Incorporating Variability in User Behavior into Systems Based Evaluation

Ben Carterette[*], Evangelos Kanoulas[‡], Emine Yilmaz[†]
carteret@cis.udel.edu, ekanoulas@gmail.com, {eminey@microsoft.com, eyilmaz@ku.edu.tr}
[*] Department of Computer & Information Sciences, University of Delaware, Newark, DE
[‡] Information School, University of Sheffield, Sheffield, UK
[†] Microsoft Research, Cambridge, UK & Koc University, Istanbul, Turkey

ABSTRACT

Click logs present a wealth of evidence about how users interact with a search system. This evidence has been used for many things: learning rankings, personalizing, evaluating effectiveness, and more. But it is almost always distilled into point estimates of feature or parameter values, ignoring what may be the most salient feature of users—their variability. No two users interact with a system in exactly the same way, and even a single user may interact with results for the same query differently depending on information need, mood, time of day, and a host of other factors. We present a Bayesian approach to using logs to compute posterior distributions for probabilistic models of user interactions. Since they are distributions rather than point estimates, they naturally capture variability in the population. We show how to cluster posterior distributions to discover patterns of user interactions in logs, and discuss how to use the clusters to evaluate search engines according to a user model. Because the approach is Bayesian, our methods can be applied to very large logs (such as those possessed by Web search engines) as well as very small (such as those found in almost any other setting).

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Performance Evaluation*

Keywords

test collections, user logs, evaluation

1. INTRODUCTION

Every user approaches search in their own way. Some users only ever use web search engines like Google, and they trust these engines implicitly to give them the only result they need at rank one. Others use web search engines for more involved tasks, clicking through several links, reformulating queries, and using advanced features of the system. Still more use other search systems (such as those that index newswire, case law, or biomedical research) frequently; their approach in those systems is probably different from their approach on the web. And of course these groups do not begin to cover the whole space, not to mention the fact that a single user may have different approaches depending on their information need, mood, and a host of other unmeasurable factors.

Recently there has been interest in developing measures that explicitly model user interaction, notably *rank-biased precision* (RBP) [10], *expected reciprocal rank* (ERR) [3], *expected browser utility* (EBU) [16], and time-calibrated measures [14]. These measures use parameterized probabilistic models of actions users may take when browsing ranked results. But when the parameters are set to a single fixed value, these measures lose the ability to reflect variability in the user space. They essentially assume that however that single value is decided on, whether it be based on logs or based on pure rational thought, it reflects users as an aggregate. Like traditional measures, they may miss out on segments of the user population that experience search results in very different ways from the majority.

Fortunately, logs contain a lot of information about variability in how users are interacting with the system. With the right methods, we can take advantage of that variability to more precisely measure the effectiveness of the system to different segments of the user base. We present a Bayesian method for using logged user data to estimate probabilistic models of user interaction: starting with a prior distribution over users, we use a log to update it and produce a posterior distribution representing variability over the user space. We then show how we can use such distributions to evaluate systems, find separate clusters of behavior types for more targeted analysis, and analyzing the log to formulate hypotheses about how users interact with the engine.

One advantage of our approach is that it does not require huge logs, and thus can be used for search systems or tasks that have much smaller user bases than web search (which is the "traditional" domain for applications of mining user logs). This contrasts with other log-based approaches to evaluation and analysis such as bucket testing [7] and interleaving [11], which rely on large amounts of data to draw statistically-powerful conclusions.

This paper is organized as follows: in Section 2, we introduce the idea of forming a probability distribution based on a structural user model and informed by user logs and show how to compute it. In Section 3, we illustrate the use of this idea to partition a log into smaller segments and analyze behavior within those segments. Section 4 takes this a step further, using clustering to automatically partition the log into segments that may not be obvious to people analyzing a log, and showing that some of the clusters that emerge are quite interesting. In Section 5 we provide evidence that evaluating system quality on the basis of the appropriate segments of user logs can offer new insights into the performance of retrieval systems. We conclude in Section 6 with some directions for future work.

2. BEHAVIOR MODELS AND PROFILES

There are many effectiveness evaluation measures represented in the information retrieval literature. Almost all of them can be seen as encoding a model of a user that interacts with ranked results by stepping down the ranking one-by-one, looking at each document, and deriving utility from those that are relevant. For example, precision at rank k could be seen as modeling a user that always goes to rank k (never beyond) and finds every relevant document to that rank. Average precision (AP) can be understood as modeling a user that browses the retrieved documents from top to bottom and is equally likely to stop at any relevant document. When the user stops at a particular relevant document d, average precision assumes that she has found every relevant document to the rank at which d is retrieved [12] (this may be an unrealistic user model, but it is a user model nevertheless). Discounted cumulative gain has a more explicit user model, using graded judgments to model relative differences in utility amongst relevant documents and a reciprocal-log discount to model waning interest on the part of the user in continuing down the ranking.

Recently there has been interest in describing user interactions with a parameterized *probabilistic* model. The first such example may be Cooper's Expected Search Length (ESL) [4], which can be described as the average number of documents a user must examine before observing all documents relevant to the query. But the probabilistic component in ESL is related more to a system's approach to breaking ties than to anything about a user's behavior. More recently, rank-biased precision (RBP) was the first measure to be based on an explicit probabilistic model of the user: it models a user who, at each rank, flips a biased coin to decide whether to continue to the next document. If the coin comes up tails, the user continues; otherwise, the user abandons the search. This results in the rank of abandonment k having a geometric distribution: $P(k) = (1-\theta)^{k-1}\theta$, where θ is the probability that the coin comes up heads. Then RBP is defined as a sum over ranks k:

$$RBP = \sum_{k=1}^{\infty} rel_k P(k) = \sum_{k=1}^{\infty} rel_k (1-\theta)^{k-1}\theta$$

Another way of understanding θ is as the user's impatience[1] for continuing down the ranked results. The higher θ is, the less patient the user is and the more likely they are to abandon the search early; likewise, the lower θ is, the more documents they are likely to see during their session.

[1]Or patience; we use the two terms interchangeably. If θ represents patience, $1-\theta$ represents lack thereof, and the model can always be reparameterized.

2.1 The cascade model

RBP's model is very simple; there is much about user behavior that it does not model. The *cascade model* is a more complex model that has been shown to more accurately model behavior [5]. Like RBP, it assumes a user stepping down ranked results one by one. Unlike RBP, it does not assume that the probability of continuing is constant over the ranking. In the most general form of the model introduced by Craswell et al. [5], each URL can have its own probability of continuing; these probabilities (which we will call θ_d) could be based on features of the URL, the summary shown for it [16], prior distributions of dwell times [14], and even global features of the ranking or interface. Then the probability that a user stops at rank k is:

$$P(k|\theta) = \theta_{d_k} \prod_{i=1}^{k-1} (1 - \theta_{d_i})$$

Using a fixed θ such that $\theta_d = \theta$ for all d produces RBP.

