Post-Rank Reordering: Resolving Preference Misalignments between Search Engines and End Users *

Chao Liu
Microsoft Research
Redmond, WA 98052
chaoliu@microsoft.com

Mei Li
Microsoft Bing Search
Redmond, WA 98052
meili@microsoft.com

Yi-Min Wang
Microsoft Research
Redmond, WA 98052
ymwang@microsoft.com

ABSTRACT

No search engine is perfect. A typical type of imperfection is the preference misalignment between search engines and end users, *e.g.*, from time to time, web users skip higher-ranked documents and click on lower-ranked ones. Although search engines have been aggressively incorporating click-through data in their ranking, it is hard to eliminate such misalignments across millions of queries. Therefore, we, in this paper, propose to accompany a search engine with an "always-on" component that reorders documents on a per-query basis, based on user click patterns. Because of positional bias and dependencies between clicks, we show that a simple sort based on click counts (and its variants), albeit intuitive and useful, is not precise enough.

In this paper, we put forward a principled approach to reordering documents by leveraging existing click models. Specifically, we compute the preference probability that a lower-ranked document is preferred to a higher-ranked one from the Click Chain Model (CCM), and propose to swap the two documents if the probability is sufficiently high. Because CCM models positional bias and dependencies between clicks, this method readily accounts for many twisted heuristics that have to be manually encoded in sort-based approaches. For this approach to be practical, we further devise two approximation schemes that make online computation of the preference probability feasible. We carried out a set of experiments based on real-world data from a major search engine, and the result clearly demonstrates the effectiveness of the proposed approach.

Categories and Subject Descriptors

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

*Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the employers and funding agencies.

CIKM'09, November 2–6, 2009, Hong Kong, China.

General Terms

Algorithms, Experimentation

Keywords

Web search, Result reordering, Preference models

1. INTRODUCTION

Web search has become indispensable from everyday life, but it is yet perfect. Although commercial search engines have been leveraging numerous signals (including user click-through data) in document ranking, search users still need to skip top-ranked ones and click on lower ones from time to time. Click patterns like this suggest preference misalignments between the search engine and end users. In general, such misalignments are inevitable, because it is hard, if not totally impossible, for a ranker to consolidate users' preferences across millions of queries into a single ranker. We therefore need an automated mechanism to resolve misalignments on a per-query basis, based on users' clickthrough data.

A quick solution is to sort the top-10 documents in descending order of the number of clicks each document receives in the past. This solution assumes that more clicks indicate higher relevance, which could be correct for some queries (*e.g.*, navigational) but may not be so in general because of the following reasons:

- **Positional and trust bias**: Documents getting more clicks could be simply because they are ranked higher and hence more visible to the users (*i.e.*, positional bias). To complicate matters even further, some clicks on top-ranked documents could be purely from users' trust in the search engine (*i.e.*, trust bias). Therefore, not all clicks are equal.
- **Dependencies between clicks**: Clicks are correlated. For example, if the current clicked document is very relevant, the user could be less motivated to click the next even if it looks similarly relevant.
- **Insufficient and dispersed clicks**: Because of the heavy tail in search, not all queries can accumulate enough clicks even over a long time. Moreover, clicks could be highly dispersed across many documents because most queries (*e.g.*, informational) do not have a single authoritative destination.

Therefore, in order to sort by clicks, one has to (1) properly interpret clicks by accounting for click biases and disentangling dependencies, and (2) craft a set of rules to determine

when and what documents to sort. As one can imagine, the set of rules will likely involve many correlated quantities, *e.g.*, the absolute number of clicks, clickthrough rate, positions, and their corresponding thresholds. Instead of hacking heuristics and tuning parameters, we ask the question: *Is there a principled approach to resolving misalignments?*

Inspired by statistical hypothesis testing, we propose to *respect the original ranking unless there is strong click evidence against it.* Specifically, we assumes that a search engine puts d_i before d_j because it prefers d_i to d_j. We take this preference, denoted by $d_i \succ d_j$, as a null hypothesis $\mathcal{H}_0$, and check whether user clicks suggest otherwise; if the null hypothesis is disproved with statistical significance, the two documents should be swapped. In this way, misalignments are resolved pair by pair, instead of through a global sorting.

Unfortunately, the inherent dependencies between clicks prevent many testing techniques from applying because they usually rely on the i.i.d. assumption. We therefore propose to compute the probability that d_j is preferred to d_i, denoted by $P(d_j \succ d_i)$, directly from clicks, and account for the dependencies in calculating the probability. If this probability is above a threshold θ, say 0.75, the two documents will be swapped. As will be seen, θ is the only parameter for preference probability-based methods, and it can be easily set either intuitively or through experiments.

Computing $P(d_j \succ d_i)$ is however non-trivial because it has to account for biases and dependencies. Previous research on click models (*e.g.*, [6, 9, 13]) considers biases and dependencies, and provides a point estimation r for the relevance of each document, but it remains unclear how to compute $P(d_j \succ d_i)$ from r_j and r_i.

We solve this problem by utilizing the Click Chain Model (CCM) (whose details are briefed in Section 2 and [12]). Different from previous models, CCM models the document relevance in a Bayesian way, so that the relevance estimate is a full posterior $p(r)$ with $r \in [0, 1]$, rather than a single number. From posteriors $p(r)$'s, the preference probability (PP) $P(d_j \succ d_i)$ can be analytically computed by integrating over the area with $r_j > r_i$ on the joint probability of $p(r_i, r_j)$. We show, through experiments based on real-world click data, that document swapping based on the computed PP significantly outperforms the sort-based methods (Section 5). Furthermore, for this approach to be practical, we devise two methods that compute PPs at query time. Also worth mentioning is that the two methods actually uncover how to relate r_i's to the preference probabilities for the CCM case.

In summary, this study makes the following contributions:

1. We propose post-rank reordering to resolve "misaligned" document pairs by checking if user clicks suggest otherwise. The proposed approach features a similar rationale as hypothesis testing, needs a single query-independent parameter, and outperforms sort-based methods that are intuitively appealing.

