# TellMyRelevance! Predicting the Relevance of Web Search Results from Cursor Interactions

Maximilian Speicher[*†]
maximilian.speicher@s2013.tu-chemnitz.de

Andreas Both[*]
andreas.both@unister-gmbh.de

Martin Gaedke[†]
gaedke@informatik.tu-chemnitz.de

[*]R&D, Unister GmbH
04109 Leipzig, Germany

[†]Chemnitz University of Technology
09111 Chemnitz, Germany

## ABSTRACT

It is crucial for the success of a search-driven web application to answer users' queries in the best possible way. A common approach is to use click models for guessing the relevance of search results. However, these models are imprecise and waive valuable information one can gain from non-click user interactions. We introduce *TellMyRelevance!*—a novel automatic end-to-end pipeline for tracking cursor interactions at the client, analyzing these and learning according relevance models. Yet, the models depend on the layout of the search results page involved, which makes them difficult to evaluate and compare. Thus, we use a *Random Mouse Cursor* as an extension to our pipeline for generating layout-dependent baselines. Based on these, we can perform evaluations of real-world relevance models. A large-scale interaction log analysis showed that we can learn relevance models whose predictions compare favorably to predictions of an existing state-of-the-art click model.

## Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*relevance feedback, selection process*

## Keywords

User Interaction Tracking; Relevance Prediction; Learning to Rank

## 1. INTRODUCTION

Due to the immensely growing amount of data available on the World Wide Web, it is crucial for the success of a search-driven web application to address its users' needs in the best possible way. In particular, it must be ensured that the search results which are most relevant are displayed where they receive the highest attention. That is, mostly at the top of the first page of returned results. This is, however, not a trivial task since *a priori* we can only guess which search results a user will find relevant. The most common approach to this problem has been using different generative models for clickthrough behavior (e.g., [12, 19, 21]). These *click models* try to predict the perceived relevance of a result based on the number of clicks it has received and certain assumptions about user behavior. But since clicks might be a misleading indicator for relevance (e.g., when a clicked result is found to be useless) and web search engines tend to answering queries directly on the *search engine results page* (SERP) if possible (e.g., when searching for the local weather forecast) [8], additional user behavior must be taken into account for a more accurate prediction of search results relevance. Previous work points to mouse cursor interaction as a promising approach [16, 18], in particular because it is a reasonably good approximation for eye gaze [6, 18]. However, web search engines still seem to considerably neglect interactions beyond clicking and their advantages [16].

Good correlations between certain features of user interaction (such as the number of hovers on a search result) and explicit human judgments of search result relevance have been shown in user studies [18]. To date there is, however, no complete system available that caters for the whole process:

a) From collecting *user interaction data* and

b) *relevance judgments* to

c) providing a *ready-to-use model* for relevance prediction.

Thus, we aim at providing a *data-driven* approach for automatically generating discriminative models based on user interactions. In particular, gathering a sufficient amount of good-quality relevance judgments in real-world settings is a crucial point that needs improvement. Moreover, only few existing research has considered the impact of layout specifics on the perceived relevance of a search result. If so, this happened mainly in terms of investigating presentation biases [24] or concepts for generating an optimal layout for a given set of search results [3]. However, the layout of a SERP is an important point concerning relevance prediction since it determines how the user perceives and interacts with results.

We present *TellMyRelevance!* (TMR)—a complete end-to-end pipeline for tracking mouse cursor interactions and relevance judgments on the client and analyzing these data and learning according relevance models on the server side. Its name reflects the the two main principles of the system.


*CIKM'13,* Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

ACM 978-1-4503-2263-8/13/10 ...$15.00.
http://dx.doi.org/10.1145/2505515.2505703.

On the client side, we ask users to (implicitly) *tell* us the relevance of search results. On the server side, we try to *tell* the relevance of results using a relevance model.

We have tested our system based on a large-scale interaction log analysis with real-world data from two German hotel booking web portals. Relevance predictions for different datasets have been obtained using TMR and compared against predictions by an existing state-of-the-art click model. Results suggest that TMR is able to generate reasonably good relevance models, which can make better predictions than the existing approach.

Since the models generated by TMR depend on the specific layout of the SERP, their actual quality is difficult to evaluate and they can hardly be compared between different sites. However, these evaluations are very important for search engine owners before they can decide whether and to what extent certain models should be incorporated into their results ranking process. Therefore, we use *Random Mouse Cursor* (RMC) as an extension to TMR that makes it possible to simulate random user interactions. In this way, we increase the impact of the approach as we can generate data that provides the possibility to learn a baseline model for a specific SERP. Comparing quality measures of the baseline (e.g., root relative squared error or f-measure) to those of the real-world model enables us to make better statements about the quality of the latter one. Also, we can compare models of different SERPs since the impact of the layout on user interactions is "encoded" in their baseline models. Using this layout-independent representation leads to a better understanding of the actual search result relevance.

To summarize, this paper makes the following contributions:

- *TellMyRelevance!:* a novel end-to-end pipeline for tracking user interactions and relevance judgments and learning ready-to-use relevance models.
- Comparison of relevance predictions by TMR to predictions from an existing state-of-the-art click model.
- An analysis of real-world travel search, i.e., large amounts of user interaction data from two German hotel booking web portals, based on TMR.
- *Random Mouse Cursor:* a novel approach for generating random user interactions that can be used for learning layout-specific baseline models for layout-independent evaluation.

In the following, we will provide relevant concepts and an overview of related research in Section 2. After that, Section 3 deals with the system functionality and architecture of TMR and describes the evaluation of relevance models built using TMR based on a large-scale interaction log analysis. Since this evaluation is layout-dependent, Section 4 explains the concept and implementation of RMC as an extension to TMR and how we use it to make layout-independent statements about the previously built model. Finally, in Section 5 we discuss the limitations of our work and potential future work before giving concluding remarks in Section 6.

