})}[\mathcal{L}^{pair}(M, M^*, i, j)] \\ = \; & \frac{1}{\sigma_{ij}\sqrt{2\pi}}\int_{-\infty}^{\infty}\exp\left(-\frac{(\delta_{ij}^* - \hat{\delta}_{ij})^2}{2\hat{\sigma}_{ij}^2}\right)(\delta_{ij}^* - \delta_{ij}^{opt})^2 d\delta_{ij}^* \\ = \; & \sigma_{ij}^2 + (\hat{\delta}_{ij} - \delta_{ij}^{opt})^2 \end{aligned}$$

which is minimized if $\delta_{ij}^{opt} = \hat{\delta}_{ij}$. Hence $M^{opt} = \hat{M}$ minimizes all terms in the sum in Equation 3 simultaneously and thus minimizes the expected loss. $\square$

We see that the mode of the distribution $P(M|\mathcal{D})$ minimizes the expected loss for two reasonable loss functions. As it can also be obtained very efficiently given $P(M|\mathcal{D})$, for the remainder of this work we will assume that the mode is, or is close to, the minimizer of the expected loss. We will refer to the ranking obtained by sorting documents by their relevance according to the mode of $P(M|\mathcal{D})$ as the *mode ranking*.

3.3 Loss Function

Given our analysis of real user behavior in Section 2, we see that the loss functions discussed above are too simple. Specifically, two properties to expect of an appropriate loss function are (1) The loss for ranking a less relevant document above a more relevant document should be larger if the documents are presented higher in the ranking (i.e. where users are more likely to observe them); (2) The loss should be larger if the difference in relevance is larger. To the best of our knowledge there is no common pairwise decomposable loss function with these properties so we propose a quadratic hinge-loss function with cost of misordering decaying exponentially with rank:

$$\mathcal{L}^{pair}(M, M^*, i, j) = e^{-r_{ij}}\left((\mu_i - \mu_j) - (\mu_i^* - \mu_j^*)\right)^2 \mathbf{1}_{misordered}$$

With r_{ij} we denote the minimum rank of d_i or d_j when all documents are ordered by M (i.e. the relevance assignments used to present results to users) divided by 10, and $\mathbf{1}$ is the indicator function. A pair of documents is considered misordered if the relative ranking according to M does not agree with that according to M^*. Making use of the pairwise form of the loss function and plugging in the mode ranking $\hat{M}$, the inner term of Equation 3 can now be written as

$$\begin{aligned} & E_{P(M^*|\mathcal{D})}[\mathcal{L}^{pair}(\hat{M}, M^*, i, j)] \\ = & \int P(\mu_i^*|\nu_i, \sigma_i)\int P(\mu_j^*|\nu_j, \sigma_j)\ \mathcal{L}^{pair}(\hat{M}, M^*, i, j)\ d\mu_i^* d\mu_j^* \\ = & \frac{1}{\sqrt{2\pi}\sigma_{ij}}\int_{-\infty}^{\infty}\exp\left(-\frac{(\delta_{ij}^* - \hat{\delta}_{ij})^2}{2\sigma_{ij}^2}\right)\mathcal{L}^{pair}(\delta_{ij}^*, \hat{\delta}_{ij}, r_{ij})\ d\delta_{ij}^* \end{aligned} \quad (4)$$

where $\delta_{ij}^* = \mu_i^* - \mu_j^*$, $\hat{\delta}_{ij} = \nu_i - \nu_j$ and $\sigma_{ij}^2 = \sigma_i^2 + \sigma_j^2$, noting that the difference of two normally distributed variables is also normally distributed. Plugging in the loss, and choosing to sum over the pairs such that $\hat{\delta}_{ij}$ is always negative, Equation 4 becomes:

$$\begin{aligned} &= \frac{e^{-r_{ij}}}{\sqrt{2\pi}\sigma_{ij}}\int_0^{\infty}\left(\delta_{ij}^* - \hat{\delta}_{ij}\right)^2\exp\left(-\frac{(\delta_{ij}^* - \hat{\delta}_{ij})^2}{2\sigma_{ij}^2}\right)d\delta_{ij}^* \\ &= e^{-r_{ij}}\left[\frac{\sigma_{ij}^2}{2}\left(1 + \operatorname{erf}\left(\frac{\hat{\delta}_{ij}}{\sqrt{2}\sigma_{ij}}\right)\right) - \frac{\hat{\delta}_{ij}\sigma_{ij}}{\sqrt{2\pi}}\exp\left(\frac{-\hat{\delta}_{ij}^2}{2\sigma_{ij}^2}\right)\right] \end{aligned} \quad (5)$$

where erf() is the error function. Substituting this into Equation 3 gives an easy to compute closed form expression for the expected loss.