2. This study expands on the existing work on Click Chain Model [12], and presents the first principled approach to computing preference probabilities from clicks. Considering search is a business about learning preferences, we believe the computed preference probabilities will find their applications in many search-related areas, *e.g.*, augmenting human judgments and learning to rank.

3. Motivated by practical requirements, we discover how to relate point relevance estimation to pairwise preference probabilities, for the CCM case. Because of the correlation between CCM and other models on the point relevance estimation, this discovery actually makes it possible for non-Bayesian click models (*e.g.*, the ICM and DCM in [13]) to compute preference probabilities.

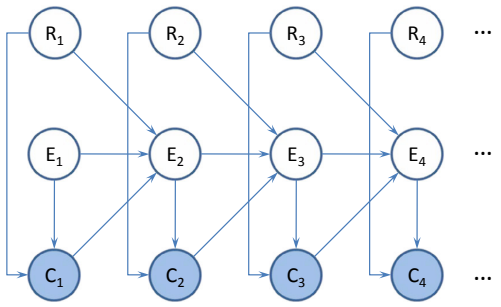


Figure 1: The graphical model representation of CCM. Shaded nodes are observed click variables.

The rest of the paper is organized as follows. Section 2 introduces preliminaries and re-caps the CCM. Section 3 elaborates on the computation of preference probabilities from CCM, and how to use them for document reordering. The two methods that compute preference probabilities at query time are discussed in Section 4. We report on the experimental evaluations in Section 5, and discuss related work in Section 6. Finally, Section 7 concludes this study.

2. PRELIMINARIES

We first introduce definitions and notations that will be used throughout the paper. A web search user initializes a *query session* by submitting a *query* to the search engine. Any re-submission or reformulation of the same query is regarded as another distinct query session. We use *document impression* to refer to the *web documents* (or URLs) presented in the first result page, and discard other elements in this page, *e.g.*, sponsored ads and related search. A document impression is represented by $D = \{d_1, \ldots, d_M\}$ (usually $M = 10$), where each d_i can be an index into a set of documents for the query. We say document d_i ranks higher than d_j if d_i appears before d_j, *i.e.*, $i < j$. All the following discussion is restricted to a given query unless otherwise noted.

Click models account for positional bias and dependencies between clicks by treating examination and clicks as probabilistic events that are correlated through document relevance. Specifically, for a particular query session, a binary random variables E_i is assigned to denote whether the snippet of d_i on the search result page is examined by the user, and C_i for the click event. For example, $P(E_i = 1)$ is the probability that position i is examined and $P(C_i = 1)$ is the corresponding probability for click. In the following, we brief the Click Chain Model. Readers interested in the technical details about CCM and its comparison to other models are referred to [12].

Figure 1 depicts the graphical model representation of CCM. A distinct feature of CCM from previous models [6, 9, 13] is its Bayesian modeling of the document relevance. In CCM, the relevance of d_i is modeled as a random variable

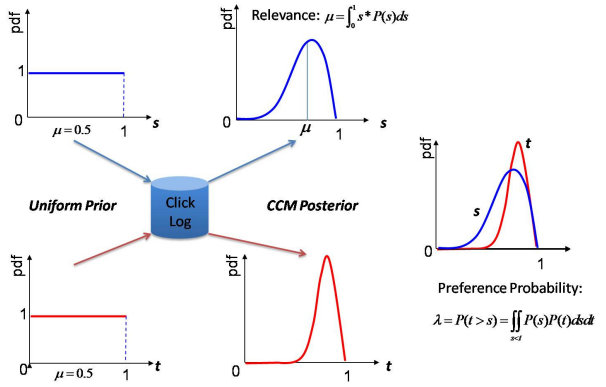


Figure 2: Illustration of Click Chain Model

$R_i \in [0,1]$, and its probabilistic dependencies with E_i's and C_i's are specified as below:

$$P(C_i = 1|E_i = 0) = 0 \quad (1)$$
$$P(C_1 = 1|E_i = 1, R_i) = R_i \quad (2)$$
$$P(E_{i+1} = 1|E_i = 0) = 0 \quad (3)$$
$$P(E_{i+1} = 1|E_i = 1, C_i = 0) = \alpha_1 \quad (4)$$
$$P(E_{i+1} = 1|E_i = 1, C_i = 1, R_i) = \alpha_2(1 - R_i) + \alpha_3 R_i, \quad (5)$$

where α's are user behavior parameters that can be automatically learnt from data.

Effectively, these dependencies describe a web user with following behaviors: the user starts the examination of the search result from the top. At each position i, the perceived relevance R_i determines how likely d_i is clicked (Eq. 2). Clicking or not, the user can continue the examination or abandon the current query session, and the probability to examine the next document d_{i+1} depends on her action at the current position i. Specifically, if she skips d_i ($C_i = 0$), this probability is α_1 (Eq. 4); on the other hand, if d_i is clicked, the probability further depends on the relevance R_i (Eq. 5). Intuitively, if d_i is perceived very relevant, the user will be less motivated to examine the next: parameters α_2 and α_3 correspond to the probability that d_i is extremely irrelevant and relevant, respectively.

Figure 2 illustrates what CCM does using click logs: before observing any log data, CCM assumes a uniform relevance prior for each query-URL pair; then upon observing click data, it adjusts the posterior according to Eqs. $1 \sim 5$ using Bayesian inference (details in [12]). Intuitively, if the URL is clicked, its posterior is pushed to the right and otherwise to the left if not clicked, for each of the search sessions in log. In general, the more data a query-URL pair has, the spikier the posterior would be; so for unfrequent query-URL pairs, the posterior could be relatively flat but still different from the uniform prior.