Chapelle et al.'s expected reciprocal rank (ERR) is an effectiveness measure based on the cascade model and graded judgments. It defines θ_d as a function of the relevance grade of d, so that the probability of a user continuing depends on the relevance of the document they are currently looking at [3]. Thus we can say $\theta_{d_g} = \theta_g$ for all documents d judged by a human assessor to have relevance grade g, thereby reducing the parameter space to $|\mathcal{G}|$, the number of grades. ERR is then defined as the expectation of reciprocal ranks:

$$ERR = \sum_{k=1}^{\infty} \frac{1}{k} \theta_{d_k} \prod_{i=1}^{k-1} (1 - \theta_{d_i})$$

Chapelle et al. set $\theta_g = (2^g - 1)/2^{g_{max}}$, with $g \in \{0, 1, ..., g_{max}\}$ [3]. This is a heuristic, though, and not based on direct empirical evidence pertaining to utility.

Rather than one parameter like RBP, ERR has as many parameters as there are grades in the evaluation. When there are five grades, there are five parameters ($\theta_0 < \theta_1 < \theta_2 < \theta_3 < \theta_4$) each representing the probability that a user stops after seeing a document of that grade. The highest grade in a five-grade system (usually called "perfect") has the highest probability of stopping, modeling a user that is very unlikely to look at anything else after seeing the perfect result. These can also be thought of as representing "patience", but they are now more specifically patience given a particular level of relevance.

Expected browser utility (EBU) [16] is another evaluation measure that is based on the cascade model. Similar to ERR, EBU also assumes that the probability of a user continuing after reaching a particular rank k depends on the relevance of the document at rank k, but adds even more complexity, including the probability that a user clicks on a link given the attractiveness of the snippet (summary) of the document at rank k. Further complexity is possible; the only limiting factors are computation time and the ability to estimate model parameters.

2.2 Computing models from logs

Since the parameters in a user model are meant to reflect user behavior, it makes sense to use logs of user behavior—query logs—to estimate them [17, 18, 16, 9]. Most work focuses on estimating point values of parameters. But point estimates fail to capture variability in the population. Consider RBP's patience parameter: some information needs will naturally require less patience than others, for example the query "google" versus "linux guide". Furthermore, some users will naturally have more patience than others for the same query; using a point estimate assumes that the model captures all users equally, i.e. each user has equal chances of being satisfied and stopping at the first rank of the results for all queries. We argue that it is more interesting to look at a *distribution* of a parameter and compute distributions of effectiveness evaluation measures over the distribution of the parameter rather than point estimates of evaluation measures.

In general, let θ be a vector of parameters. Let $P(action|\theta)$ be a model of user actions parameterized by θ. For the RBP model, we could define $action$ = stop at rank $k, k \in 1, 2, ...$ and $P(k|\theta) = (1-\theta)^{k-1}\theta$. Here θ is the probability that a user goes on to look at result $k+1$ given that they are currently looking at result k, and can be thought of as "impatience" just as we described above. Given no information about user patience, we may initially assume that all values of θ are equally likely, e.g. $\theta \sim Unif(0, 1)$. Then we could compute a distribution of RBPs over the population of users by sampling a θ (which models a user) and then computing RBP with that θ.

2.2.1 Rank-biased precision

Of course, a uniform distribution does not come close to modeling the actual population of user patience. Fortunately we have a large source of data from which to glean information about user patience: a query log. Given a log L containing queries and clicks, we will want a posterior for θ conditional on the log:

$$P(\theta|L) \propto P(L|\theta)P(\theta)$$

Carterette et al. [2] showed how to compute $P(\theta|L)$ with a single pass over a log. First, for each individual search (unique query/user pair at a unique time) registered in the log, they count c (the number of clicked URLs) and r (the number of unclicked URLs before the last rank clicked); c is modeled as a function of r and θ using the negative binomial distribution. Using a beta prior for $P(\theta)$, which is conjugate to the negative binomial distribution, gives the following three equations to compute $P(\theta|L)$ in one pass:

$$P(\theta|L) = \sum_{r=0}^{\infty} P(r|L)P(\theta|r, L)$$

$$P(r|L) = \frac{M_r + 1}{\sum_{i=0}^{\infty} M_i + 1}$$

$$P(\theta|r, L) = Beta\left(\theta|\alpha + \sum_{i=1}^{M_r} C_i, \beta + rM_r\right)$$

Here M_r is the total number of searches with r unclicked URLs and the sum $\sum_{i=1}^{M_r} C_i$ is the total number of clicks for searches with r unclicked URLs. α and β are hyperparameters which we define to be 1. There is a special case when r and c are both 0: since there are no clicks from which to estimate patience, we also keep a flat prior that has probability of being sampled from equal to the proportion of searches for which r and c are both 0. The pseudocode in Algorithm 1 illustrates this.

Algorithm 1 Compute posterior distribution $P(\theta|L)$ for ERR with a single pass over a log L.

```
Require: A log $L$ in which each line has a user/cookie ID,
    a query ID, URLs with relevance judgments, and the
    ranks clicked.
Ensure: A multivariate distribution $P(\theta|L)$.
 1: for each line $\ell$ in $L$ do
 2:   $c \leftarrow$ total number of clicks
 3:   $k_{last} \leftarrow$ rank of deepest click
 4:   for each grade $g$ with a URL in $\ell$ do
 5:     if $c = 0$ then
 6:       $M_g[null] \leftarrow M_g[null] + 1$
 7:     end if
 8:     $k_g \leftarrow$ min rank of URL with grade $g$
 9:     $c_g \leftarrow$ number of clicks on documents with rank $\geq k_g$
10:     if $c_g > 0$ then
11:       $r \leftarrow k_{last} - c_g$
12:       $M_g[r] \leftarrow M_g[r] + 1$
13:       $C_g[r] \leftarrow C_g[r] + c_g$
14:     end if
15:   end for
16: end for
17: for each grade $g$ do
18:   for $B$ trials do
19:     sample a value $r$ from indexes of $M_g$ according to
        frequency distribution
20:     if $r = null$ then
21:       sample $\theta_g$ from $Beta(1, 1)$
22:     else
23:       sample $\theta_g$ from $Beta(1 + C_g[r], 1 + r \cdot M_g[r])$
24:     end if
25:   end for
26: end for
```

To compute $P(\theta|L)$ with a single pass, we will assume that each line in the log has (at a minimum) a user or cookie ID, a query ID, and the ranks of clicked URLs. From that information we extract the deepest rank clicked k_{last}, the total number of clicks c, and then compute $r = k_{last} - c$. We increment M_r by one and increment C_r by c. When complete, we know the total number of queries with r no-clicks as well as the total number of clicks for those queries; this is what we need to compute $P(\theta|r, L)$. To do that, we repeatedly sample a value of r from the M_r distribution and use the corresponding M_r and C_r to compute parameters for a beta distribution from which we sample to generate θs.