3.4 Estimating P(M|D)

As discussed in Section 2, clickthrough data is best interpreted as relative relevance judgments. We can write them in the form $d_i \succ d_j$, indicating that d_i was judged more relevant that d_j. A standard approach to modelling noise in pairwise comparisons is to assume that the probability of an outcome is determined by the Bradley-Terry model [2]:

$$P(d_i \succ d_j) = \frac{rel(d_i)}{rel(d_i) + rel(d_j)}, \tag{6}$$

where $rel(d_i)$ is the relevance of d_i. The Bradley-Terry model can be reparameterized setting $rel(d_i) = 10^{\mu_i/\sigma}$ where σ is a known, global and fixed parameter. Assuming the pairwise judgments are independent (as can be reasonably expected with clickthrough data from multiple users),

$$\begin{aligned} P(\mathcal{D}|M = (\mu_1, \ldots, \mu_{|\mathcal{C}|})) &= \prod_{d_i \succ d_j \in \mathcal{D}} P(d_i \prec d_j | \mu_i, \mu_j) \\ &= \prod_{d_i \succ d_j \in \mathcal{D}} \frac{1}{1 + 10^{-(\mu_i - \mu_j)/\sigma}} \end{aligned}$$

Given this likelihood model and a Gaussian prior, we can apply an off-the-shelf algorithm to maintain $P(M|\mathcal{D})$, namely the glicko rating system commonly used for rating chess players [11]. Given an estimate of player ability (document relevance) ν_i and error in the estimate σ_i, this algorithm provides a set of approximate online update equations for maintaining the estimated relevance and error as data is collected. The update to the estimates for d_i, following a single comparison to d_j (where s_i is 1 if d_i wins and 0 otherwise) is presented in Table 1.

While it would also be interesting to compare alternative ways of maintaining $P(M|\mathcal{D})$ (e.g. [17]), or using a batch algorithm (see [19] for a discussion of alternatives), the simplicity and online aspects of the glicko system are appealing. In particular, in real world settings where large amounts of data are collected for large document collections with a large number of queries, a global optimization is likely to be slow and thus infeasible.

4. EXPLORATION STRATEGIES

As we have seen that users are much more likely to provide feedback on highly ranked documents, we turn to the question of optimizing the data collection process to most quickly minimize the loss. In particular, by selecting which documents to present at high rank, we influence the pairs of documents for which we obtain relevance judgments. In this paper, we will consider only modifications that change two documents in a ranking, limiting ourselves to the top two most of the time. We will see that despite the simplicity of this approach, substantial improvements in performance can be obtained at small cost in presented ranking quality.

We consider the following algorithms for determining which ranking to present users.

Passive Collection (Top2). Present the mode ranking, i.e. sorting documents by $\hat{M} = (\nu_1, \ldots, \nu_{|\mathcal{C}|})$.

The algorithm Top2 assumes no changes are made to the mode ranking, ignoring bias in data collection. This is the approach used in all previous work in learning to rank, and would be effective if users provided feedback about results throughout the ranking. In some settings this may be the case, for example in search engines for academic articles where many users thoroughly consider all retrieved results. However in general web search settings, as discussed above, users focus their attention on the highest ranked results.

$$\nu_i \leftarrow \nu_i + \frac{q}{\frac{1}{\sigma_i^2} + \frac{1}{\delta^2}} g(\sigma_j^2)(s_i - E(s|\nu_i, \nu_j, \sigma_j^2)) \tag{7}$$

$$\sigma_i^2 \leftarrow \left(\frac{1}{\sigma_i^2} + \frac{1}{\delta^2}\right)^{-1} \tag{8}$$

where

$$\begin{aligned} q &= \frac{\log 10}{400} \\ g(\sigma^2) &= \frac{1}{\sqrt{1 + 3q^2\sigma^2/\pi^2}} \\ E(s|\nu_i, \nu_j, \sigma_j^2) &= \frac{1}{1 + 10^{-g(\sigma_j^2)(\nu_i - \nu_j)/400}} \\ \delta^2 &= \frac{1}{q^2 g(\sigma_j^2)^2} \times \\ &\quad \frac{1}{E(s|\nu_i, \nu_j, \sigma_j^2)(1 - E(s|\nu_i, \nu_j, \sigma_j^2))} \end{aligned}$$

Table 1: glicko update equations

Random Exploration (Random). Select a random pair of documents and present them first and second. Then rank the remaining documents according to $\hat{M}$.

This algorithm is a naive modification to the presented ranking. Two random documents are picked uniformly and inserted at the top of the ranking presented to users. Given the uniform distribution, this perturbation is likely to often pick documents that have a low prior expectation of being relevant, thus likely presents users with poorer results. However, it benefits from the potential for feedback on all documents regardless of rank, even in the presence of significant user bias. A similar method was proposed by Pandey et al. [23] in the context of identifying new web pages that would soon become popular, suggesting to randomly insert new documents into web search results.

Largest Expected Loss Pair (LELpair). Select the pair of documents d_i and d_j that have the largest pairwise expected loss contribution, and present these first and second. Rank the remaining documents according to $\hat{M}$. Formally, this means we select the pair d_i and d_j that satisfies:

$$\underset{i \neq j}{\operatorname{argmax}}\, E_{P(M^*|\mathcal{D})}[\mathcal{L}^{pair}(\hat{M}, M^*, i, j)]$$

LELpair selects the pair of documents with largest pairwise contribution to the expected loss out of all pairs of documents. By presenting these documents at a high rank, the feedback given on them will reduce the uncertainty in the relative relevance of the documents. This will, in the long run, drive the expected loss contribution of the pair of documents down. Given the glicko update rules, the pairwise contribution of all other pairs of documents will not increase. Hence this method will eventually drive the expected loss down. Additionally, due to the exponential decay in the loss of misordered pairs as the rank increases, LELpair tends to select pairs of documents where at least one has a high estimated relevance. Lower ranked documents are also eventually selected, but only after high rank documents have been evaluted and their expected loss contribution is reduced. If we ignore the effect of rank in the loss function, this approach is similar to previous work in active learning where users are asked to label items where the predicted label is most uncertain (e.g. [3, 27]). In our setting, document pairs with high pairwise contribution tend to be those with large estimated error in relevance.