By exploiting the particular dependence structure of CCM, [12] derives the posterior for each document in closed form:

$$p(R_i|C^{1:U}) = N * R_i^{e_0} \prod_{m=1}^{2M+2} (1 - \beta_m R_i)^{e_m} \quad R_i \in [0,1], \quad (6)$$

where β_m's are constants that are determined by α's, $C^{1:U}$ represents the U search sessions for the given query, and N is the normalizer for the probability density function. Eq. 6 shows that the relevance posterior is fully characterized by the $(2M+3)$ integers $\{e_m\}_{m=0}^{2M+2}$, which can be obtained by a single scan of the click log and incrementally updated as new log data comes in. These properties, together with its proven effectiveness in modeling clicks and accounting for positional bias [12], make CCM a good choice for reordering documents in practice. For uncluttered notations, we denote $p(R_i|C^{1:U})$ by $p(R_i)$ in what follows.

As illustrated in Figure 2, we can take the distribution mean μ_i of $p(R_i)$ as the point estimation of the relevance for d_i, and compute the preference probabilities between two documents analytically, as detailed in the next section.

3. PREFERENCE PROBABILITY FOR DOCUMENT REORDERING

The idea of reordering documents is simple: for each document pair (d_i, d_j) with $i < j$, we compute the preference probability from d_j to d_i, as defined below, and if the probability is above the threshold θ, d_i and d_j are swapped.

DEFINITION 1 (PREFERENCE PROBABILITY (PP)). *Given a document pair (d_i, d_j), the preference probability from d_j to d_i, denoted by $P(d_j \succ d_i)$, is defined as*

$$P(d_j \succ d_i) = P(R_j > R_i),$$

where $R_i, R_j \in [0,1]$ are the random variable for the relevance of d_i and d_j respectively.

By assuming posterior independence, $P(d_j \succ d_i)$ can be computed as below:

$$\begin{aligned} P(d_j \succ d_i) &= P(R_j > R_i) \\ &= \int\int_{R_i<R_j} P(R_j, R_i) dR_j dR_i \\ &= \int\int_{R_i<R_j} p_j(R_j) p_i(R_i) dR_j dR_i \\ &= \int_0^1 p_j(R_j) \Big(\int_0^{R_j} p_i(R_i) dR_i \Big) dR_j \\ &= \int_0^1 p_j(R_j) \Phi_i(R_j) dR_j, \end{aligned} \quad (7)$$

where Φ_i is the cumulative distribution function (CDF) of R_i.

As can be seen here, there is no special treatment for infrequent queries in computing the preference probability. The large uncertainly of document relevance for infrequent queries has already been encoded in the relatively flat posterior, and hence no heuristics is needed to differentiate between frequent and infrequent query-URL pairs.

Although we have $p(R)$ in closed form, directly plugging Eq. 6 into Eq. 7 does not give a tractable solution, so we compute $P(d_j \succ d_j)$ through numeric integration. Specifically, we evaluate $p_j(R_j)$ and $\Phi_i(R_j)$ at B sample points $r_b = \{\frac{i+0.5}{B}\}_{i=0}^{B-1}$, and calculate $P(d_j \succ d_j)$ by

$$\frac{1}{B} \sum_{b=0}^{B-1} p_j(r_b) * \Phi_i(r_b). \quad (8)$$

The accuracy of Eq. 8 increases as B gets larger, and $B = 1000$ is sufficient for most cases. With a little misnomer, we call the value coming from Eq. 8 as the *exact PP*.

With preference probabilities computed, we now define θ-inconsistent document pairs as below.

Algorithm 1 PPSwap($\mathcal{L}$, θ)

Input: $\mathcal{L}$: The original ranking of top-M documents
θ: The inconsistency threshold
Output: $\mathcal{L}$: The re-ordered ranking based on PP

1: **for** $i = 1$ to (M-1)
2: **for** $j = M$ to $i + 1$
3: **if** $P(\mathcal{L}[j] \succ \mathcal{L}[j-1]) \geq \theta$
4: **Swap**$(\mathcal{L}[j], \mathcal{L}[j-1])$
5: **return** $\mathcal{L}$;

DEFINITION 2 (θ-INCONSISTENT PAIR). *A document pair (d_i, d_j) is a θ-inconsistent pair if $i < j$ and $P(d_j \succ d_i) \geq \theta$, where θ is a constant greater than 0.5, and $P(d_j \succ d_i)$ is called the inconsistency probability of the pair.*

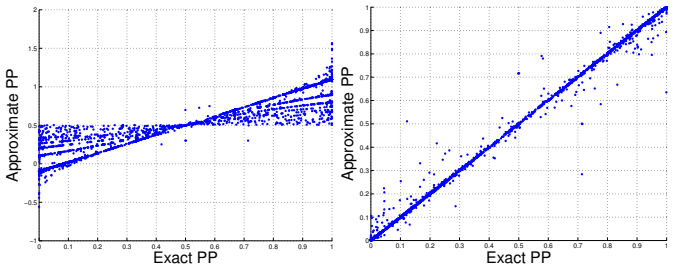
Because the preference probability from CCM does not constitute triads, document reordering is similar to a sort, whose comparison operator is replaced by checking if the inconsistency probability is above the threshold θ. This gives the algorithm listed in Algorithm 1. The algorithm is presented using a bubble sort template for its clarity, although any *comparison sort* template can be used. In practice, insertion sort could be a better choice because (1) it is very efficient on nearly-sorted list, which is exactly the case for web ranking and (2) it is ideal for sorting small list and perfectly fits the top-10 scenario.

Although Algorithm 1 seems to be the same as sorting, there are two major differences. First, the reordered list from Algorithm 1 depends on the original ranking whereas sort-based approaches are independent of the original. For example, any document pairs $(d_i, d_j)'s \quad i < j$ with $P(d_j \succ d_i) \leq \theta$ will not be swapped. Second, Algorithm 1 solely relies on the preference probability and there is no score explicitly associated with each document that can be used for sorting.

4. PREFERENCE PROBABILITY AT QUERY TIME

Post-rank reordering can be implemented in two ways in practice. The first is to pre-compute reordered rankings for interested queries using Algorithm 1, and serve the reordered rankings online through result caching [11]. In this way, the exact PP can be computed offline using Eq. 8. The second approach is to reorder documents at query time, which could help avoid some limitations of result caching in the first approach, *e.g.*, query coverage and query dynamics [3]. For the second approach, because it is impractical to pre-compute all PP's and load them online, we need efficient ways to compute preference probabilities at query time.