2.2.2 Expected reciprocal rank

In ERR, the parameter θ represents the probability of a user continuing to browse conditional on seeing a document of a particular grade. We will need to estimate a distribution $P(\theta|L, g)$ for each grade g. As with our RBP distribution, we sum over r to obtain:

$$P(\theta|L, g) = \sum_{r=0}^{\infty} P(r|L, g)P(\theta|r, L, g)$$

We will understand the distribution $P(r|L, g)$ as the probability of observing a query with r unclicked documents given that there is a document of grade g in its results. $P(\theta|r, L, g)$ is the probability that the user will have patience to continue browsing *past* the document of grade g in those results.

We change the RBP algorithm outlined above to only

	ℓ_1			ℓ_2		
rank	g	URLs	click?	g	URLs	click?
1	4	URL$_1$	✓	1	URL$_1$	✓
2	0	URL$_2$		2	URL$_2$	
3	0	URL$_3$		1	URL$_3$	
4	0	URL$_4$		2	URL$_4$	✓
5	0	URL$_5$		1	URL$_5$	
6	1	URL$_6$		0	URL$_6$	
7	0	URL$_7$		2	URL$_7$	✓
8	0	URL$_8$		0	URL$_8$	✓
9	0	URL$_9$		0	URL$_9$	
10	0	URL$_{10}$		0	URL$_{10}$	✓

Table 1: Two example rankings of URLs with relevance judgments (given as a numeric grade g) and clicks (✓).

count clicks that occur on or below a document of a particular grade; Algorithm 1 gives complete pseudocode. The first 16 lines compute numbers from the log, storing counts of queries and total numbers of clicks for each r and each grade in the $M_g[]$ and $C_g[]$ arrays. The special case when $r = 0$ and $c = 0$ is held in a special place in M_g given a *null* index in this pseudocode. The last 10 lines actually perform the sampling.

As an example of how the algorithm works, suppose we have a log with two lines shown in Table 1 (as columns). The first, ℓ_1, has a "perfect" document at rank 1 and a "fair" document at rank 6. The user has clicked at rank 1, and that is the last registered click. There are no unclicked documents, so $r = 0$. $M_4[0]$, the number of lines with at least one "perfect" document in the ranking and no unclicked documents, is incremented by 1; $C_4[0]$, the total number of clicks for lines with at least one "perfect" document and no unclicked documents, is also incremented by 1. Since there were no clicks on or after the "fair" document, we simply assume the user never saw it, and thus do not take it into consideration; M_1 and C_1 are unchanged (as are M_2, C_2, M_3, and C_3).

The second line, ℓ_2, has three "fair" documents (at ranks 1, 3, and 5) and three "good" documents (at ranks 2, 4, and 7). The user has clicked on ranks 1, 4, 7, 8, and 10. For $g = 1$, there are 5 unclicked documents and 5 total clicks on or below the first rank with a "fair" document; $M_1[5]$ is incremented by 1, and $C_1[5]$ is incremented by 5. For $g = 2$, $k_g = 2$. Then, treating the document at rank 1 as unclicked, there are 6 unclicked documents and 4 total clicks, so $M_2[6]$ is incremented by 1 and $C_2[6]$ is incremented by 4. The click at rank 1 provides no evidence for whether a user will continue after rank 2, so treating the document at rank 1 as unclicked functions as a prior for the user having the patience to get to rank 2 in the first place (independently of the relevance judgments).

2.3 Fitting the models to data

To fit one of the models above, we need query log data. For RBP, we only require queries and ranks of clicks. For ERR we need query, ranks of clicks, and relevance judgments. The *amount* of query log data is somewhat incidental; we can compute a posterior whether the log is very large or very small.

We have accumulated a rather large amount of log data from different search engines. The AOL data is well-known;

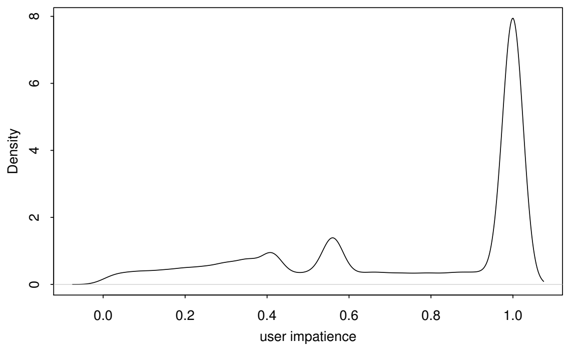

Figure 1: Posterior distribution for RBP's patience parameter θ computed from a web search engine log. Note that higher values of θ reflect *less* patience for browsing down a ranked list.

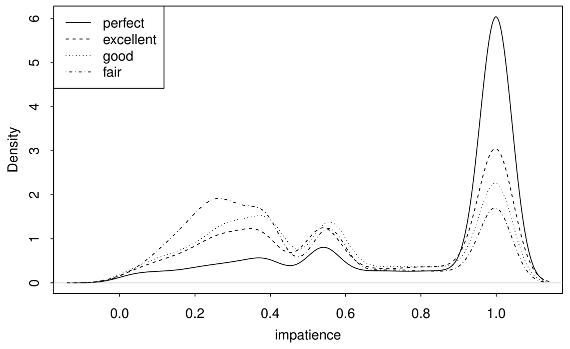

Figure 2: Posterior distributions for ERR's parameters $\theta_1..\theta_4$ computed from a web search engine log. Note that higher values of θ reflect greater likelihood of stopping after encountering a document of that grade.

it has queries, user IDs, and click ranks, but no relevance judgments. Yahoo! recently released log data under its Academic Relations Webscope program[2] consisting of anonymized query IDs, cookie IDs, click ranks, and graded relevance judgments (on a five-point scale). We have an additional proprietary log that is similar to the Yahoo! log but that also contains human judgments of query type. Most of the results we present are based on the Yahoo! log.