One Step Lookahead (OSL). For each pair of documents, compute the expected pairwise loss and the expected pairwise loss after a comparison based on the Bradley-Terry model (using $\hat{M}$ to estimate the probability of possible outcomes) and glicko updates. Select the pair of documents with the largest expected reduction in the pairwise loss and present these first and second. Rank the remaining documents according to $\hat{M}$. Formally, if $\hat{M}'$ is the mode of $P(M|\mathcal{D})$ after updating it given the outcome of a comparison of d_i and d_j, we select the pair d_i and d_j that satisfies:

$$\underset{i \neq j}{\operatorname{argmax}} \Big[E_{P(M^*|\mathcal{D})}[\mathcal{L}^{pair}(\hat{M}, M^*, i, j)] - E_{\hat{M}'}\Big[E_{P(M^*|\mathcal{D})}[\mathcal{L}^{pair}(\hat{M}', M^*, i, j)]\Big]\Big]$$

Intuitively, this algorithm performs approximate gradient descent on the loss function. OSL finds the pair whose contribution to the expected loss is likely to decrease most following a pairwise comparison. The expected contribution of the pair after a comparison is a weighted sum of the expected loss contribution for the two possible outcomes (either d_i wins the comparison or d_j wins). In this computation, the effect of possible rank changes is ignored for efficiency reasons. This method is also related to an approach proposed by Chajewska et al. in the context of utility estimation where they found that the true utility of many different outcomes can be quickly discovered by maximizing the reduction in expected loss given new data [6].

Largest Expected Loss Documents (LELdoc). For each document d_i, compute the total contribution of all pairs including d_i to the expected loss of the ranking. Present the two documents with highest total contributions first and second, and rank the remainder according to $\hat{M}$. Formally, this method selects the pair d_i and d_j that satisfies:

$$\underset{i \neq j}{\operatorname{argmax}} \Big[\sum_{a \neq i} E_{P(M^*|\mathcal{D})}[\mathcal{L}^{pair}(\hat{M}, M^*, i, a)] + \sum_{a \neq j} E_{P(M^*|\mathcal{D})}[\mathcal{L}^{pair}(\hat{M}, M^*, j, a)]\Big]$$

This method addresses a potential limitation of LELpair and OSL: They only consider individual pairwise document contributions to the expected loss despite the contributions of pairs not being independent. LELdoc addresses this by computing the total contribution of each document by summing over all pairs including that document. For example, if some document d is ranked third, it's total contribution is the risk from d and the top ranked document, plus the contribution from d and the second document, plus the risk from d and the fourth document and so forth. LELdoc selects the two documents with highest total contributions and presents them first and second. By comparing these two documents and reducing the uncertainty in their relevances, we are likely to reduce the contributions to the risk of all pairs including the documents.

An alternative selection algorithm proposed in previous work (e.g. [13, 7]) is to compare pairs of items such that the probability distribution over models changes most in terms of KL-divergence or entropy. We do not pursue this alternative as it does not take into account the loss function being optimized.

Finally, we note that explorations strategies for rankings are related to the opponent assignment problem in sports tournaments. However, there are two key differences. First, a tournament has a different concept of loss. A criterion often optimized is the probability of the true best player winning the final game (e.g.[12, 26]). Second, pairwise comparisons in a tournament have no cost. In most sports, a common constraint is that all teams or players must compete for at least n rounds. This means that each "item" must be compared with some other item every round and the optimization problem is to select which pairs are compared such that the loss is eventually minimized rather than aiming to minimize the loss as quickly as possible.

5. EVALUATION METHODOLOGY

We now have a number of strategies for eliciting useful training data from users of a search system, and have a method to estimate the relevance of the documents using our probabilistic model. In this section, we will describe how these strategies were evaluated. In particular, we will compare how effective each strategy is at improving the quality of the rankings shown to users.

We evaluate as follows: Given an initial ranking of one thousand documents as returned by a search engine in response to a query, we derive a prior $P(M)$. This prior initializes $P(M|\mathcal{D})$. For a particular exploration strategy, we next select a ranking to present to users. We evaluate the loss of the presented ranking and of the mode ranking derived from $\hat{M}$. Next, we simulate user behavior on the presented ranking, using a simple behavioral model, and collect training data that is used to update the model parameters. We repeat this process 3,000 times for each initial ranking. This experimental setup is formalized in Algorithm 1.

The behavioral model we use to simulate clickthrough data is detailed in Algorithm 2. By using a simulation, it is possible to evaluate the exploration strategies in detail without needing large numbers of test subjects, and avoid effects that may be unique to specific users (e.g. to academic users). Our model simplifies real behavior by assuming that users only click on top two results, and do so with probability specified by the Bradley-Terry model. This is motivated by the fast decay observed in the number of clicks as rank increases in real search systems. Clearly, in a real setting some additional data would be collected from lower ranks, making the results we report conservative in this respect. However, the amount of data collected about results at lower ranks would be significantly smaller.