Eq. 8 suggests a naive approach: if the B sample points of the relevance posterior are registered with each document, the probabilities can be computed at query time with the same accuracy as offline. But the challenge is whether we can compute preference probability from much fewer registered values than B. In the following, we will describe two approximation methods that require only one value registered to each document. The first method is through regression (Section 4.1) and the other is by parameterizing the Bradley-Terry model (Sections 4.2). Finally, Section 4.3 reveals that the two methods are roughly equivalent.



(a) Regression with μ (b) Regression with $f(\mu)$

Figure 3: Goodness of Fit with Regression-Model

4.1 Regression Model

As we can compute exact PPs based on Eq. 8, a natural approach is to regress the exact PP based on some key quantities from the relevance posterior. We therefore randomly sampled 10000 document pairs and their preference probabilities from the experimental data (see Section 5.1 for data detail), and randomly split them into 3:1 for training and testing. For each document pair (d_i, d_j), we take μ_i, μ_j as the regression features, and obtain the following linear regression model

$$\hat{P}(d_j \succ d_i) = 0.4855 + 1.2006 * \mu_j - 1.1815 * \mu_i, \qquad (9)$$

which, as expected, is proportional to the differences in μ. Its goodness-of-fit (GoF), as evaluated on the test set, is depicted in Figure 3(a), which exhibits a general lack of fitness, especially when the exact PP is near 0 and 1. We augmented regression features with other quantities like standard deviation and median, and did not improve the GoF in any visible way.

The lack of fitness is unsurprising, because PP is not linear to the difference in μ. For example, suppose $\mu_i = 0.90$ and $\mu_j = 0.95$. Although the difference is merely 0.05, $P(d_j \succ d_i)$ could be nearly 1 as posteriors with μ in that range are likely very spiky. This observation suggests that μ should be somehow stretched to emphasize the spikiness of the posteriors when μ is near 0 or 1.

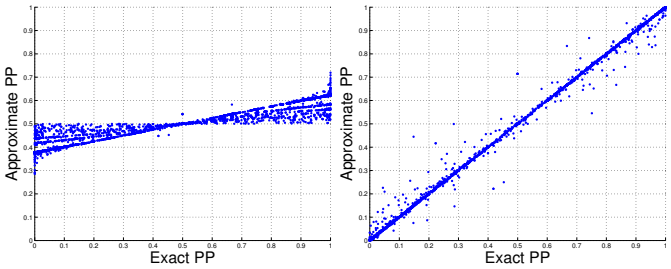
We therefore apply the log-odd function $f(x) = ln(\frac{x}{1-x})$ to both μ and the exact PP, and obtained the following regression model

$$f(\hat{P}(d_j \succ d_i)) = 1.0096 * f(\mu_1) - 1.0080 * f(\mu_2) - 0.0292, \qquad (10)$$

whose GoF is depicted in Figure 3(b), which looks much better than Figure 3(a).

Readers may have noticed that Eq. 10 is essentially a logistic regression of $P(d_j \succ d_i)$ with $f(\mu)$'s as the features. We verified that a logistic regression with μ's as the feature does no much better than Figure 3(a), which indicates that log-odd transformation is the key to good GoF.

Finally, we note that we here intentionally forgo nonlinear kernel regressions because (1) they need to remember a large number of support vectors, whose memory cost is exactly what we try to avoid, and (2) in kernel regression, many kernel functions will be evaluated for each document pair, and the time cost will delay the response time. As Eq. 10 has already approximated exact PP pretty well and is easy to compute, we will use it in this study.



(a) Parameterize $h(d_i) = \mu_i$ (b) Parameterize $h(d_i) = f(\mu_i)$

Figure 4: Goodness of Fit with BT-Model

4.2 Bradley-Terry Model

The Bradley-Terry (BT) model [4] is commonly used for paired comparison [7], and has recently been adopted in machine learning, *e.g.*, estimating multi-class probabilities from pairwise binary classification results [14] and learning to rank [5, 28]. In its general form, BT model states that

$$\hat{P}(d_j \succ d_i) = \frac{\lambda_j}{\lambda_j + \lambda_i},$$

where λ_i and λ_j are the "merit" of d_i and d_j respectively.

The BT model can be viewed as a special linear model [7]:

$$P(d_j \succ d_i) = F(h(d_j) - h(d_i)), \tag{11}$$

where $h(d_i)$ characterizes the utility of d_i and $F : \mathcal{R} \mapsto \mathcal{R}$ is a symmetric CDF, meaning that $F(-\infty) = 0$, $F(+\infty) = 1$, and $F(x) = 1 - F(-x)$. By choosing

$$F(x) = \frac{1}{1 + e^{-x}},$$

we have

$$\hat{P}(d_j \succ d_i) = \frac{e^{h(d_j)}}{e^{h(d_j)} + e^{h(d_i)}}, \tag{12}$$

which is equivalent to parameterizing λ_i with $e^{h(d_i)}$.

Based on Eq. 12, we need to figure out a proper utility function $h(x)$ for each document such that Eq. 12 approximates the exact PP. We tried $h(d_i) = log(\mu_i)$ and $h(d_i) = \mu_i$, which correspond to taking μ_i and e^{μ_i} as the merit of the document d_i, but neither gave satisfying results. Figure 4(a) depicts the goodness-of-fit with $h(d_i) = \mu_i$, which is the better of the two.

The dramatic effect of log-odd transformation in the above subsection reminds us that the same log-odd transformation may help with the BT model as well. By taking $h(x) = f(x)$, the log-odd function, we obtain

$$\hat{P}(d_j \succ d_i) = \frac{\frac{\mu_j}{1-\mu_j}}{\frac{\mu_j}{1-\mu_j} + \frac{\mu_i}{1-\mu_i}}, \tag{13}$$

and it gives what is shown in Figure 4(b). It is better than Figure 4(a), and is comparable to Figure 3(b).