## 2. BACKGROUND AND RELATED WORK

The relevance of a search result can usually only be reliably determined through explicit human relevance judgments, i.e., asking a user to rate the result either absolutely or relatively compared to one or more other results. Since such judgments are difficult to obtain in sufficient numbers, Joachims [19] argues that we can *instead* use clickthrough data, i.e., which results have been clicked for a certain query, because it is present in the logs of a search engine anyway. He represents clickthrough data as triplets ($q$, $R$, $C$) with $q$ being the search query, $R$ the returned set of results and $C$ the set of clicked results. Clickthrough data cannot be used to infer the *absolute* relevance of a result since the intrinsically most relevant result might not even be displayed on the first SERP, which is in most cases the only page considered by users [19]. Rather, the data can be used for the analysis of *relative* relevances, i.e., if the result at position 3 is clicked and the two results above are not, result #3 can be assumed to be more relevant than results #1 and #2. Joachims [19] uses such relative relevances with a support vector machine to learn a "highly effective retrieval function" [19] consisting of "traditional" ranking features such as the cosine between URL-words and query.

Similar research in this direction has been mostly dealing with the prediction of relevance from clickthrough data based on models for click behavior. For example, the above assumption concerned with determining relative relevances is based on the *Cascade Model* [9]. That is, a user examines the results linearly from top to bottom and does not consider results below the clicked position. Moreover, the *examination hypothesis* says that a result must be examined and relevant to be clicked [21].

The *Dependent Click Model* [12] takes this one step further by considering multiple clicks, i.e., a result might be examined (and clicked) with a certain probability even if a preceding one was already clicked. The *User Browsing Model* [10] extends this assumption by stating that the probability of a result being examined depends on its rank as well as the distance to the latest preceding click. The underlying idea is that users will abandon a search if they encounter a long sequence of non-relevant results. Chapelle and Zhang present the *Dynamic Bayesian Network Click Model* (DBN) [5], which aims at yielding relevance estimations not affected by position bias (cf. [20]). For more precise predictions, they model the perceived relevance of search results and landing pages separately. There are numerous other papers proposing different models for click behavior (e.g., [7, 21, 23]) that will, however, not be discussed in detail here.

While we aim at providing a complete system for generating *discriminative* relevance models (i.e., data-driven) and predicting relevance, all of the above related work describes *generative* models that are not data-driven. Particularly, TMR requires training data in the form of relevance judgments while the click models aim at providing a substitute for these [19]. Still, both approaches ultimately yield predictions of the perceived relevance of search results, which can be, e.g., incorporated into existing ranking functions.

Huang [16] has found that server logs are still the primary source for information about user behavior, omitting anything happening on the client side, including mouse cursor behavior. However, he finds that these data contain valuable information for the prediction of search result relevance that "allow search engines to improve their ranking and user assistance features" [16]. Thus, in [18] they examine correlations between certain mouse features (hover rate and hover time, among others) and human relevance judgments. Besides,

they present a scalable approach for collecting data on the client side—on which we base part of our system. Their results show that positive correlations exist and how they can be used to predict search result relevance and differentiate between good and bad abandonment[1]. However, Huang et al. do not propose a complete and ready-to-use system including all relevant client-side and server-side components. In particular, they do not focus on means for automatically gathering human relevance judgments in a real-world setting and processing them accordingly.

Additionally, in [17], Huang et al. take the click models explained earlier one step further and show how an existing model (DBN [5]) can be extended by incorporating additional behavior. Their evaluation shows that the extended model is able to better predict future clicks compared to the basic click model. Contrary to our approach, this is again a generative model. In terms of interactions other than clicks, the authors consider only two additional features, i.e., hovers and scrolling. A comparison with their approach is planned as future work.

Guo et al. [14] present an approach for detecting searcher success from user interactions in mobile settings. Rather than engaging mouse features they rely on interactions such as zooming and sliding. Guo and Agichtein [13] propose another interesting approach by incorporating user interactions on the landing page into relevance prediction for the corresponding search result. This is a promising approach for determining whether a click has led the user to a good result, e.g., if the landing page is thoroughly scanned. Yet, the drawback is that a typical search engine will not have access to client-side interactions on the majority of landing pages presented unless they are enhanced with an according tracking script.

While the above related work is mainly concerned with predicting result relevance or similar from particular user interactions, little attention has been paid to the impact of a SERP's layout on the actual user behavior in this respect. There has been research on effects of presentation bias, e.g., [24], but these are more focused on attractiveness of result titles and content snippets rather than the layout surrounding the results. Moreover, Huang et al. [17] take a step in this direction by considering positions of the browser viewport and which search results were effectively viewed by the user as one basis of their user interaction model. Finally, Bozzon et al. [3] propose a solution to the automatic layout definition problem, i.e., where to place a set of heterogeneous results on a SERP based on their relevance and visibility. In contrast to our approach of generating baseline models using RMC, none of the described approaches focuses on the impact of the layout on the users' behavior. In particular, how it can be "encoded" into relevance models to establish comparability of real-world models for search-driven web applications.

# 3. TMR: FROM USER INTERACTIONS TO RELEVANCE MODELS

TMR is an end-to-end pipeline for collecting and processing large amounts of user interaction data and relevance judgments. Therefore, it comprises a number of components on the client as well as the server side (cf. Figure 1):

- The **TMR jQuery Plug-in** for tracking mouse cursor interactions and sending them to the server.
- The **TMR Raw Data Processor** for handling the data received from the client and preparing them for mouse feature extraction.
- The **TMR Mouse Features Processor** for translating the raw mouse events into features that can be used for learning a relevance model.
- The **TMR Classifier** for using mouse features and relevance judgments to train a classifier based on the WEKA[2] API [15].

The aim of the system is to take advantage of the additional information one can gain from mouse cursor interactions. We automate the whole process, from collecting according data and human relevance judgments on the client side to learning ready-to-use models for classifying a result for a given query based on its estimated relevance. In the following, we describe the functionality of each component to give an overview of the concept of the system before explaining how we used TMR for conducting a large-scale interaction log analysis on two German hotel booking web portals.