We repeated each experiment with either 30 or 100 initial rankings, each giving a different initial set of relevance estimates. We report the mean loss across all runs (normalized such that the initial loss is 1), or the mean average precision (MAP). In some results we present a single final

Algorithm 1 Evaluation Setup

```
1: Input: Estimated relevances $\{\nu_i\}$ for $d_i \in \mathcal{C}$
2: $\sigma_i \leftarrow \sigma_0$ for $d_i \in \mathcal{C}$
3: for iteration 1 through 3,000 do
4:     Pick two documents $d_i$, $d_j$ to rank 1st and 2nd
5:     Randomly swap $d_i$ and $d_j$ (see Section 2)
6:     Show the selected ranking to user
7:     Record training data given user feedback
8:     Update $\nu_i$, $\nu_j$, $\sigma_i$, $\sigma_j$ per Equations 7 and 8
9: end for
```

Algorithm 2 User Behavioral Model

```
1: Input: Ranking of documents $(d_1, \ldots, d_{|\mathcal{C}|})$
2: Input: True relevances of documents $(\mu_1^*, \ldots, \mu_{|\mathcal{C}|}^*)$
3: if $UniformRandom(0,1) < \frac{1}{1+10^{-(\mu_1^*-\mu_2^*)/400}}$ then
4:     Winner is $d_1$: $s_1 \leftarrow 1$; $s_2 \leftarrow 0$
5: else
6:     Winner is $d_2$: $s_1 \leftarrow 0$; $s_2 \leftarrow 1$
7: end if
```

performance, i.e. the loss or MAP after 3,000 pairwise comparisons and model updates. Note that as our rankings are of 1,000 documents, 3,000 comparisons is on average just six noisy pairwise comparisons involving each document.

6. RESULTS

We start by evaluating the exploration strategies on synthetic data, where we evaluate their effectiveness if the assumed prior distribution matches the true data generating model. This will be followed by an evaluation using TREC-10 data.

6.1 Synthetic Data

We randomly generated a corpus of 1000 documents with expected relevances μ_i^* drawn from $\mathcal{N}(1500, 147^2)$. We chose this scale as it is comparable to typical chess scores. We then drew ten independent initial models, drawing ν_i from $\mathcal{N}(\mu_i^*, 147)$ and initializing $\sigma_i = 147 \equiv \sigma_0$. We repeated this process three times, giving 30 initial rankings over three different random corpora.

For each initial ranking, we ran each strategy for 3,000 iterations. Figure 1 shows the loss of the mode ranking at each iteration. Along the horizontal axis is the number of pairwise comparisons. After each pairwise comparison, $P(M|\mathcal{D})$ is updated and a new pair to compare is selected. The vertical axis is the average loss of the mode ranking relative to the initial average loss. The error bars indicate one standard error in the mean scaled loss.

Which exploration strategy learns fastest?

The first question to answer is which strategy learns fastest. The passive Top2 approach does not lead to a meaningful overall reduction in the loss. This is because our user model assumes that users only provide feedback on the top two documents. After a few comparisons, the top few documents have their relative position correctly established and no new documents are ever compared again. Our other baseline algorithm, Random, sees the loss decrease slowly.

We see that the other exploration strategies all perform substantially better than the baselines. Both LELpair and OSL quickly reduce the total loss by selecting pairs of documents with high contributions to the expected loss, and high expected reductions in it. We see this improvement continues for a large number of comparisons. In contrast, LELdoc appears to asymptote more quickly. This is because the documents selected continue to be those at high ranks even after many comparisons. In effect, LELdoc is too biased toward highly ranked documents. Comparing with LELpair and OSL, we see that while lower ranked documents may have lower total contribution to expected loss, they often have higher inidividual pairwise contributions.

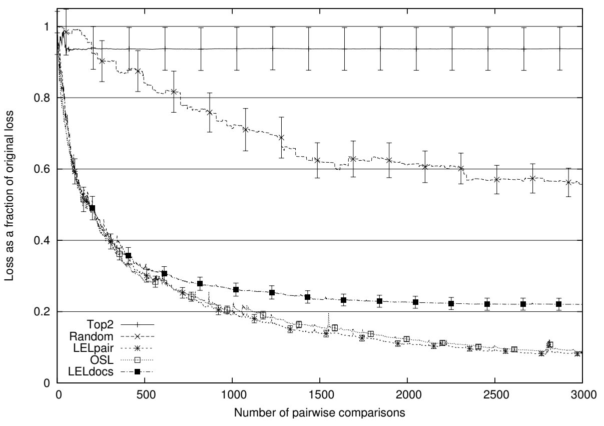


Figure 1: Change in loss as a function of the number of pairwise comparisons for each exploration strategy on synthetic data

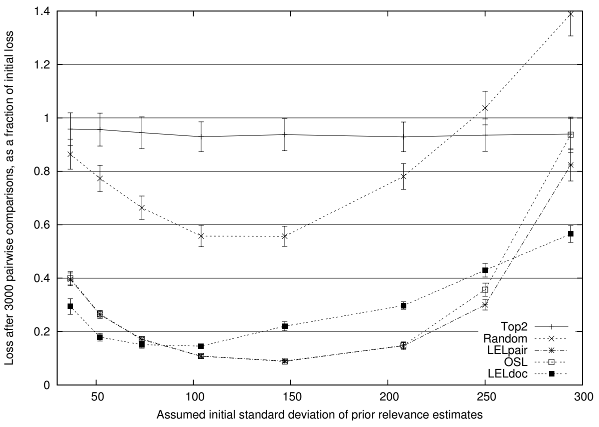


Figure 2: Effect of weight of prior on the final loss evaluated on synthetic data (true noise is $\sigma_0 = 147$)

How robust is the approach to prior assumptions?

The second natural question to ask is how robust the results are to the weight given to the initial ranking, as in the case of real data the correct weight is likely to be unknown. This weight is encoded by the initial values of σ_i, i.e. σ_0. We now explore the effect of changing that value. This experiment is possible because we know the level of noise used when generating synthetic data.