We also tried the Thurstone-Mosteller (TM) model, which is essentially the same linear model (Eq. 11) but with $F(x)$ being the unit Gaussian CDF. Because of the similar curvatures between the sigmoid function and the unit Gaussian CDF, TM model behaves similarly as the BT model, and is hence not included in this study.

4.3 Model Connections

The above two methods are dramatically different on the surface: one is supervised and the other unsupervised. But the great similarity in GoF as shown in Figures 3(b) and 4(b) hints some underlying connections between them.

Noticing that $f(x) = F^{-1}(x)$, and applying $F(x)$ to both sides of Eq. 10, we get

$$\begin{aligned}
\hat{P}(d_j \succ d_i) &= \frac{1}{1 + e^{-(1.0096 * f(\mu_j) - 1.0080 * f(\mu_i) - 0.0292)}} \\
&= \frac{e^{1.0096 * f(\mu_j)}}{e^{1.0096 * f(\mu_j)} + \eta e^{1.0080 * f(\mu_i)}} \qquad (14) \\
&\approx \frac{\frac{\mu_j}{1-\mu_j}}{\frac{\mu_j}{1-\mu_j} + \frac{\mu_i}{1-\mu_i}}, \qquad (15)
\end{aligned}$$

where $\eta = e^{0.0292} \approx 1$ and the last equation holds by approximately taking 1.0096 and 1.0080 as 1.

Eq. 15 indicates that the regression model learnt from data is roughly the same model as that by parameterizing the BT model with $h(x) = f(x)$. This means that the *ad hoc* log-odd transformation as used in the regression model actually gives the "merit" that is appreciated by the BT model. Although the relevance of a document is fully characterized by the relevance posterior in CCM, $ln(\frac{\mu}{1-\mu})$ can serve as a single-number summary of the relevance. This summary is better than μ in that its difference between two documents approximates the preference probability (Eq. 10), which could be very handy in learning-to-rank algorithms.

However, it is yet unknown whether such observations generalize to relevance estimates from other click models. But based on previous studies that show strong consensus among various models [13, 12], we expect an affirmative answer. If the observation does generalize, then we have actually found a justifiable way to compute preference probability for non-Bayesian models. We will investigate this direction in the future.

Finally, we note that the $\eta = e^{0.0292}$ in Eq. 14 corresponds to the threshold that controls the probabilities of ties. Specifically, it means that the regression model represents a robust BT model that takes (d_i, d_j) pairs with $|P(d_i \succ d_j) - 0.5| \leq 0.0292$ as ties, regardless of the threshold θ. This echoes the idea of a previous work on "learning to rank with ties" [28].

5. EXPERIMENTAL RESULTS

We report on the experimental evaluation in this section. We first describe the experiment setup in Section 5.1, and then estimate the optimal value for threshold θ in Section 5.2. Finally, Section 5.3 elaborates on the comparison between PP-based and sort-based approaches with details.

5.1 Experiment Setup

As this study tries to quantify the relevance improvement of different reordering algorithms over a major search engine, substantial real-world click data is needed for the experiment. However, to the best of our knowledge, no public data is appropriate for this purpose. The data prepared for the "Workshop on Web Search Click Data"[1] looks promising, but it is unfortunately insufficient for building CCMs,

[1] `http://research.microsoft.com/~nickcr/wscd09/`

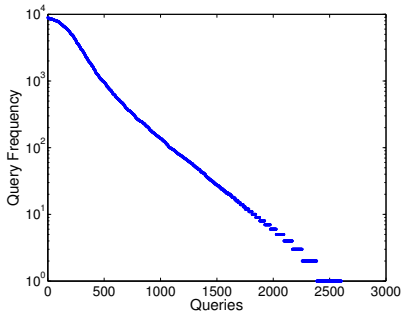


Figure 5: Query Frequency in Log Scale

e.g., unclicked documents are not available in the data. We therefore construct the experiment data as below.

We randomly sampled 2826 distinct queries from a commercial search engine, and cached the top-10 URLs for each query. Each query-URL pair was then rated by a trained human assessor in five scales from "perfect" to "bad". We computed the CCM relevance posterior for each query-URL pair, and consequently the pairwise preference probabilities for URLs w.r.t. the same query, using two weeks' click log in Aug. 2008. The query frequencies (capped at 10,000) in that time period is depicted in Figure 5. As can be seen, neither infrequent nor frequent queries were excluded in the experiment.

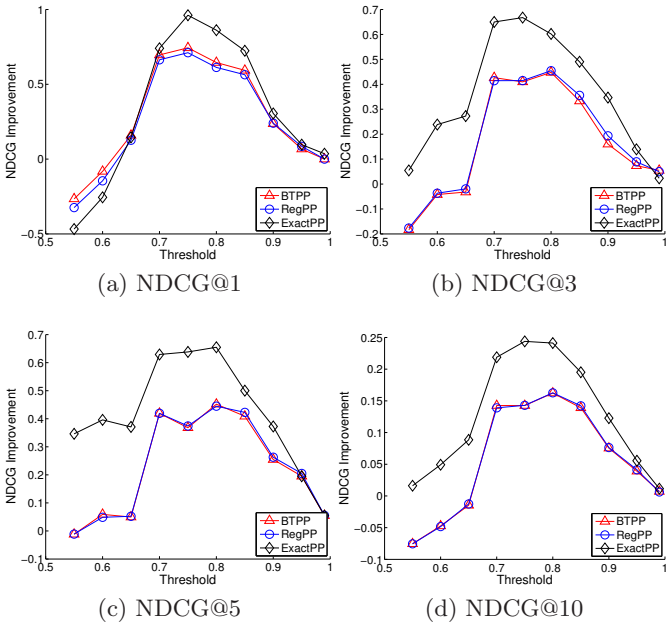
The Normalized Discounted Cumulative Gain (NDCG) [15] is used to compare the relevance of each ranking before and after reordering. The NDCG at position n for a given ranking is calculated by

$$NDCG@n = N\sum_{i=1}^{n}(2^{r_i}-1)/log(i+1),$$

where r_i is the rating assigned to the ith document in the ranking ($r_i = 5$ for "perfect" and 1 for "bad"), and N is a constant that makes NDCG equal to 1 for a perfect ranking. In order not to reveal the raw NDCG of the search engine, the NDCG difference between the reordering and the original ranking is taken as the evaluation metric, so that a positive number means improved relevance and otherwise for a negative value. For convenience, this NDCG change is multiplied by 100. As in previous literature, the NDCG changes at positions 1, 3, 5 and 10 are reported.