## 3.1 User Interaction Tracking

The client-side mouse tracking component of TMR, the **TMR jQuery Plug-in**, is realized as a minimally invasive jQuery plug-in that is embedded into a website (cf. Figure 1) and initialized with a set of certain options that specify, e.g., the user ID, among other things. The plug-in tracks all mouse cursor interactions that happen inside the container of a search result, i.e., when a result is entered (MOUSEENTER) or left (MOUSELEAVE), when the user clicks a link to a landing page (CLICKTHROUGH) or another link within the result (CLICK) and when the mouse cursor remained still for at least 40 ms. The latter case first triggers a "pause" event (MOUSEPAUSE) and a subsequent "start" event (MOUSESTART) when the mouse cursor moves again. This is a trade-off between granularity of events and the amount of information that has to be processed and is in accordance with a heuristic finding by [18].

Moreover, the plug-in also saves the specifics of the search session itself. That is, we determine the *anonymous user ID*[3], the *current query*, the *page number* and the *ordered list of shown results* and store these along with the timestamp of arrival at the SERP. Every mouse event triggered is then associated with the corresponding search session for further processing. For our purpose, a search session is uniquely identified by user ID, query and arrival timestamp. In particular, if a user navigates to the second results page for the same query, we consider this a new search session.

Finally, the plug-in also provides novel functionality for automatically collecting human relevance judgments. It is up to the developer to trigger implicit/explicit judgments, e.g., when the user has submitted a form or clicked a result's "thumbs up" button. This function is not restricted to SERPs, but can also be called from any other page, such as landing

[1]Abandonment means the case in which a user leaves the SERP without clicking a result. This can be either good (if the answer was presented directly on the SERP) or bad (if the user found none of the results relevant).

[2]http://www.cs.waikato.ac.nz/ml/weka/ (2013-05-16).

[3]This ID identifies a user across several pages based on a cookie, but no personal data is stored anywhere in the pipeline.

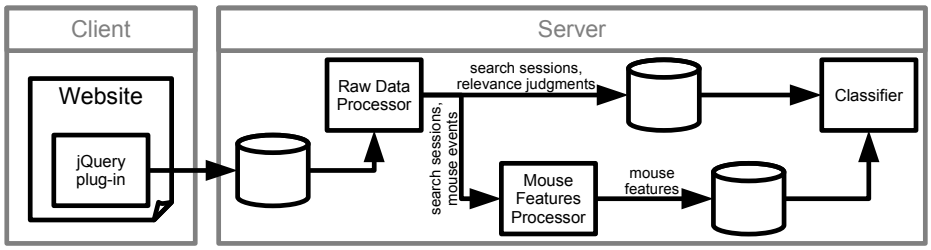


**Figure 1: The architecture and overall process flow of TMR.**

pages. Each judgment is associated with the user ID, query and result ID.

Collected data is compressed and sent to a server-side key-value store at suitable, heuristically determined intervals, where it can be accessed by TMR's Raw Data Processor. This makes the approach highly scalable, which has been proven during the large-scale interaction log analysis described later on.

## 3.2 Raw Data Processing

The **TMR Raw Data Processor** is responsible for preprocessing the raw tracking data in terms of sorting, decompressing and interpreting, so that mouse features can be extracted (cf. Figure 1). There are three possible types of packets to be processed: *search session packets*, *judgment packets* and *mouse events packets*. Particularly, the latter are split into single mouse events that are associated with the corresponding search session.

Since a judgment can be triggered on a different page than a SERP, it is also possible that users who are not associated with a search session produce a judgment packet. This can happen if, e.g., someone performs a search and then sends the link of the best result to a friend who finally triggers the judgment. Therefore, the TMR Raw Data Processor checks whether the user ID contained in a judgment packet is also associated with an already processed search session whose result list contains the corresponding result ID for which the judgment was triggered. It is up to the developer to define the "look back" time, e.g., checking only search sessions from the past two days or checking all search sessions in the database, which is a trade-off between accuracy and performance.

## 3.3 Mouse Feature Extraction

From a review of existing literature concerned with mouse cursor tracking [13, 18, 22], we derived a set of mouse features that have proven to deliver meaningful information concerning user behavior:

**Hover time** [18, 22]: The overall time the mouse cursor spent hovering the result.

**Arrival time** [18, 22]: The time elapsed from arrival at the SERP until the result is hovered for the first time.

**Clickthroughs** [18]: The number of clicks on hyperlinks that lead to the landing page for the result.

**Hovers** [18, 22]: The overall number of hovers the result has received.

**Unclicked hovers** [18]: The overall number of hovers during which no CLICKTHROUGH event happened.

**Maximum hover time** [18]: The maximum time of a single hover of the result.

**Cursor trail** [13, 22]: The overall amount of pixels the mouse cursor has traveled across the result.

**Cursor speed** [13]: *Cursor trail* divided by *cursor movement time*.

**Position** [18, 13]: The position of the result within all returned SERPs for a search query.[4]

Moreover, we included two additional features:

**Clicks** The number of clicks on hyperlinks within the result not leading to the landing page. This has been added because results involved in the conducted interaction log analysis contained links opening pop-up info boxes. These showed, e.g., images of the respective hotel. Thus, *clicks* imply a different intention/interaction than *clickthroughs*.

**Cursor movement time** The overall amount of time during which the mouse cursor has moved on the result. This has been added because we did not want to base *cursor speed* on the overall dwell time like in [13]. Otherwise, users leaving the cursor placed on a result and, e.g., leaving their computer for some time would produce unrealistically low cursor speeds.

This set of 11 features is computed by TMR's **Mouse Features Processor** on a per-result basis for each search session that is passed in (cf. Figure 1). For example, the *hover time* is computed by subtracting the timestamp of a MOUSELEAVE event from the timestamp of the corresponding MOUSEENTER event. Once all the sets of features for a search session have been computed, they are associated with the corresponding search query and result IDs. For a given query–result pair $(q, r)$, the values of the individual features $f$ are averaged over the number of hovers as follows[5] (except for features such as arrival time that obviously do not need to be averaged):

$$\text{value}'(f,q,r) = \begin{cases} \frac{\text{value}(f,q,r)}{\text{value}(\text{hovers},q,r)}, & \text{if } f \in MF_{avg} \\ \text{value}(f,q,r), & \text{otherwise} \end{cases}$$

where $MF_{avg}$ = {hover time, clickthroughs, clicks, unclicked hovers, cursor trail, cursor movement time}.