Figure 2 shows the effect of selecting an assumed noise level that differs from the true noise level. With our default setting, and that used to generate our synthetic datasets, the probability of a document in the top 10 according to the prior not being in the top 100 according to true relevance is about 8%. We can see that selecting a suboptimal σ_0 does not drastically reduce performance after a fixed number of pairwise comparisons. Apart from LELdoc, the best performance is achieved when the actual noise is correctly known. Interestingly, LELdoc performs best when the error in the prior is underestimated as it then selects documents further down the ranking for comparison.

6.2 TREC Data

In addition to the fully synthetic data, we also evaluated the exploration strategies using the TREC-10 web track queries (topics 501 through 550) in the WT10g document corpus. This subset of the corpus includes 50 topics and topic descriptions, run as queries on documents that are part of the corpus. As part of the 10th Text REtrieval Conference (TREC-10) [16], 18 teams submitted a ranking of documents for each topic. Then, for each topic, documents ranked highly by the teams were manually judged to be either highly relevant (relevance score of 2), relevant (score of 1) or non-relevant (score of 0). All other documents in the corpus were assumed non-relevant (score of 0). The discretized nature of these relevance judgments is unrealistic, as few documents are likely to have precisely the same relevance in the real world. To compensate and make the learning problem more realistic, we added uniform random noise in the range $[-0.5, 0.5]$ to the true relevance judgments, preserving the relative order of highly relevant, relevant and non-relevant documents.

For each of the 50 TREC-10 topics, we randomly selected two submissions and used the submitted scores to initialize our model. We then repeated the evaluation described for synthetic data. Each submission includes a ranking of typically 1000 documents, with a score given to each document. The scores are arbitrary and unnormalized. Clearly they can be interpreted as a prior in any number of ways. As each ranking typically contains both highly relevant documents and non-relevant documents, we chose to normalize the scores to a linear interval $[1500 + \sigma_0, 1500 - \sigma_0]$ with $\sigma_0 = 147$. The resulting scores were used to set the initial ν_i. The initial estimated error σ_i were set to σ_0. This means that a document with a score near the maximum score is estimated to have approximately a 30% percent chance of in fact being in the lower half of the ranking.

Which exploration strategy learns fastest?

Figure 3 shows the performance of the exploration strategies. We see that LELpair, OSL and LELdoc improve the loss significantly and rapidly. We also see that Top2 performs as it did on the synthetic data. One the other hand, Random performs differently: The loss of the mode ranking initially increases, then improves slightly but remains high. We believe that this is due to the mismatch between the prior and model. When two documents are compared, if the outcome is not the expected one (e.g. due to noise) then the update to the relevance estimates can be large. Sometimes, the lower ranked document is projected to a much higher rank, and the loss does not quickly recover due to few pairwise comparisons per document. Interestingly, performance of Random depends on the quality of the initial ranking. When the initial loss is high, after 3,000 pairwise comparisons the loss tends to be reduced. The opposite is true when the algorithm starts with a very good ranking.

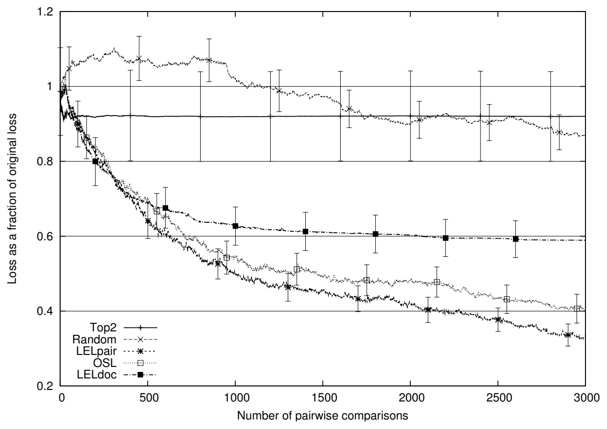


Figure 3: Change in loss as a function of the number of pairwise comparisons for each selection algorithm on TREC-10 data

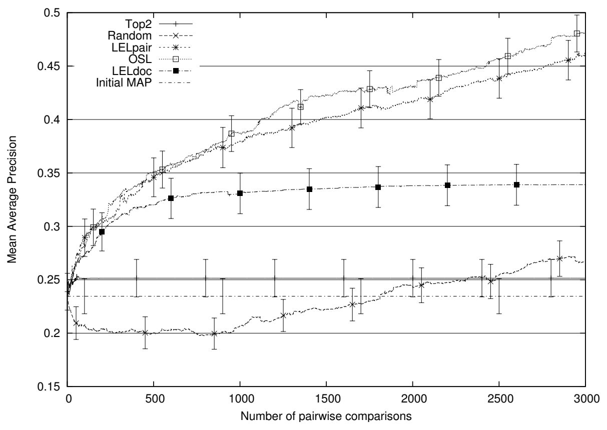


Figure 4: Change in MAP score as a function of the number of pairwise comparisons for each selection algorithm on TREC-10 data

How do the strategies perform with respect to MAP?

The loss function presented earlier is not one commonly used to evaluate rankings. A measure much more widely used by the research community is the mean average precision (MAP). To compute the MAP, we consider each document with true relevance μ_i^* above some threshold as relevant and others as irrelevant. The average precision of a ranking is the average of the precision measured at each relevant document[1]. The MAP score is the mean of the average precisions across all 100 experiments. We used a threshold of 0.5 scaled in the same way the scores were scaled.