We have elected to define $\theta_0 \equiv 0$, i.e. we assume that a user will never abandon the search after seeing a "bad" URL (they will always continue to look at the next one). This is partly because ERR will not use judgments of nonrelevance otherwise, and partly because there are many missing judgments in our log data. These missing judgments likely affect the distribution of θ_0 more than any other parameter, since there are far fewer documents that have actually been judged "bad" versus "fair".

Figure 1 shows a posterior distribution for RBP's θ based on a web search engine log. Note the high peak at $\theta \approx 1$; this means that it is very common for users to only click on the top-ranked result. Figure 2 shows four posterior distribu-

[2]`http://webscope.sandbox.yahoo.com`

tions, one for each of ERR's parameters for four relevance grades. Note that the peak is present in all four, but its prominence decreases dramatically as relevance decreases, while the prominence of the hump to the left increases as relevance decreases. This supports the assumption of the cascade model that users are less likely to continue browsing after seeing more highly-relevant documents.

2.3.1 Qualities of the distribution

We note a few peculiar qualities of the distribution plots in Figures 1 and 2. First, there are clear peaks at certain points. Second, the peaks begin to disappear below 0.4, with the distribution becoming more smooth. Third, the distribution is quite flat in between the peaks above 0.4.

All three of these qualities are emergent from the Bayesian updating of the negative binomial model. The key parameters are c, the number of clicks, and r, the number of unclicked URLs. Consider an infinitely-long log in which every query has the same values of c and r. Using our approach, a posterior distribution of θ given this log will have all its mass at a single fixed value of θ. The negative binomial distribution is such that for most values of c and r, those θs occur below 0.4; thus if c and r were uniformly distributed through the log, the distribution would have most of its mass to the left. This is counteracted by the fact that low values of c and r are far more common—in particular, $c = 1$ and $r = 0$ or $r = 1$ (i.e. clicks at ranks 1 and 2)—and result in higher values of θ. For $c = 1, r = 0$ (a single click at rank 1) in an infinite log, θ will be 1; since this case is common in logs, there is a peak there. For $c = 1, r = 1$ (a single click at rank 2), it will be 0.5, and the peak near 0.5 is again explained by the commonness of that case. As c increases and r stays low (i.e. as users click more and more of the ranked URLs), values of θ increase towards 1, but these cases are rare in the logs, and therefore mostly subsumed by the flat prior. As r increases (and users click fewer of the ranked URLs), low values of θ become more frequent and less discriminant, and therefore the distribution is smoother in the lower range.

So while the distributions look somewhat strange, the peculiarities are explainable from the model and the nature of the log. Furthermore, a log that looks fundamentally different (by not having so many queries for which users abandon early, for instance) will have a different distribution. We shall see some examples below.

2.4 Evaluating using parameter distributions

Once we have a posterior distribution, we can form a marginal distribution for an evaluation measure reflecting effectiveness over the population. For each possible value of RBP for a topic, we compute the probability of getting that value if we were to sample θ from the posterior $P(\theta|L)$:

$$P(RBP = y) = \int P(RBP = y|\theta)P(\theta)d\theta \qquad (1)$$

where $P(RBP = y|\theta)$ is 1 if RBP computed with parameter θ is y and 0 otherwise. This looks a bit complicated but it is really quite simple; we can compute it by sampling θs from $P(\theta|L)$, computing RBP based on that θ, and over many trials forming the density $P(RBP)$. When we have many topics to evaluate, we just do this for each one, then average the RBPs for each value of θ. The marginal distribution is then over mean RBP.

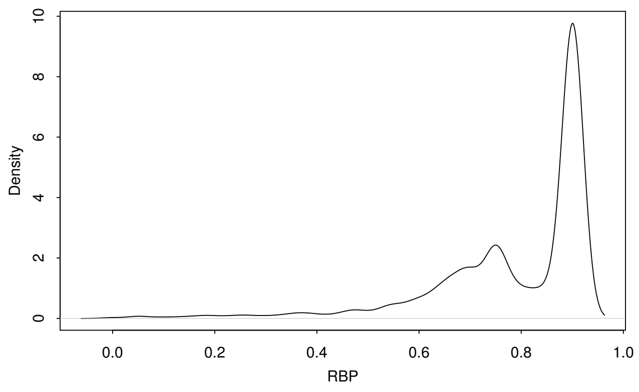

Figure 3: Distribution of mean RBP over the distribution of θ shown in Figure 1.

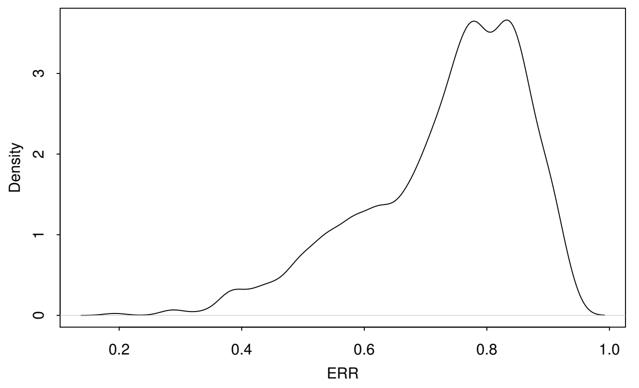

Figure 4: Distribution of mean ERR over the distribution of θs shown in Figure 2.

Similarly,

$$P(ERR = y) = \int \cdots \int P(ERR = y|\theta)P(\theta)d\theta_1 \cdots d\theta_{g_{max}} \qquad (2)$$

Since θ is a vector of $|\mathcal{G}|$ parameters, there is an integral over each parameter. Still, computing $P(ERR)$ is as simple as computing $P(RBP)$; we sample a vector of parameters θ from distributions such as those in Figure 2 and compute ERR with those parameters. Note that we are *not* integrating over θ_0. As discussed above, we fix $\theta_0 = 0$, since ERR does not make sense otherwise.

Figure 3 shows the distribution of mean RBP for one of our web search engines based on $P(\theta|L)$ computed from its log. Note the peak near RBP = 0.9; this reflects the fact that for most users only the first document is examined, and for many queries that is indeed good enough. Figure 4 shows the distribution of mean ERR for the same engine based on the distributions in Figure 2. ERR's more precise user model suggests that while this engine has a high ERR for most users, there may be some for which the user experience is substantially less positive.

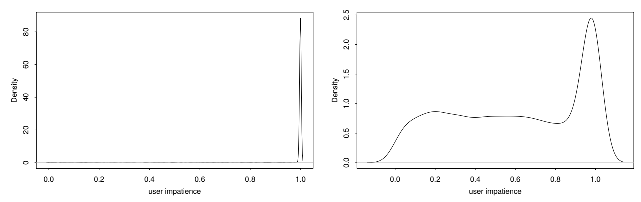

Figure 5: Left: posterior distribution for RBP's parameter θ for the query "google" in the AOL log. Right: the same for the query "weather".