The following six algorithms are compared in this study:

- NumClk: Sort documents in descending order of the number of clicks each document receives.
- NumLastClk: Sort documents in descending order of the number of sessions in which the document is the last clicked one.
- NumOnlyClk: Sort documents in descending order of the number of sessions in which the document is the only clicked one.
- ExactPP: Re-order documents by Algorithm 1
- RegPP: Re-order documents by Algorithm 1 but with Line 3 replaced by $1.0096 * f(\mu_j) - 1.0080 * f(\mu_{j-1}) \geq f(\theta) + 0.0292$, according to Eq. 10.
- BTPP: Re-order documents by Algorithm 1 but with Eq. 13 plugged into Line 3.



(a) NDCG@1 (b) NDCG@3

(c) NDCG@5 (d) NDCG@10

Figure 6: NDCG Improvements w.r.t. Parameter θ

We also tried to augment the three sort-based algorithms by considering the positional bias (using the examination curve in Figure 1 of [19]). Specifically, we normalized a click on position i as a click with weight (1/the probability position i is examined), but this failed to improve the relevance. This result agrees with the findings in [25]. A possible reason for this result is the brittleness of the examination curve that is summarized from a subject study, whereas the true examination curve likely varies from query to query and even from session to session. We therefore agree with [25] that it is non-trivial to incorporate positional bias and dependencies through heuristics. In contrast, as will be seen in the following, the PP-based algorithms uniformly improve the relevance, because CCM implicitly estimates the positional bias for each query based on its click data and quantitatively measures the extent to which one URL is preferred to another.

5.2 Estimating Parameter θ

Before comparison, we need to first determine the threshold value θ for PP-based algorithms. For this purpose, we hold out 10% of the data, which consists of 283 distinct queries, and run the three PP-based approaches with different threshold values. Figure 6 plots the NDCG changes with different thresholds at different positions.

The four sub-figures exhibit the same trend across different algorithms and positions: the NDCG improvement starts from somewhere unimpressive with $\theta = 0.55$, stably picks up as θ goes up, and finally drops as $\theta \to 0.99$. This is a well-expected behavior: when θ is too low, many pairs with only marginal PP values are swapped. On the other hand, when θ is too high, many pairs with inconsistency probabilities high enough but not over θ are prevented from being swapped.

As seen from the four sub-figures, $\theta = 0.75$ appears to be a good choice, and this value also appeals to human intuition:

Table 1: Pairwise Hypothesis Testings between Algorithms: Each cell contains the p-value of the corresponding test, and those significant with $\alpha = 0.05$ are highlighted in bold. Values less than 0.001 or greater than 0.999 are written as 0 and 1, respectively

(a) p-values for testings with NDCG@1

	Orig.	BT	Reg	Exact	Clk	Last	Only
Orig.	1	0.997	0.996	1	0.336	0.319	0.319
BT	**0.004**	1	0.500	0.951	**0.007**	**0.006**	**0.006**
Reg	**0.006**	0.813	1	0.960	**0.009**	**0.008**	**0.008**
Exact	**0.001**	0.100	0.081	1	**0.002**	**0.002**	**0.002**
Clk	0.698	0.995	0.994	0.998	1	0.500	0.500
Last	0.714	0.996	0.994	0.999	0.875	1	0.688
Only	0.714	0.996	0.995	0.999	1	0.688	1

(b) p-values for testings with NDCG@3

	Orig.	BT	Reg.	Exact	Clk	Last	Only
Orig	1	1	1	1	0.612	0.581	0.111
BT	**0**	1	0.500	0.899	**0.009**	**0.009**	**0**
Reg.	**0**	0.726	1	0.918	**0.001**	**0.001**	**0**
Exact	**0**	0.125	0.102	1	**0.001**	**0.001**	**0**
Clk	0.408	0.992	0.991	0.999	1	0.313	**0**
Last	0.439	0.993	0.992	0.999	0.938	1	**0**
Only	0.898	1	1	1	1	1	1

(c) p-values for testings with NDCG@5

	Orig.	BT	Reg.	Exact	Clk	Last	Only
Orig.	1	1	1	1	0.892	0.878	0.473
BT	**0**	1	0.598	0.666	0.0576	0.052	**0.003**
Reg.	**0**	0.598	1	0.612	0.052	**0.047**	**0.003**
Exact	**0**	0.370	0.425	1	**0.037**	**0.033**	**0.003**
Clk	0.117	0.948	0.953	0.967	1	0.313	**0**
Last	0.132	0.953	0.958	0.970	0.938	1	**0**
Only	0.546	0.997	0.998	0.998	1	1	1

(d) p-values for testings with NDCG@10

	Orig.	BT	Reg.	Exact	Clk	Last	Only
Orig.	1	1	1	1	0.787	0.774	0.279
BT	**0**	1	0.428	0.308	**0.048**	**0.043**	**0.005**
Reg.	**0**	0.708	1	0.295	**0.048**	**0.043**	**0.005**
Exact	**0**	0.720	0.733	1	**0.003**	**0.003**	**0**
Clk	0.226	0.957	0.957	0.997	1	0.313	**0**
Last	0.240	0.961	0.961	0.997	0.938	1	**0**
Only	0.737	0.996	0.995	1	1	1	1

if a lower-ranked URL is preferred to a higher-ranked one with more than 0.75 probability, the two should be swapped.