Figure 4 shows how the MAP of the ranking changes as more pairwise comparions are performed. MAP visibly behaves very similarly to our loss function. We see that LELpair and OSL result in the largest improvement in MAP, and appear likely to continue to improve further with more pairwise comparisons. As before, LELdoc performance plateaus after a small number of pairwise comparisons and Top2 sees almost no improvement. Random performs poorly, with an

[1]For simplicity, we only consider the 1000 ranked documents when computing the MAP score. While the corpus may contain relevant documents that never made it into the top 1000, these do not contribute to the MAP score.

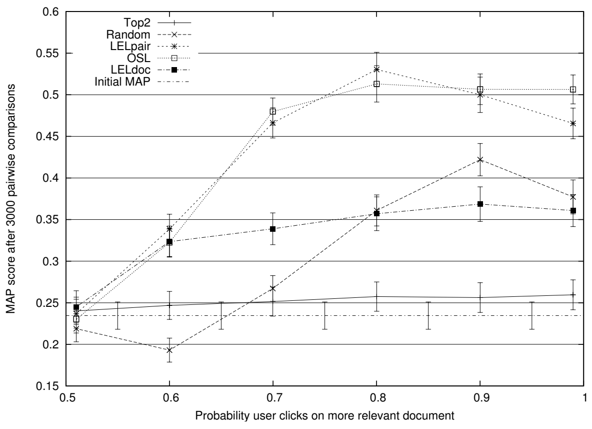


Figure 5: Effect of different noise levels in pairwise preferences on final MAP score, evaluated on TREC-10 data.

initial drop in MAP although after 2,000 pairwise comparisons the MAP is above baseline.

How does noise influence learning speed?

So far, our simulation has assumed a particular amount of noise in user clicks. Given two documents that differ in true relevance by one TREC relevance level, we have assumed that the user will click on the more relevant one 70% of the time. While this level of noise is realistic [21], it is of particular interest to observe what would happen to the difficulty of the learning task if the noise in pairwise preferences were different. We modified our user model to change the level of noise, and changed the glicko update equations equivalently. Figure 5 shows the final MAP score after 3,000 pairwise comparisons as the level of noise changes. First, we see that irrespective of the noise level, the best pair exploration strategies remain LELpair and OSL. On the left of the figure we see that if there is a lot of noise in the pairwise preferences, all the algorithms perform more poorly. This is to be expected, given that the amount of information contained in 3,000 pairwise preferences decreases as the amount of noise in user clicks increases. On the other end of the scale, we see that even if users select the more relevant document almost 100% of the time, the final MAP score decreases somewhat. This is a side-effect of the normal approximation implicit in the glicko updates [11].

How robust is this method to noise assumptions?

The above results assume the noise level is known in advance. Figure 6 shows the effect of a mismatch between the assumed noise level (used to scale the initial scores, and in the glicko updates) and the true level of noise in pairwise preferences (in the user model). Along the horizontal axis we have the probability a user selects the correct document in a pair as more relevant, if the true relevances of the pair differ by one TREC relevance level. The figure shows how the MAP of the mode ranking after 3,000 pairwise comparisons is affected by different estimates of the noise in pairwise clicks. Each line corresponds to a different assumed noise level. The figure shows that if the amount of noise is underestimated, performance is poorer although not drastically so (unless the noise is underestimated substantially). On the

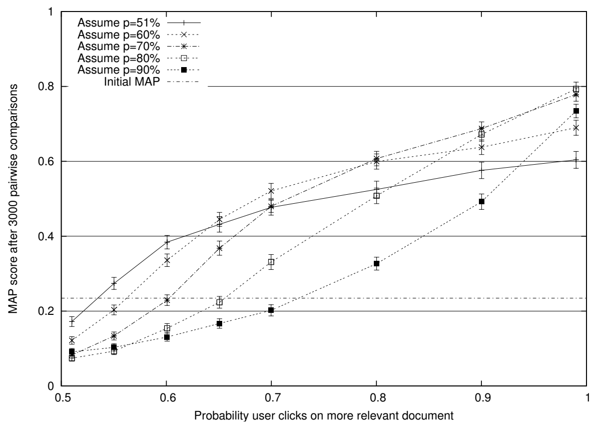


Figure 6: Effect of incorrect assumptions about the noise level in relevance judgments on final MAP score, evaluated on TREC-10 data using OSL.

Loss Function	MAP Score
$e^{-r_{ij}}\ \left((\mu_i^* - \mu_j^*) - (\nu_i - \nu_j)\right)^2 \mathbf{1}_{misordered}$	0.481 ± 0.017
$\left((\mu_i^* - \mu_j^*) - (\nu_i - \nu_j)\right)^2 \mathbf{1}_{misordered}$	0.281 ± 0.017
$e^{-r_{ij}}\ \left((\mu_i^* - \mu_j^*) - (\nu_i - \nu_j)\right)^2$	0.287 ± 0.012
$e^{-r_{ij}}\ \mathbf{1}_{misordered}$	0.337 ± 0.020

Table 2: Mean Average Precision after 3000 iterations of optimizng different loss variants using OSL.

other hand, we see that if the pairwise preferences are less noisy than assumed, the final performance does not suffer.

Of particular interest, these results tell us that the best strategy is to assume the level of noise conservatively in order to see the best performance improvements. Finally, it is worth noting that in the case of extremely noisy clicks it may be beneficial to aggregate user clicks, i.e. collect a number of pairwise judgments for each pair of documents, then count this as a single pairwise judgment for the document with the most preference votes. This would reduce the effective level of noise in pairwise judgments at the expense of necessitating more data.