Subsequently, the averaged features for each result in a search session are saved to a database (cf. Figure 1). If there are already feature values present for the same query–result

[4]This is not a *mouse* feature but since the position of a result has shown correlations with the relevance perceived by the user in existing research (e.g., [20]), we have added it to the list of features.

[5]This is an important point to remember regarding the baseline generation using RMC in Section 4.

pair, the existing values are updated to the average values over the number of search sessions.

## 3.4 Relevance Model Generation

The last step in our pipeline is the **TMR Classifier**, which combines mouse features and relevance judgments for learning an according relevance model (cf. Figure 1). First, the sets of mouse features for all query–result pairs are fetched and extended with statements about their relevance. Thereby, the corresponding relevance judgments for a query–result pair are combined either by summing them up or using their average, depending on the developer's choice. Moreover, each mouse feature value of a query–result pair $(q, r)$ is normalized by dividing it by the respective maximum value for the corresponding query:

$$\text{value}_\text{N}(f, q, r) = \frac{\text{value}'(f, q, r)}{\max\limits_{s \in R} \{\text{value}'(f, q, s)\}} \ ,$$

$\forall f \in MF$, the set of mouse features, and with $R$ being the set of possible result IDs for the given query.

This is particularly important for compatibility with baseline data generated by RMC (cf. Section 4). Concerning the combined relevance judgment for a query–result pair, it is up to the developer whether it should also be normalized by expressing it in terms of the relative frequency for the corresponding query. A special feature of the TMR Classifier is the developer's possibility to choose whether "relevance" should be treated as a numeric attribute—i.e., based on an interval scale—or as an ordinal attribute in the resulting model. In the latter case, our system asks the developer for the desired classes and their order (e.g., bad, neutral, good) and automatically puts the combined relevance judgments into according bins.

After all of the above has been determined, the data set containing the normalized mouse features and relevance statements is either passed to one of TMR's built-in classifiers based on the WEKA API [15] or used for manual processing with WEKA. Currently, TMR supports a variety of decision trees, linear regression and an ordinal meta-classifier [11].

The whole process flow of the TMR pipeline can be seen in Figure 1.

## 3.5 Evaluation

We conducted a large-scale log analysis to evaluate the functionality and performance of the TMR pipeline. For this, we collected a huge amount of user interaction data and relevance judgments on two hotel booking portals. During collection, we engaged a novel approach to gathering implicit judgments of search result relevance based on hotel booking conversions—i.e., the user has formally agreed to book a hotel and pay for it. First, we analyze the raw correlations of mouse features with conversions. Second, we learn relevance models using TMR and obtain corresponding predictions, followed by a comparison to predictions by an existing state-of-the-art approach. Results suggest that we are able to train reasonably good models from real data that yield better results than the existing approach. The quality of a model learned with TMR is, however, difficult to evaluate since it depends on layout specifics of the considered SERP.

### 3.5.1 Method

Between December 2012 and April 2013, we collected anonymous user interaction data on two large German hotel booking web portals. The underlying assumption for this log analysis was that a completed *conversion* is a very strong indicator for the relevance of a search result, in this case a hotel presented to the user, which stands in contrast to earlier case studies. This strengthens our approach because clicks in common search engines—which do not involve payments for a concrete product—are much weaker indicators for relevance. Therefore, we considered a conversion to be a positive implicit relevance judgment for the booked hotel regarding the search query that led the user to the landing page[6]. Since the hotel booking portals did not feature a "traditional" single search input field (cf. Google, Yahoo! etc.), we considered the alphabetically sorted combination of options of the search form to be the search query. For example, a search form with the field values *Region = Italy* and *Hotel Pool = yes* would be interpreted as the query "hotelpool/yes/region/italy".

Since the data we collected contained critical information concerning the co-operating company's business model, we were not allowed to save the data to our own key-value store. Rather, all the information gathered by the TMR jQuery Plug-in was saved to a key-value store controlled by the company. From this store, we were allowed to fetch a certain fraction of data that was randomly chosen to be used for this work.

At this point in time, the two hotel booking portals involved feature the exact same template with deviations of the resulting layouts in the order of only a few pixels. Therefore, we decided to train only one relevance model that can be applied to SERPs from both websites.

### 3.5.2 Data Preparation

TMR collected a total of ∼23 GB of raw tracking data. From these, the co-operating company provided us with 29,483 randomly chosen search sessions from users that also triggered at least one judgment/conversion, i.e., they booked a hotel. This denoted a predefined fraction of all user interaction data collected. It is common for hotel booking portals that most search sessions do not lead to a conversion, which means that for the provided data *#search sessions* $\gg$ *#conversions*. Of these conversions, 36.7% were found to be invalid in the sense that there existed no related search session (cf. Section 3.2). From the provided search sessions, we extracted mouse features for 77,759 query–result pairs. For each pair $(q, r)$, to obtain its relevance $rel$, the implicit judgments (i.e., the number of conversions *conv*) were summed up and divided by the sum of all conversions for the respective query for normalization:

$$\text{rel}(q, r) = \frac{\sum\limits_{u \in U} \text{conv}(u, q, r)}{\sum\limits_{s \in R} \sum\limits_{u \in U} \text{conv}(u, q, s)} \ ,$$

where $U$ is the set of users who triggered a conversion for the given query–result pair and $R$ is the set of possible result IDs for the given query.

Due to the relatively low number of conversions, this gives us *#judged results* $\ll$ *#unjudged results*, which is not optimal for classification. In particular, despite the rather large number of search sessions collected, the diversity of issued

[6]In the following, we are going to use "conversion(s)" and "relevance" synonymously. That is, more conversions mean higher relevance and vice versa.