Besides suggesting a proper threshold, the four sub-figures also show that ExactPP is consistently better than RegPP and BTPP. This observation reaffirms the validity of the exact PP, because it shows that the lack of GoF in Figures 3(b) and 4(b), even a little, immediately translates into a degradation in reordering quality. Although we cannot say the exact PP is the optimal measure to guide reordering, it is guaranteed that both RegPP and BTPP will perform better if they manage to achieve higher GoF with the exact PP.

After determining the value of the parameter θ using the 10% data, we take the rest 90% for testing. We note that here the generalization test is not across queries, but is instead on how different algorithms better aligns documents with human relevance judgments through click-based reordering, so the experiment is not a "in-sample" test.

5.3 Effectiveness Comparison

Figure 7 presents the comparison, and demonstrates the superiority of PP-based methods over those based on sorting. Within the sort-based methods, NumClk and NumLastClk are comparable with NumClk being slightly better, and NumOnlyClk appears to be the worst of the three. The unsatisfactory performance of NumOnlyClk is due to the following reasons. First, most only clicks are on URLs associated with frequent queries (*e.g.*, navigational), for which the current search engine has done pretty well, and hence there is no much space to further improve. Second, NumOnlyClk credits a URL only when the URL is the solely clicked one within a session, and this renders the number of only clicks less discriminative than the number of clicks. For this similar reason, we see NumLastClk is slightly worse than NumClk. While sort-based approaches are in general less effective, NumClk and NumLastClk clearly pick up for NDCG@5. This result indicates that clicks are indeed useful signals for reordering: in general, relevant documents in lower positions could be brought up by clicks, but refinement is needed to maximize the relevance improvement.

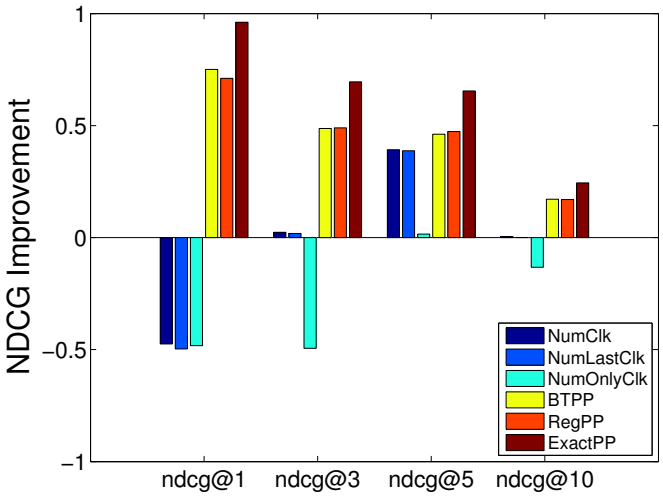


Figure 7: NDCG Improvements

On the other hand, the three PP-based methods improve the relevance across all positions. BTPP and RegPP achieve comparable performance while both of them are inferior to ExactPP. The comparable effectiveness between BTPP and RegPP is unsurprising because of their similar GoF as shown in Figures 3(b) and 4(b).

In order to quantify the comparison between algorithms, a set of pairwise hypothesis testings are performed between different algorithms and the original ranking. The objective is to quantify whether a method is better than another with statistical significance. The one-side Fisher's Sign Test [20] is used so that for each pair of methods (A, B) (*e.g.*, (ExactPP, NumClk)), A is better than B with statistical significance if the p-value is less than 0.05.

Table 1 lists the testing p-values with NDCG@1, 3, 5, and 10, respectively, and method names are shorthanded for typesetting requirements. The number within each cell

is the p-value for the corresponding test, and those less than 0.05 (meaning significant) are highlighted in bold. For example, the (ExactPP, NumClk) cell in Table 1(c) means that ExactPP is better than NumClk with statistical significance, based on measurements on NDCG@5. To see whether NumClk is better than ExactPP, one needs to resort to the cell (NumClk, ExactPP). As can be seen, the relative superiority between algorithms are consistent with what is shown in Figure 7.

6. DISCUSSION AND RELATED WORK

This study, in the first place, relates to the *U Rank*[2] and *SearchWiki*[3] features, which fall into the category of personalized search (*e.g.*, [21, 16, 26]). Both features allow registered users to explicitly reorder search results for their own use, *i.e.*, the re-ordered results will not affect the ranking for the general audience. Our study differs from them in the following two aspects. First, we aim at improving the general ranking while the two features facilitate personalization and/or sharing between friends. Second, our reordering is automatically derived from users' implicit feedbacks (*i.e.*, clicks) whereas the two features let users explicitly edit their own ranking. Despite the differences, the three approaches all agree that the original ranking is unlikely 100% aligned with users' preferences, and adjustments, either implicitly or explicitly, are needed as appropriate. Also, it is worth mentioning that here the reordering is post-rank, which is different from relevance feedback for retrieval models that is practiced in a "in-rank" fashion [24].

Second, this study builds upon, and hence relates to, click data analysis, which is a big and active research area. To some extent, clicks are like user votes on web search results, but with the caveat that, different from conventional votes, web users may not examine and vote on each document with equal care. Therefore, disciplined approaches are needed to interpret users clicks. Joachims et al. pioneer the study based on some eye-tracking experiments, and examine how users browse and click search results [18, 19]. Their study suggests that positional bias is significant, by showing that even when the ranking is totally reversed, top results (in the reversed order) still receive more clicks.

In an attempt to explain positional bias, Craswell et al. compare four click models that try to explain how the *first* click arises based on a real-world experiment [6]. Some follow-up work [10, 8, 9, 13] extends the study to modeling how multiple clicks arise in a session, and provides estimates of document relevance from clicks. While effective, the relevance estimates are mostly point-estimate, and it remains obscure how to derive preference probabilities from these point estimates. In this study, we leverage the Click Chain Model (CCM) [12] for its Bayesian modeling of the relevance, and compute the preference probabilities outright from relevance posteriors. Moreover, we also solve the problem of deriving preference probabilities from point estimates for the CCM case. We will investigate whether and how these CCM-specific findings generalize to other models in the future.