Which loss functions are a good proxy for MAP?

So far, our experimental results show that minimizing the expected loss also improves the MAP. We now demonstrate that the loss function presented in Section 2 leads to particularly good MAP performance. Table 2 shows the MAP performance after 3,000 pairwise comparisons if we optimize OSL to different variants of the loss function presented, by ignoring individual properties suggested in Section 3.3.

We see that using a loss function without exponential decay, without a distance penalty or without a hinge leads to substantial and significant reductions in the final MAP scores. In particular, optimizing a loss function that simply depends on document ranks, rather than on the actual relevance estimates (the fourth line of the table) leads to poorer MAP performance. This shows that despite MAP only being sensitive to document order, minimizing error in relevance estimates leads to better MAP performance.

6.3 Controlling for Presentation Loss

The figures in the previous two sections show the loss and

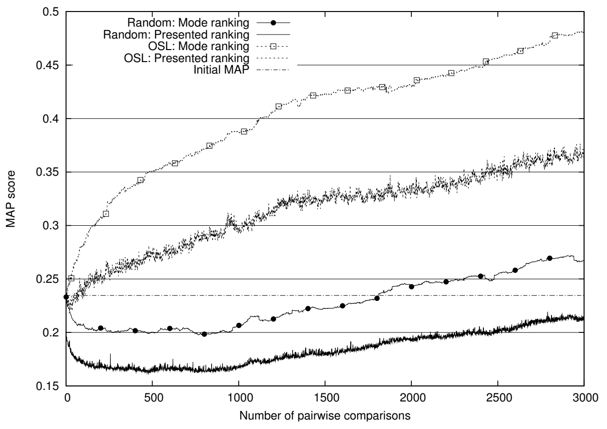


Figure 7: MAP scores of the mode rankings, and the presented rankings as a function of number of pairwise comparisons for OSL and Random.

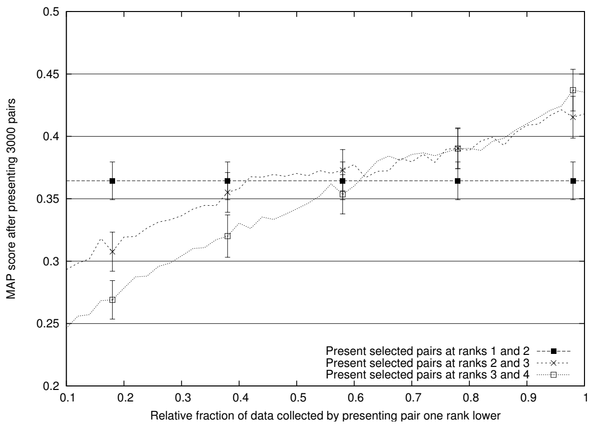


Figure 8: MAP scores of the presented ranking after 3000 pairwise comparisons assuming pairs presented at different ranks.

MAP of the mode ranking as pairwise preference data is collected. However, to collect the data, the ranking of documents that is presented to users is not the mode ranking. Rather, users see rankings with a pair of results inserted at the top. This means that usually the presented ranking has a higher loss and a lower MAP score than the mode ranking. The difference between the MAP of the presented and mode rankings is shown in Figure 7. The top two lines are for OSL. The top line shows the MAP of the mode ranking. The second line shows the MAP of the ranking shown to users. We see that while the presented ranking is worse than the mode ranking, it is almost immediately above the initial MAP, and improves quickly as data is collected. The separation between the two rankings increases because as the relevance of higher ranked documents is established, lower ranked documents are shown more often. We see a similar effect comparing the presented and mode rankings of the Random strategy in the lower two lines, although the presented ranking does not have a higher MAP than the initial ranking for all 3,000 pairwise comparisons. We also note that the MAP of the presented ranking using OSL is substantially better than the mode ranking when the Random exploration strategy is used.

An interesting final experiment is to consider the tradeoff between the quality of the presented ranking and the quality of the mode ranking. One possibility to reduce the impact of data collection would be to presenting selected pairs at lower ranks instead of at the top two positions. With real users, this would lead to a reduction in the amount of data collected, but may improve the MAP of the presented ranking by reducing the performance gap. Figure 8 shows how the MAP score of the presented ranking after 3,000 pairwise comparisons would change if the selected pair was presented at a lower rank. The reduction in the amount of data collected as a function of rank is shown along the horizontal axis. For example, if for some user population moving the selected pair down by one rank reduces the number of pairwise preferences collected by 60%, the correct point for an evaluation would be 0.4 on the horizontal axis. This would mean that by presenting a pair at rank 2 and 3 rather than 1 and 2, we would only receive 40% as many clicks. Presenting selected pairs at ranks 3 and 4 would receive 16%

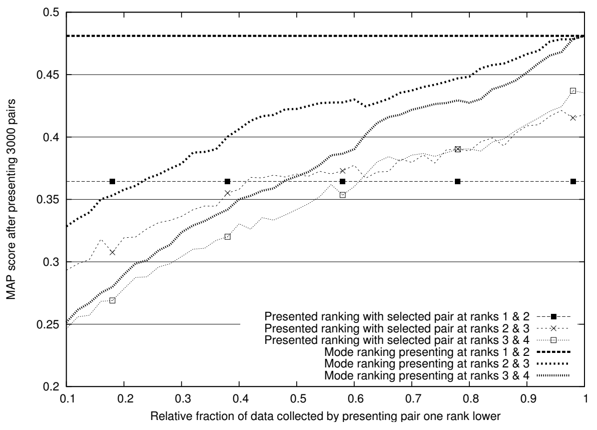


Figure 9: MAP scores of the mode and presented rankings after 3000 pairwise comparions.

as many clicks. At 0.4, we see that due to the reduction in the amount of data collected, the MAP of the *presented* ranking would be lower if the selected pairs were at ranks 3 and 4 rather than at ranks 1 and 2.