**Table 1: Correlations between mouse features and conversions for the individual and the combined datasets.**

| Pearson's $r$ | DS1 | DS2 | DS3 | comb. |
|---|---|---|---|---|
| avg. hover time ($a$) | 0.23 | 0.20 | 0.20 | 0.21 |
| arrival time ($r$) | 0.14 | 0.12 | 0.11 | 0.12 |
| clicks ($c$) | 0.08 | 0.08 | 0.07 | 0.08 |
| clickthroughs ($l$) | 0.35 | 0.31 | 0.34 | 0.34 |
| hovers ($h$) | 0.14 | 0.15 | 0.14 | 0.15 |
| unclicked hovers ($u$) | -0.35 | -0.31 | -0.34 | -0.34 |
| max. hover time ($m$) | 0.23 | 0.21 | 0.21 | 0.22 |
| cursor trail ($t$) | 0.12 | 0.12 | 0.13 | 0.12 |
| cursor move time ($o$) | 0.20 | 0.21 | 0.19 | 0.20 |
| cursor speed ($s$) | 0.12 | 0.15 | 0.13 | 0.13 |
| position ($p$) | -0.04 | -0.05 | -0.06 | -0.05 |
| combined | 0.39[a] | 0.36[b] | 0.38[c] | 0.38[d] |

queries was large enough so that only for a very tiny fraction of all queries, more than one hotel was booked. This results in more than 99% of the query–result pairs having a relevance of either 0.0 or 1.0. Therefore, we treat our log analysis as a binary classification problem with classes "bad" (relevance $< 0.5$) and "good" (relevance $\geq 0.5$) in order to obtain meaningful results. However, these classes are very unbalanced with >90% bad results and <10% good results.

For analysis, we have divided the query–result pairs together with their relevances (i.e., normalized conversions) into three disjunct datasets, each corresponding to 20 days of collected data. The datasets are denoted *DS1*, *DS2* and *DS3* in the following.[7]

### 3.5.3 Raw Correlations

Correlation of the 11 mouse features with conversions (see Table 1) shows very consistent results across all three datasets as well as the combination thereof. That is, although the datasets are disjunct, the average user behavior changes only very slightly w.r.t. whether a conversion is triggered or not. The most expressive feature is the number of *clickthroughs*, which is contrary to [18], where hover rate is most expressive ($r$=0.46). Moreover, we found a low positive correlation for *maximum hover time* while [18] describe a low negative correlation ($r$=-0.15) for the majority of search sessions. *Position* effectively shows no correlation with conversions, which we did not expect since position bias suggests that higher-ranked results are considered more relevant by users [20]. Also, heatmap analyses have shown that the top four results (out of 13) receive the highest attention in terms of mouse cursor interactions. However, our finding is in line with [13], who also describe a correlation of only $r$=-0.07 between a result's position and its relevance. To summarize, the raw correlations indicate that a) *clickthroughs* are the strongest single indicator for relevance, b) there are slight differences between "traditional" SERPs as investigated by, e.g., [18] and the more specific setting of travel search, and c) particularly *position* is useless as a single indicator for predicting a conversion, although position bias suggests otherwise.

[7]Much to our regret, we are not allowed to provide either the complete raw tracking data or specific information about it. Particularly, we cannot provide information about the concrete ratio of search sessions to conversions due to the fact that our data contains critical information concerning the co-operating company's business model.

[a] $y = 0.01a+0.02r+0.07c+0.11l+0.02h-0.12u+0.04m+0.01s+0.09$

[b] $y = 0.09c + 0.10l + 0.03h - 0.11u + 0.03m + 0.02o + 0.03s + 0.07$

[c] $y = 0.03a+0.02r+0.08c+0.11l+0.04h-0.11u+0.03o+0.04s+0.06$

[d] $y = 0.01a + 0.01r + 0.08c + 0.11l + 0.03h - 0.11u + 0.02m + 0.02o + 0.03s + 0.07$

**Table 2: Performance of different WEKA classifiers based on DS1 (10-fold cross-validated).**

| classifier | MCC |
|---|---|
| J48 | 0.64 |
| RandomTree | 0.76 |
| **RandomForest** | **0.81** |
| BFTree | 0.65 |
| FT | 0.54 |
| J48graft | 0.65 |
| NBTree | 0.54 |
| REPTree | 0.59 |
| SimpleCart | 0.67 |

**Table 3: Confusion matrix for the Random Forest trained with DS1 (the other datasets are omitted due to similar results).**

| | predicted class | |
|---|---|---|
| actual class ↓ | bad | good |
| bad | 23,037 | 93 |
| good | 258 | 747 |

### 3.5.4 Relevance Models Learned Using TMR

To evaluate models built with TMR, we chose to use the *Matthews correlation coefficient* (MCC), which is particularly useful for analyzing binary problems with unbalanced classes [2], along with Precision-Recall (PR) and Receiver Operating Characteristic (ROC) curves. For classification, we tested a variety of decision trees available in WEKA [15] (cf. Table 2) and found Random Forests [4] to be the best choice in terms of performance according to MCC. 10-fold cross-validation of the learned Random Forest (cf. Table 3) yielded an MCC of 0.81 for DS1, which is a reasonably good result. DS2 and DS3 reached MCC values of 0.73 and 0.75, respectively (cf. Figure 3).

### 3.5.5 Comparison to Other Approaches

To compare TMR's predictions to predictions by an existing state-of-the-art approach for estimating search result relevance, we have reimplemented the Dynamic Bayesian