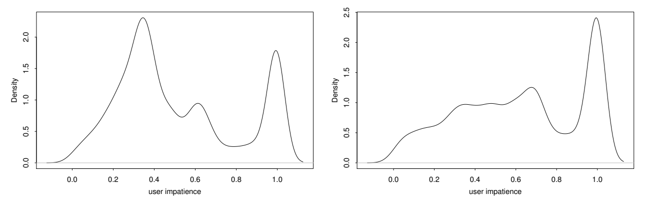

Figure 6: Posterior distributions for RBP's parameter θ for two users in the AOL log. Both users submitted more than 1,000 queries in the log.

3. DECOMPOSING USER PROFILES

Above we showed how to compute a distribution for an effectiveness measure parameter given a complete log. The log encompasses a great variety of different types of users, queries, and tasks or information needs. Instead of computing one distribution for the whole log, it may make sense to compute different distributions for different segments of the log, then evaluate separately using those distributions.

For instance, returning to the example of "google" and "linux guide", one would expect that for queries similar to "google", most users would interact with the search engine in a similar way, mainly clicking on the top returned result. Hence, we expect a narrow distribution of the parameter θ around high values. On the other hand, for queries similar to "linux guide" we expect users to demonstrate a high variability in their interaction with the returned results. It makes more sense for queries similar to "google" that the retrieval system to be evaluated against a narrow distribution of the parameters θ, while for queries similar to "linux guide" it be evaluated against a different and possibly high-variance distribution of the parameters θ. (Note that similarity will be defined more carefully in Section 4.)

In the limit, we could compute a distribution for a single user interacting with results for a single query; this would reflect the possibility that a user could interact with results differently depending on many factors that are not captured in the log, not known to researchers currently, or cannot be measured in any practical way.

3.1 Query profiles

We can take a log and partition it into multiple smaller logs, one for each unique query. We can then use these smaller logs (which we will call L_q) to inform a posterior distribution for θ, which we can denote as $P(\theta_q|L_q)$. Looking at a posterior distribution for a query tells us something about how users are interacting with the system for that particular query; comparing distributions between two queries tells us something about differences in user interactions for the two. We have done this for all of our logs, but the only one that has original query text is the AOL log, so in this section we focus on that. Note that since the AOL log has no relevance judgments, we can only look at the RBP parameter, not the ERR parameters.

Figure 5 shows the RBP distribution for the queries "google" and "weather" in the AOL query log. The former is clearly highly navigational: 75% of users who input the query clicked on rank 1 and stopped, and another 22% didn't click anything. The remaining 3% went deeper into the ranking, with about half of those looking at rank 3 and the other half going beyond rank 3.

The latter is more varied. It has some navigational qualities, as 74% of users that clicked something clicked on the top-ranked result. But only about 36% of users with this query clicked anything; 64% did not click any of the ranked results. This may suggest that the query is poorly-formed (some users may be looking for a site like `weather.com`, others may be looking for local weather). It may also suggest that there was a weather vertical in the interface (like there is in modern search engines) that gave the user everything they needed without requiring a click. Clearly using the same distribution of the parameter θ when evaluating a system over these two queries is not appropriate.

3.2 User profiles

Similarly, we can partition the log into multiple smaller logs, one for each unique user. We will call these L_u, and again we can use them to inform a behavioral profile for an individual user. Looking at these can tell us something about how individual users experience results (conditional on the queries they are submitting).

Again, the AOL log is the only one with user information that can be directly linked to queries. The other logs have dissociated user information from queries and so cannot be used for these illustrations.

Figure 6 shows the RBP parameter distributions for two different users in the AOL query log (both of whom have more than 1,000 queries in the log). These users are quite different; both are patient enough to have significant probability mass in the left part of the distribution, though both submit a significant number of queries for which they only click at rank 1. The right plot shows a user that clicks often enough that there is only a small dip at $\theta_u = 0.8$; as we discussed in Section 2.3.1, most of these distributions have a gap between $\theta_u \approx 1$ and $\theta_u \approx 0.5$ because few users click more times than they do not click (i.e. c is rarely greater than r except when $c = 1$ and $r = 0$). This particular user *does* have a tendency to click on many results, and that shows up in the shape of the distribution.

3.3 Task profiles

If we have some indication of user task, we can aggregate over user/queries that match that task. One example alluded to above is *informational* and *navigational*, a very common classification scheme for web queries. We have human navigational and informational judgments for one of our search engine logs; the posterior parameter distributions for the two classes are shown in Figure 7. The navigational class clearly shows a high peak near $\theta = 1$ with almost nothing below that. The informational class also has a peak near $\theta = 1$, but it is a smaller peak and there is still significant

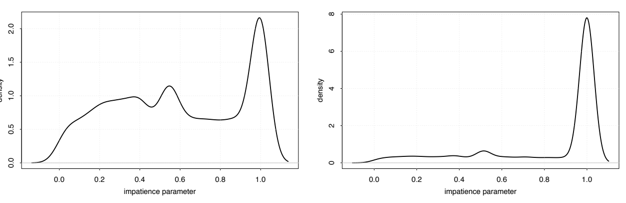

Figure 7: Profiles for informational and navigational classes.

density in the lower part of the domain. These reflect the expected behavior for these two query types.

An interesting question here is whether there are queries that have been assigned a particular task class but have particular instances for which user behavior reflects a different class. For example, "google" is almost certainly a navigational query, and indeed 76% of users with that query in the AOL log clicked only on rank 1. But there is a small group of users—about 3%—that click as far down as rank 5; for these, it might be argued that they have a more informational need that spurred that query. The remaining 21% did not click anything.

4. CLUSTERING PROFILES

Clustering query logs is useful for discovering patterns in the ways users are interacting with an engine. Most work on clustering within logs focuses on clustering queries, using similarities among clicked URLs [15, 1] or similarities in terms used in query refinements [13]. Such work could be applied to users as well, forming clusters based on similar query formulations [8].

Behavior profiles give us the ability to cluster the log according to a user model and the *variability* of interaction with the engine rather than anything about text or specific URLs. Text-based clustering cannot capture some of the most fundamental divisions that have been identified in logs, particularly informational versus navigational needs—there is not much in the way of term similarity to connect two randomly-chosen navigational queries, for instance, nor is there much that would allow a clustering algorithm to put two randomly-chosen informational queries together. By clustering on distributions, we use click position information, but we also take advantage of variability in click positions. Furthermore, we can cluster along any dimension of interest we can identify in the log (e.g., clustering per query, per user, per session, etc.).