Taking the same lines as in Figure 8 and overlaying the MAP of the mode rankings for all three ranks, we get Figure 9. It shows that the separation between the mode ranking and the presented ranking decreases as the rank of the selected pair is decreased. However, the MAP of the mode ranking also decreases, especially if much less data is collected when pairs are presented at lower ranks.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we have formalized the ranked relevance elicitation problem, and demonstrated that by using active exploration the quality of a ranking can be improved faster than by collecting pairwise training data passively or naively. We presented a number of strategies to minimize the expected loss and showed that two in particular perform well. Our experiments showed a significant level of robustness to noise in the clickthrough data and to prior assumptions, and demonstrated how presentation loss and quality of learned ranking can be traded off.

A natural extension of this work would be to find a global optimization approach for this problem so as to derive a provably optimal algorithm to collect training data using a regret minimization approach. There has also recently been interest in minimizing the number of documents that need to be evaluated by human judges to evaluate performance of ranking algorithms on a document collection (e.g. [5, 28]). While such human judgments are absolute, i.e. a human is asked to rate each document on a fixed scale, it would be interesting to adapt our approach to a setting that minimizes the number of evaluations necessary with an appropriate prior and loss function.

8. ACKNOWLEDGMENTS

The second author was partly supported by a Microsoft Ph.D. Student Fellowship. This work was also supported by NSF Career Award No. 0237381 and a gift from Google.

9. REFERENCES

[1] E. Agichtein, E. Brill, S. Dumais, and R. Ragno. Learning user interaction models for predicting web search result preferences. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

[2] R. A. Bradley and M. E. Terry. The rank analysis of incomplete block designs. 1. the method of paired comparisons. *Biometrika*, 39:324–345, 1952.

[3] K. Brinker. Active learning of label ranking functions. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2004.

[4] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2005.

[5] B. Carterette, J. Allan, and R. Sitaraman. Minimal test collections for retrieval evaluation. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

[6] U. Chajewska, D. Koller, and R. Parr. Making rational decisions using adaptive utility elicitation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2000.

[7] W. Chu and Z. Ghahramani. Extensions of gaussian processes for ranking: Semi-supervised and active learning. In *Proceedings of the NIPS 2005 Workshop on Learning to Rank*, 2005.

[8] W. Chu and Z. Ghahramani. Preference learning with gaussian processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2005.

[9] O. Dekel, C. D. Manning, and Y. Singer. Log-linear models for label ranking. In *Proceedings of the Internation Conference on Advances in Neural Information Processing Systems (NIPS)*, 2003.

[10] N. Fuhr. Optimum polynomial retrieval functions based on the probability ranking principle. *ACM Transactions on Information Science (TOIS)*, 7(3):183–204, 1989.

[11] M. E. Glickman. Parameter estimation in large dynamic paired comparison experiments. *Applied Statistics*, 48:377–394, 1999.

[12] M. E. Glickman. Bayesian optimal design of knockout tournaments, 2006. Under Review.

[13] M. E. Glickman and S. T. Jensen. Adaptive paired comparison design. *Journal of Statistical Planning and Inference*, 127:279–293, 2005.

[14] L. Granka. Eye tracking analysis of user behaviors in online search. Master's thesis, Cornell University, 2004.

[15] L. Granka, T. Joachims, and G. Gay. Eye-tracking analysis of user behavior in WWW search. In *Poster Abstract, Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2004.

[16] D. Hawking and N. Craswell. Overview of the TREC-2001 web track. Nov 2001.

[17] R. Herbrich and T. Graepel. Trueskill^{TM}: A bayesian skill rating system. Technical Report MSR-TR-2006-80, Microsoft Research, 2006.

[18] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In A. S. et al., editor, *Advances in Large Margin Classifiers*, pages 115–132, 2000.

[19] D. R. Hunter. MM algorithms for generalized bradley-terry models. *The Annals of Statistics*, 32(1):384–406, 2004.

[20] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.

[21] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)*, 25(2), April 2007.

[22] R. Lin, T. A. Louis, S. M. Paddock, and G. Ridgeway. Loss function based ranking in two-stage hierarchical models. *Bayesian Analysis*, 1(4):915–946, 2006.

[23] S. Pandey, S. Roy, C. Olston, J. Cho, and S. Chakrabarti. Shuffling a stacked deck: The case for partially randomized ranking of search engine results. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 2005.

[24] F. Radlinski and T. Joachims. Query chains: Learning to rank from implicit feedback. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2005.

[25] F. Radlinski and T. Joachims. Minimally invasive randomization for collecting unbiased preferences from clickthrough logs. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2006.

[26] D. Ryvkin. The predictive power of noisy elimination tournaments. Under review.

[27] M. Saar-Tsechansky and F. Provost. Active sampling for class probability estimation and ranking. *Mach. Learn.*, 54(2):153–178, 2004.

[28] L. Torrey. An active learning approach to efficiently ranking retrieval engines. Technical Report TR2003-449, Computer Science Dept., Dartmouth College, 2003.