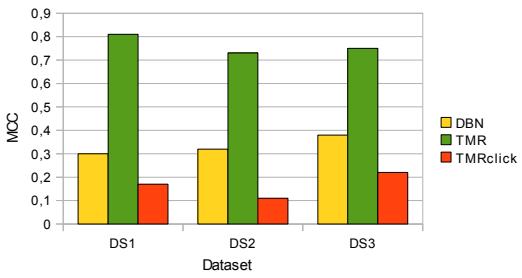


**Figure 3: Comparison of different approaches for predicting relevance.**

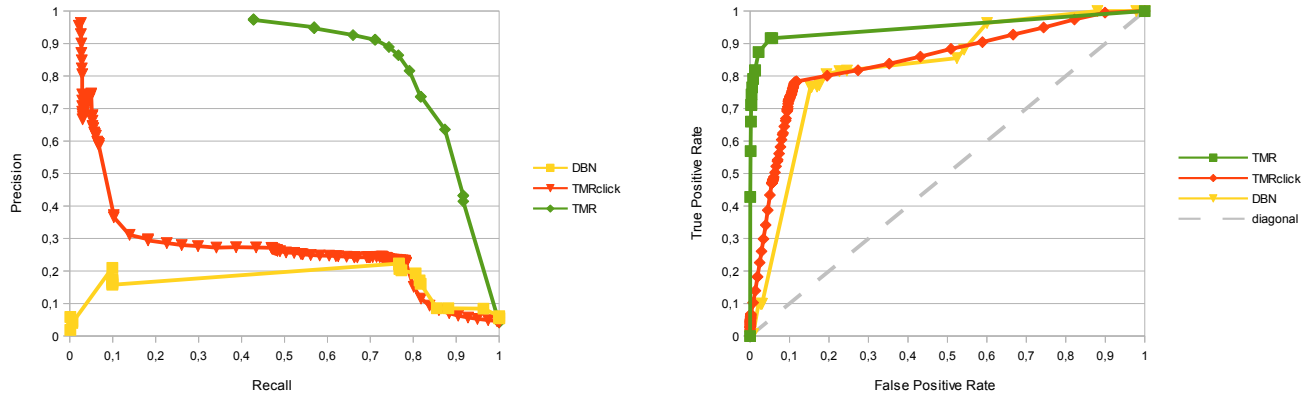


**Figure 2: PR (left) and ROC curves (right) for DS1 (the other datasets are omitted due to similar results).**

Network Click Model (DBN) by Chapelle and Zhang with $\gamma$=1 [5, Algorithm 1]. According to [17], DBN has proven its good performance and "is the most cited searcher model since the Cascade Model (which compared favorably to all models before it)". As an additional point of reference, we implemented a variation of TMR reduced to considering clickthroughs only, which we denote $\text{TMR}_{\text{click}}$. All approaches were given the same amount of information during analysis.

Analysis of the confusion matrices of the three approaches applied to each of our three datasets (see Figure 3) shows that in terms of prediction quality, TMR clearly outperforms DBN, which reaches MCC values of 0.30, 0.32 and 0.38 for DS1, DS2 and DS3, respectively. However, DBN still performs considerably better than $\text{TMR}_{\text{click}}$, which reaches values of 0.17, 0.11 and 0.22. While MCC values depend on a threshold for relevance (in this case, 0.5), the PR curves (Figure 2, left) show that TMR dominates the other approaches independent of this threshold. For example, TMR reaches its maximum precision of 0.97 at a recall of 0.43 while DBN never exceeds a precision of 0.22. This is underpinned by the ROC curves (Figure 2, right), where TMR clearly dominates as well.

These results indicate that our data-driven model enriched with additional information about user interactions can already yield better predictions than a more complex generative model relying on clickthroughs only (clearly, the discriminative clickthrough-only model performs worst). This confirms previous work by, e.g., Huang, who states that "adding additional independent data provides greater improvements than smarter algorithms" [16]. Our findings show that the information gained from user interactions other than clickthroughs—although the latter still show the highest correlation with relevance—and engaging data-driven approaches (if suitable data is available in sufficient amounts) yield great potential for improving web search.

### 3.5.6 *Discussion*

A considerable amount of users triggered a conversion without having produced a search session (36.7% of conversions were invalid, as mentioned above). This could be caused by several situations. For example, users who search for hotels but pass the result to friends/family who ultimately book the hotel, bookmarked results that are older than the user ID lifetime, or results that directly appear in external search engines due to SEM. This is negative for classification since we lose a considerable amount of valuable relevance judgments that cannot be associated with a search session.

Concerning the comparison of TMR with DBN, we expected a better performance of the latter one since it has proven to compare favorably to other well-known click models (e.g., Cascade Model or Logistic Model) [5]. We assume the relatively low performance to be mainly based on two reasons. First, as has already been shown, information from additional user behavior yields massive gains already for relatively simple models. Second, the setting of the evaluation (i.e., travel search) was rather specific compared to "traditional" web search settings in which clicks models are usually evaluated. Differences were already pointed out in Section 3.5.3. Thus, in a more general setting with higher-quality data, we would expect DBN to perform better than is expressed by MCC values of 0.30–0.38 and the PR and ROC curves. We are also aware of the fact that contrary to TMR, DBN cannot make use of the available training data since it is a generative model. Yet, in this evaluation, we have purely focused on comparing the relevance predictions of the ready-to-use models rather than the two approaches as a whole (i.e., generative vs. discriminative). All models have been provided the same data for prediction while potential training phases were not considered in the evaluation.

To summarize, in this evaluation involving real data from two hotel booking portals we found that a very huge amount of data is necessary to learn high-quality relevance models. Moreover, we need a high search session–to-conversion ratio since in this specific setting $\#\textit{search sessions} \gg \#\textit{conversions}$. Still, with our relatively low-quality data we were able to train a reasonably good model that is able to outperform an existing state-of-the-art click model [5]. We expect an increasing model quality with larger amounts of data. This is in line with a statement by Huang et al., who say that "researchers in data mining and machine translation have found that simply adding more data can result in an order of magnitude of greater improvement in the system than making incremental improvements to the processing algorithms" [17].

The training data and serialized real-world models for reproducing our results based on WEKA are available online at ⟨http://vsr.informatik.tu-chemnitz.de/demo/TMR⟩.

# 4. TMR + RMC: LAYOUT-INDEPENDENT MODEL EVALUATION

We have shown that we are able to learn reasonably good relevance models using TMR. However, these are highly dependent on the specific layout of a SERP, which affects users' interactions. For example, if the area a search result covers is larger on one SERP, the cursor trail per hover will on average be longer compared to a SERP with smaller result areas. Thus, learned models are difficult to evaluate and compare regarding their actual quality if we cannot make statements about the layout for which they were learned. These evaluations are particularly important for search engine owners before they can decide whether and to what extent certain models should be incorporated into their results ranking process.