Preference probability is an important quantity because search is simply about learning preferences, and learning preferences from clicks is not new. As early as 2002, Joachims propose rules like "click $\succ$ skip above" to derive document preferences for the use of the Ranking SVM algorithm [17]. Later, he and his collaborators investigate a spectrum of rules based on the eye-tracking experiments [18, 19], and generalize the preference derivation from single query to across query chains [22]. A recent study by Shokouhi et al. shows that document reordering based on these preference rules does "*not lead to consistent improvement*", and "*may even lead to poorer results if not used with care*" [25]. In this paper, we circumvent heuristics by computing preference probabilities from CCM. Not only does this give a continuous probability instead of a binary preference, but it also saves many hassles and heuristics that have to be dealt with in practice, *e.g.*, query frequencies and document positions, *etc.*

We expect that the computed preference probabilities can be useful for many applications, and learning to rank [27] is one of them. In the first place, we can attach the probabilities to those binary preferences, which could immediately lead to a cost-sensitive version of Ranking SVM for instance. Second, the computed preference probabilities can also augment the training data for algorithms (*e.g.*, RankNet [5]) that currently rely on human judgements for pairwise preferences. Third, the document "merit" that is appreciated by the BT model can be incorporated as supplemental training features for learning a better ranker, which is line with previous work done by Agichtein et al [1, 2]. All these are in our list of future work.

Finally, in more general sense, this study is related to paired comparisons [7], because we find that a proper parametrization of the Bradley-Terry model coincides with the preference probability that is computed from CCM. Restricted to information retrieval, the BT model is usually used to regress the outputs from a ranking function for two documents to the desired preference between them [23, 28, 5]. In this study, we find that the odd of the point estimation is actually the "merit" appreciated by the BT model. This finding not only leads to an online computation method for preference probability, but it also backs the validity of the preference probability. As said before, it could be interesting, as well as rewarding, to check how the result generalizes to other relevance estimates.

7. CONCLUSIONS

In this paper, we proposed to accompany a search engine with an "always-on" component that resolves unsatisfactory orderings between documents by monitoring user clicks. Contrary to human intuitions, we showed that simple sorting by clicks is not precise enough. We put forward a preference probability-guided approach to reordering search results, and devised a principled way to compute preference probabilities from clicks, both precisely and approximately. A set of experiments based on real-world data were reported on in this study, which validated the effectiveness of the proposed approach. As outlined in the above section, there are many topics to be investigated in the future.

8. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR '06: Proceedings of the 29th*

[2] http://research.microsoft.com/projects/urank/

[3] http://googleblog.blogspot.com/2008/11/searchwiki-make-search-your-own.html

annual international ACM SIGIR conference on Research and development in information retrieval, pages 19–26, 2006.

[2] E. Agichtein, E. Brill, S. Dumais, and R. Ragno. Learning user interaction models for predicting web search result preferences. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–10, 2006.

[3] R. Baeza-Yates, A. Gionis, F. Junqueira, V. Murdock, V. Plachouras, and F. Silvestri. The impact of caching on search engines. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 183–190, 2007.

[4] R. Bradley and M. Terry. The rank analysis of incomplete block designs. 1. the method of paired comparisons. *Biometrika*, 39, 1952.

[5] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 89–96, 2005.

[6] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM '08: Proceedings of the First International Conference on Web Search and Data Mining*, pages 87–94, 2008.

[7] H. A. David. *The Method of Paired Comparisons*. Oxford University Press, second edition, 1988.

[8] G. E. Dupret, V. Murdock, and B. Piwowarski. Web search engine evaluation using click-through data and a user model. In *Proceeding of the Workshop on Query Log Analysis: Social and Technological Challenges (WWW '07)*, 2007.

[9] G. E. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 331–338, 2008.

[10] G. E. Dupret, B. Piwowarski, C. A. Hurtado, and M. Mendoza. A statistical model of query log generation. In *String Processing and Information Retrieval, 13th International Conference, SPIRE 2006*, pages 217–228, 2006.

[11] T. Fagni, R. Perego, F. Silvestri, and S. Orlando. Boosting the performance of web search engines: Caching and prefetching query results by exploiting historical usage data. *ACM Transaction on Information System*, 24(1):51–78, 2006.

[12] F. Guo, C. Liu, T. Minka, Y.-M. Wang, and C. Faloutsos. Click chain model in web search. In *WWW'09: Proceedings of the 20th International World Wide Web Conference*, 2009.

[13] F. Guo, C. Liu, and Y.-M. Wang. Efficient multiple-click models in web search. In *WSDM '09: Proceedings of the Second International Conference on Web Search and Data Mining*, 2009.

[14] T.-K. Huang, R. C. Weng, and C.-J. Lin. Generalized bradley-terry models and multi-class probability estimates. *J. Mach. Learn. Res.*, 7:85–115, 2006.

[15] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[16] G. Jeh and J. Widom. Scaling personalized web search. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 271–279, 2003.

[17] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, 2002.

[18] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 154–161, 2005.

[19] 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 Transaction on Information System*, 25(2):7, 2007.

[20] E. Lehmann and J. P. Romano. *Testing Statistical Hypotheses*. Springer, third edition, 2008.

[21] J. Pitkow, H. Schütze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel. Personalized search. *Commun. ACM*, 45(9):50–55, 2002.

[22] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 239–248, 2005.

[23] F. Radlinski and T. Joachims. Active exploration for learning rankings from clickthrough data. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 570–579, 2007.

[24] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. pages 355–364, 1997.

[25] M. Shokouhi1, F. Scholer, and A. Turpin. Investigating the effectiveness of clickthrough data for document reordering. In *ECIR'08: Proceedings of the The 30th European Conference on Information Retrieval*, pages 591–595, 2008.

[26] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 449–456, 2005.

[27] A. Trotman. Learning to rank. *Information Retrieval*, 8(3):359–381, 2005.

[28] K. Zhou, G.-R. Xue, H. Zha, and Y. Yu. Learning to rank with ties. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–282, 2008.