Apart from revealing interesting behavior profiles, clustering can be helpful in evaluating retrieval systems. If we could collect clicks for every query in a test collection such as those produced by TREC, then we could infer a patience distribution for each query separately and evaluate retrieval systems using the appropriate parameter distribution. In systems-based experimentation, we do not have the ability to observe users interacting with the retrieval systems. If query clustering based on behavior profiles could provide a classification of queries with meaningful semantics (e.g. navigational vs informational queries, or precision-oriented vs recall-oriented queries), then human assessors could identify the class for each query in the collection and use a single parameter distribution for each such class.

4.1 Clustering algorithm

As mentioned above, we can form posterior distributions for a single instance of a query, all instances of a single unique query, any single search performed by a user, all searches performed by one user, or any other groups of queries or users. In this section we assume we have some number m of posterior distributions $P(\theta_m|L)$ and our goal is to cluster them.

In order to cluster the distributions, we use the clustering algorithm proposed by Kejzar et al. [6] for clustering discrete distributions. The proposed algorithm is based on a modification to the standard K-means (leaders) clustering algorithm. Instead of using the standard squared Euclidean distance as the distance metric, it uses the *relative error metric*, which its authors have shown to provide more informative results than the Euclidean distance.

Our distributions are (strictly speaking) continuous, not discrete, so to use this algorithm we must transform them to discrete distributions. We do this by sampling and binning: for each value of θ sampled from a posterior $P(\theta|L)$, we place it into one of eleven bins by rounding it to the nearest tenth. Therefore we are essentially clustering on 11 features, each of which represents the probability that a sampled θ would have (roughly) a particular value, or in other words, the probability that a sampled "user" would have a given degree of patience for browsing results. We tried other discretizations into more bins, but they did not produce substantially different clusters than what we show here.

Once the algorithm has placed each posterior distribution into a cluster C_i, we re-compute a posterior $P(\theta_{C_i}|L)$ for the cluster by aggregating log data. When the clustering is complete, we have K posterior distributions, each associated with one of our clusters.

4.2 Cluster-based exploration

We first computed posterior distributions $P(\theta_q|L)$ for every unique query in our proprietary log. We then formed increasing numbers of clusters of queries to look for patterns. Our first two clusters are shown in Figure 8. Note the similarity to the posterior distributions for the navigational and informational classes shown in Figure 7: our first cluster looks very much like the navigational class, and the second looks like the informational class only with less of a peak at $\theta_{C_2} \approx 1$.

For our proprietary log we have human labels of navigational/informational intent, and we compared the queries in each cluster to those judgments. The first cluster in Figure 8 consists of about 1600 queries, of which about 1200 (75%) have been labeled "navigational" by a human assessor. The second cluster consists of about 3900 queries, with 700 (18%) labeled "navigational". About 62% of all navigational queries fell into the first cluster, with the remaining 38% in the second. Thus, while the clusters seem similar to navigational and informational intents, they are also somewhat different. It is gratifying to see that the informational/navigational split is "real" (indeed seemingly the primary dimension along which queries are clustered), but also interesting that there are queries that have been labeled as one class but have behavioral patterns more like the other. It may be that instead of an informational/navigational split, the more salient description is "high-precision" versus "high-recall", with users issuing the former satisfied by one highly-relevant document (which may be a Wikipedia page for an

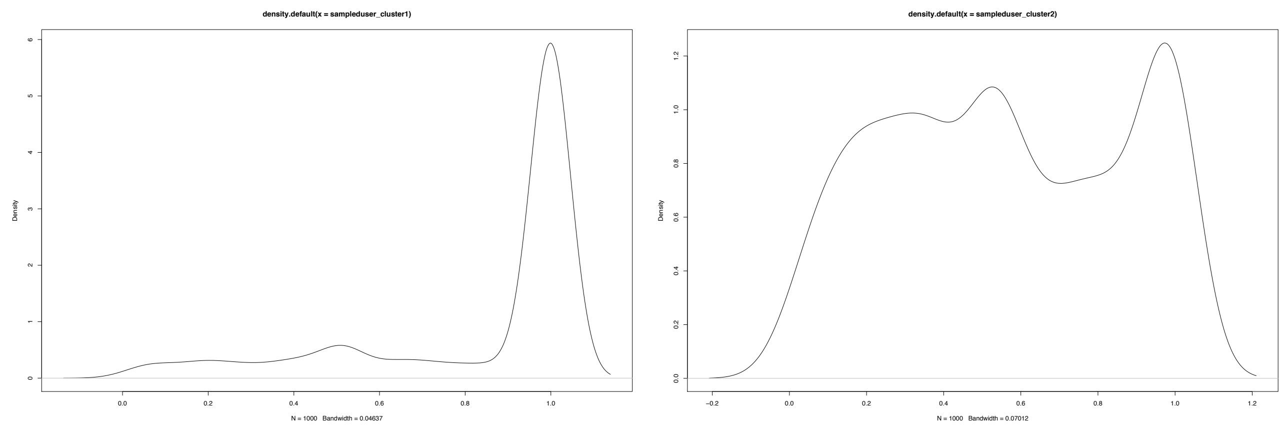

Figure 8: Posterior distributions computed from click information for two clusters of queries. The left seems to be a "navigational" cluster and the right an "informational" cluster, but these labels are not perfectly correlated to the clusters.

informational query) and users issuing the latter requiring more relevant documents and thus progressing deeper into the ranking.

We also tried forming three clusters from this data (not shown). In this case there are two clusters that are even more clearly navigational and informational (comprising 83% and 15% navigational queries respectively), with a third cluster that is somewhere in between (37% navigational). This third cluster seems to consist of lower-volume queries for which there is not much click evidence to move the prior, but it may still suggest that there are types of intents other than the two described above.

Figure 9 shows two clusters of queries for the Webscope log, this time with the distributions superimposed. These distributions look quite similar to those in Figure 8: one has a very high peak near 1 with little density elsewhere, while the other has a smaller peak at 1 that is only slightly higher than the other high points in the distribution. We do not have navigational/informational class labels for this data, but since we have relevance judgments we can look at the proportion of queries that have at least one "perfect" document that appear in each cluster (assuming that a query that has an associated document labeled as "perfect" is likely to have navigational intent). By that measure, the first cluster has 63% of the queries with at least one "perfect"; the other has about 37%.