We assume that even random movements on a SERP can show small correlations with result relevance due to certain features of a layout. Moreover, these small correlations are also contained in the real-world data since many users have a more or less random component in their cursor movements (cf. [1]). Thus, we use RMC as an extension to TMR for generating random user interactions on a SERP. That is, we model a user who is performing completely arbitrary cursor movements all over the SERP without preferring certain areas of the page. Based on the simulated data, we can use our previously described pipeline to learn a baseline model that "encodes" the SERP's layout specifics that lead to correlations already for random data. Comparing the same quality measure for the baseline model, $QM_{baseline}$, and a real-world model learned from interactions on the same SERP, $QM_{real}$, then enables us to make statements about the quality of the latter. Thereby, the *lift value* acts as a "layout-independent quality factor" ($LIQF$) that also enables comparison of a real-world model to models based on different layouts. Suitable quality measures are, e.g., f-measure, mean squared error and MCC.

$$lift = LIQF = \frac{QM_{real}}{QM_{baseline}}$$

## 4.1 Concept

RMC simulates a virtual mouse cursor on top of the TMR jQuery Plug-in. This means, a page already enhanced with TMR is prepared for RMC by just embedding an additional JavaScript file. We choose random coordinates within the current browser viewport to which the virtual cursor moves at a random speed lying in a predefined range. If the coordinates lie within a hyperlink element, the virtual cursor clicks[8]. After a random pause time from a predefined range, there is a 50% chance that new random coordinates are chosen and a 50% chance that the viewport is moved to a new random vertical offset. During the latter, the virtual cursor's position remains fixed w.r.t. the moving viewport. All actions that happen within the bounding box of search results are recorded based on the functions provided by TMR.

Since RMC can not perfectly model the average user of an arbitrary SERP (e.g., with respect to cursor speed and pause times between movements), it is important that mouse features are averaged and normalized as described in Sections 3.3 and 3.4. Otherwise, a meaningful comparison between models derived from baseline and real-world data would not be possible. The only adjustment left for the developer is how to model judgments. It is either possible to predefine relevances for each result or to let the virtual cursor trigger judgments based on the according function provided by TMR. For example, a judgment could be recorded with a certain probability if a special link is virtually clicked.

To ensure a behavior close to a real-world browser, RMC is executed based on PhantomJS[9], which is a browser-less web stack. This means, the random user interaction data can be generated instantly from the command-line. Hence, our process enables to generate a baseline automatically every time we build a real-world model and instantly use it to evaluate the latter one (cf. Figure 4).

[8]Since coordinates are chosen randomly and hovers on hyperlink elements are not treated as a mouse feature, clicks happen randomly and do not introduce biased correlations with other features.

**Table 4: Confusion matrix for classification of DS1 using the baseline model (the other datasets are omitted due to similar results).**

| | predicted class | |
|---|---|---|
| actual class ↓ | bad | good |
| bad | 21,950 | 1,180 |
| good | 671 | 334 |

## 4.2 Evaluation

The model built in Section 3.5 based on the large-scale interaction log analysis can only be evaluated and compared to other models with respect to its associated layout. Due to the lack of existing approaches, we have used RMC to generate baseline data for the SERP layout of the two hotel booking portals involved. Subsequently, we learned a model from these data and compared it to the real-world models. The results of this evaluation confirm our assumption that even completely random interactions can lead to small correlations concerning relevance and that baseline models are a convenient way for making layout-independent statements about the quality of a corresponding real-world model.

Since at this point in time, the two hotel booking portals involved in the log analysis feature the exact same template with deviations of the resulting layouts in the order of only a few pixels, it was sufficient to generate one set of baseline data for both. These data were generated based on 100 virtual search sessions with a simulated dwell time of 60 minutes each. Conversions were triggered with a probability of 50% in case the virtual cursor paused on a link to a landing page. This gave us approximately the same ratio of query–result pairs to relevance judgments as present in the real data. Subsequently, we trained a model from these simulated interactions and used it to classify DS1 (cf. Table 4), DS2 and DS3. We derived MCC values of 0.23, 0.15 and 0.20, respectively ($\mu$=0.19, $\sigma$=0.04).

As can be seen, the mean MCC of the baseline lies slightly above a value of 0.0 that we would expect for completely random data leading to no correlations. Thus, we conclude that the mean value of 0.19 is caused by certain features of the page layout. The MCC values of the real-world models exceed that of the baselines by factors of $\frac{0.81}{0.23} = 3.52$, $\frac{0.73}{0.15} = 4.87$ and $\frac{0.75}{0.20} = 3.75$ for the three datasets, which are considerable improvements. As can be seen, DS2 has the highest lift value ($= LIQF$) despite having the lowest MCC.

[9]http://phantomjs.org/ (2013-05-16).

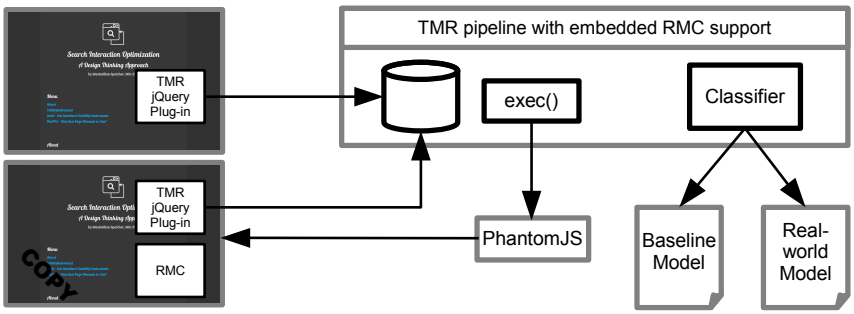


**Figure 4: Process flow of TMR with embedded RMC.**

We moreover investigated whether the real-world model for DS1 is *significantly* better than the baseline. For this, we conducted an additional experiment with 10 iterations for each model based on stratified 10-fold cross validation. That is, in each iteration, 90% of DS1 were used to train a Random Forest and the remaining 10% subset was used to test this classifier as well as the baseline model. The baseline model remained stable in each iteration since it was trained based on a different data set. The experiment resulted in a mean MCC of 0.232 ($\sigma$=0.036) for the baseline model and a mean MCC of 0.800 ($\sigma$=0.032) for the real-world model, which underpins our previous results. We applied the paired *Wilcoxon signed rank test* to the resulting MCC values (10 pairs per iteration), which showed that the difference between the two models is significant ($\alpha$=0.05, $V$=4950, $p$<0.01). Therefore, we can state that the relevance model from DS1 is significantly better than a model from random interactions by a lift value of 3.52. The other datasets are omitted at this point due to similar results.