Figure 10 shows four clusters of queries for the same log. The first two clusters look very similar to the two in Figure 9, and the next two look very flat, with only a slight rise in cluster C_4 near $\theta_{C_4} \approx 0.8$. Their flatness suggests that they are clusters of queries with low click volume; there does not seem to be information to update the prior. Looking at them closer shows that they have peaks and valleys much like others we have seen, and thus are capturing *something* about behavior, but at this scale those peaks and valleys are not visible. It turns out, however, that they are meaningful; we will see this in Section 5 below.

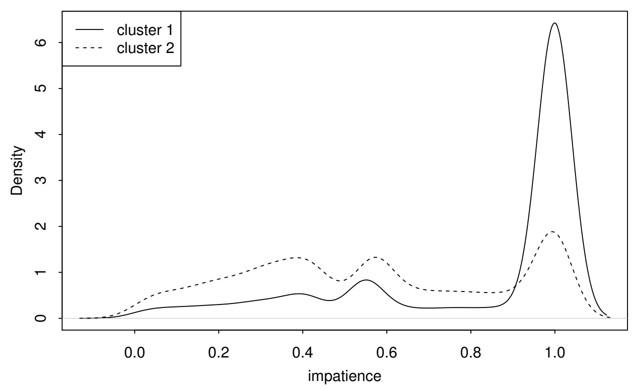

Figure 9: Posterior distributions computed from click information for two clusters of queries.

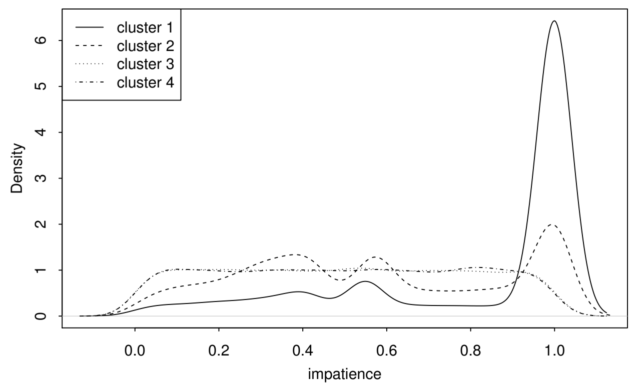

Figure 10: Posterior distributions computed from click information for four clusters of queries.

5. EVALUATION BASED ON CLUSTERS

The next question is whether the evaluation of a system differs depending on which data it has been evaluated against, and in particular whether the choice of cluster affects conclusions drawn from the evaluation. If we reach substantially different conclusions about a system depending on the cluster we evaluate against, it suggests that system evaluation should *not* be performed with the same uniform set of parameters, but instead should be adapted for the particular query, user, or task.

To perform these evaluations, we used the cluster distributions shown above in Eq. 1 and 2 to generate distributions for mean RBP and mean ERR; that is, we select a patience cluster, sample a value of θ (either a scalar or a vector value, depending on measure) from that cluster, evaluate all queries in the log using that value of θ in RBP or ERR, and then average the RBP/ERR values over queries. Over many such trials we obtain a marginal distribution for the measure over the parameter posterior distribution in each chosen cluster. Note that all of our clusters are based on the RBP parameter distributions; we formed posterior distributions for ERR parameters by computing them from the queries and clicks in each existing cluster based on RBP.

Figure 11 shows evaluation results given the two clusters in Figure 9. The left plot shows distributions for mean RBP: it is always quite high (above 0.8), but it also clearly tracks the differences between the two clusters. RBP calculated against the first cluster has a high peak near RBP $\approx$ 1, since most queries that fall into that cluster have a "perfect" document at rank 1 that is the only one that users click on. There are still enough queries in this cluster with clicks deeper in the ranking that some of the users it represents are more patient; these patient users find few relevant documents after the top-ranked one and therefore find the system has a lower RBP (the peak around 0.75). RBP calculated against the second cluster has a much smaller peak at RBP $\approx$ 1 and more mass lower in the distribution. For more patient users, the system represented by this log is significantly underperforming on average.

The right plot in Figure 11 shows distributions for mean ERR. They are clearly very different from each other. The distribution based on the first cluster evaluates the system highly on average, but has a great deal of variability, with values below 0.5 not uncommon. The peak around ERR$\approx$ 0.8 reflects the high probability of users abandoning after the first "perfect" document. The distribution based on the second cluster tends to evaluate the system much worse, with an expected ERR around 0.55. This strongly suggests that the data used to evaluate the engine is important, and different sources of data can lead to very different conclusions.

We also looked at evaluation against four patience clusters. Recall from Figure 10 that two of the clusters seemed quite flat, with little apparent different between them, suggesting that they may be artifactual rather than reflect some real difference in user behavior. But as Figure 12 shows, the evaluation results for the four clusters differ greatly. The first two clusters result in similar RBP distributions as those in Figure 11, suggesting that queries in the other two clusters are not important to defining the shape of the first two clusters (in some sense supporting the idea that those two clusters reflect something real about behavior). The two new clusters are similar, but one clearly has a higher peak at RBP$\approx$ 0.8, reflecting the very slight bump in the posterior distribution near $\theta_{C_4} \approx 0.8$ above θ_{C_3}. Furthermore, these two distributions are both very different from the other two.

Figure 12 also shows distributions for mean ERR. Again the first two are similar to those in Figure 11, and the two new distributions are both different from each other as well as different from the original two.

These results point to a very high variability in how users will perceive a system's effectiveness depending on both the query and the users themselves. There is therefore much potential for improving effectiveness.

6. CONCLUSIONS

We have presented a method for estimating a posterior distribution for parameters of a probabilistic user model from log data. In particular, we showed how to apply the method to the user models that the evaluation measures RBP and ERR are based on and compute posterior distributions for RBP's "patience" parameter and ERR's relevance grade-based parameters with a single pass over a log. Since our method is Bayesian (and thus involves updating a prior based on evidence), it can be used with very small amounts of log data as well as very large. We used it to form distributions for unique queries, unique users, and individual submissions of a query by a user. With these sorts of distributions, we can cluster instances in the log according to similarity in browsing behavior; by examining these clusters we found that evaluation results can depend heavily on the data used to estimate evaluation measure parameters.

As noted above, this reveals a great deal of opportunity for better understanding and improving effectiveness. Techniques could include using different retrieval algorithms for queries or personalizing the retrieval algorithm for users, in both cases depending on which patience cluster they fall into. Our results also support the idea that a simple evaluation by RBP or ERR using point estimates of parameters is not enough to understand system effectiveness; there is still too much variance in users that is not modeled by those parameter values.

We further intend to extend the models we have presented to more complex models of user behavior, such as those presented recently using summary quality [16] or user dwell time [14]. There has been much work on this in the information science and information retrieval literature, and we believe we have a novel contribution to the body of literature.

7. REFERENCES

[1] Doug Beeferman and Adam Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 407–416, 2000.

[2] Ben Carterette, Evangelos Kanoulas, and Emine Yilmaz. Simulating simple user behavior for system effectiveness evaluation. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 611–620, New York, NY, USA, 2011. ACM.

[3] Olivier Chapelle, Donald Metzler, Ya Zhang, and Pierre Grinspan. Expceted reciprocal rank for graded relevance. In *Proceedings of CIKM*, 2009.

[4] William S. Cooper. On selecting a measure of retrieval effectiveness. Part I. *Readings in Information Retrieval*, pages 191–204, 1997.

[5] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the international conference on Web search and web data mining*, WSDM '08, pages 87–94, New York, NY, USA, 2008. ACM.

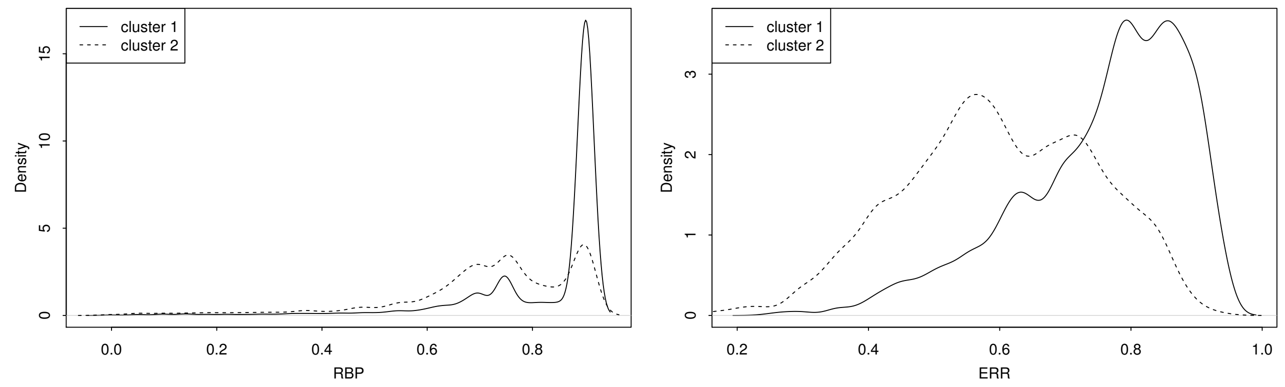

Figure 11: Left: distributions of mean RBP given the two cluster-based posterior distributions in Figure 9. Right: distributions of mean ERR given posterior distributions computed from queries in the clusters in Figure 9.

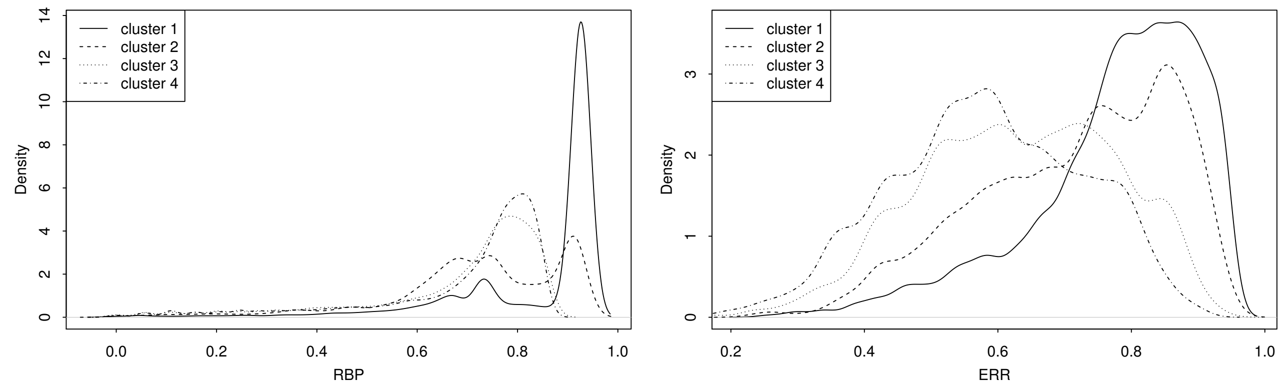

Figure 12: Left: distributions of mean RBP given the four cluster-based posterior distributions in Figure 10. Right: distributions of mean ERR given posterior distributions computed from queries in the clusters in Figure 10.

[6] Natasa Kejzar, Simona Korenjak-Cerne, and Vladimir Batagelj. Clustering of distributions: A case of patent citations. *Journal of Classification*, 28(2):156–183, 2011.

[7] Ron Kohavi, Thomas Crook, and Roger Longbotham. Online experimentation at microsoft, 2009.

[8] Lyes Limam, David Coquil, Harald Kosch, and Lionel Brunie. Extracting user interests from search query logs: A clustering approach. In *Proceedings of the 2010 Workshops on Database and Expert Systems Applications*, DEXA '10, pages 5–9, 2010.

[9] Yury Logachev, Lidia Grauer, and Pavel Serdyukov. Tuning parameters of the expected reciprocal rank. In Alain Mille, Fabien L. Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab, editors, *WWW*, pages 571–572. ACM, 2012.

[10] Alistair Moffat and Justin Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Trans. Inf. Syst.*, 27(1):1–27, 2008.

[11] Filip Radlinski, Madhu Kurup, and Thorsten Joachims. How does clickthrough data reflect retrieval quality? In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 43–52, New York, NY, USA, 2008. ACM.

[12] Stephen Robertson. A new interpretation of average precision. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 689–690, New York, NY, USA, 2008. ACM.

[13] Eldar Sadikov, Jayant Madhavan, Lu Wang, and Alon Halevy. Clustering query refinements by user intent. In *WWW 2010*, pages 841–850. Stanford InfoLab, April 2010.

[14] Mark D. Smucker and Charles L. A. Clarke. Time-based calibration of effectiveness measures. In *Proceedings of SIGIR*, 2012.

[15] Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. Query clustering using user logs. *ACM Trans. Inf. Syst.*, 20:59–81, January 2002.

[16] Emine Yilmaz, Milad Shokouhi, Nick Craswell, and Stephen Robertson. Expected browsing utility for web search evaluation. In *Proceedings of CIKM*, pages 1561–1564, 2010.

[17] Yuye Zhang, Laurence A. Park, and Alistair Moffat. Click-based evidence for decaying weight distributions in search effectiveness metrics. *Inf. Retr.*, 13:46–69, Feb 2010.

[18] Yuye Zhang, Laurence A. F. Park, and Alistair Moffat. Parameter sensitivity in rank-biased precision. In *Proceedings of ADCS*, 2008.