Finally, it has to be noted that evaluation of DBN and $\text{TMR}_{\text{click}}$ using RMC does not make sense. These models are based on clickthrough data only, which is a feature not affected by layout specifics. The training data and serialized baseline model for reproducing our results based on WEKA are available online at ⟨http://vsr.informatik.tu-chemnitz.de/demo/TMR⟩.

## 5. DISCUSSION AND FUTURE WORK

In the following, we discuss current limitations of our approach and give an outlook on potential future work.

Although we were able to obtain good results regarding the relevance model for hotel booking portals, the collected data was of relatively poor quality due to unbalanced classes and query diversity. One might assume that this could be solved by collecting more data, but this alone would most probably not solve the problem. First, the ratio of query–result pairs to conversions was extremely low and will not change even if the amount of available data increases. Thus, we will always face very unbalanced classes. Second, the arrival/departure dates for a hotel booking are part of the search form and thus also part of the query. Since booking dates change with time and therefore also with the amount of collected data, this adds to the problem of query diversity (normalized relevance $\in \{0.0, 1.0\}$ for >99% of the hotels). A solution to all this rather has to involve some kind of query clustering. That is, if we aggregated queries based on a certain degree of similarity and/or ranges of dates, we could decrease query diversity while increasing the ratio of query–result pairs to conversions. This would positively affect overall data quality.

Besides, limitations in model quality could also be caused by the mouse features we selected for training. While they were thoroughly selected based on existing research, it is still possible that omitted features would add to model accuracy.

Since TMR is based on cursor input, we have to filter out search sessions produced by mobile devices before computing mouse features. This is not desirable because small-screen and touch-operated devices are constantly gaining shares of web search (cf. [14]). Therefore, we are also focusing on porting TMR into the mobile setting. This will involve the search for suitable touch-based events and more sophisticated interaction, such as multi-touch gestures. From these, we want to derive novel ranking features and establish a comparison to those based on mouse input. This will enable investigations of user interaction at a higher level of abstraction for describing web search behavior independent of input modalities.

A major shortcoming of our pipeline is the fact that it is batch-oriented. That is, raw tracking data have to be fetched from the key-value store at predefined intervals and it is at the moment not possible to learn incremental classifiers. Instead, we need to completely re-process all collected search sessions and mouse features if we want to update an already existing model. Thus, it would be desirable to have a streaming-based pipeline that works on a per–search session basis and learns a model incrementally that is automatically fed back into the ranking process. This is our current work-in-progress.

Another shortcoming of TMR is the fact that it needs training data for providing models. In our log analysis, this data was available as implicit relevance judgments in the form of conversions. Yet, in a general web search setting, such indications of relevance are less common. Thus, TMR has disadvantages compared to generative click models. We aim at conducting a second large-scale interaction log analysis for engaging functionality of TMR that could not be tested with the hotel booking portals. Particularly, this applies to collecting explicit judgments through, e.g., thumbs-up/thumbs-down icons. The log analysis shall be based on a search-driven web application that is different from hotel booking portals in terms of, e.g., a single search input field and a larger variety of result types besides hotels. From this, we expect deeper insights into how TMR can be applied to more general settings.

Finally, a comparison to the extended DBN model by Huang et al. [17] is intended. While a re-implementation of the approach is possible, it is time-consuming and also

requires larger adjustments to TMR. Thus, the comparison is currently planned as future work.

## 6. CONCLUSIONS

This paper presented TMR, which is a new automatic end-to-end pipeline for collecting user interaction data and relevance judgments on SERPs and learning ready-to-use relevance models from these. Yet, the learned models are difficult to evaluate and compare regarding their actual quality—which is particularly important for search engine owners—without taking into account the layout specifics of the corresponding SERPs. Due to the lack of existing approaches, we make use of RMC, a new approach to learning baseline models based on random user interactions simulated for a certain SERP. Using such a baseline that "encodes" the involved layout, we can determine a layout-independent quality factor for real-world models. We conducted a large-scale interaction log analysis involving a novel way of collecting implicit relevance judgments based on hotel booking conversions. Results show that TMR, not only provides reasonably good relevance models but also relevance predictions that compare favorably to predictions by DBN, an existing state-of-the-art click model. Thus, already relatively simple models can gain massively from considering user interactions other than clickthroughs. Search engines as well as the research community could highly benefit by leveraging these additional information using TMR. Furthermore, we used RMC to generate a baseline model for the SERP layout involved, which shows that even random interactions lead to small correlations. Based on this, RMC is a suitable approach to layout-independent evaluation of relevance models. Future work will include research on how to gain user interactions of better quality from real-world settings, transforming TMR into a streaming-based approach, and a second large-scale interaction log analysis in a more general web search setting.

## 7. ACKNOWLEDGMENTS

We would like to thank Martin Gleditzsch, Michael Röder, Ricardo Usbeck, Christiane Lemke, Steffen Becker and the attendees of the Unister Friday PhD Symposia for their valuable feedback. This work has been supported by the ESF and the Free State of Saxony.


## 8. REFERENCES

[1] E. Arroyo, T. Selker, and W. Wei. Usability Tool for Analysis of Web Designs Using Mouse Tracks. In *CHI Extended Abstracts*, 2006.

[2] P. Baldi, S. Brunak, Y. Chauvin, C. A. Andersen, and H. Nielsen. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, 16(5), 2000.

[3] A. Bozzon, M. Brambilla, and S. Comai. A Characterization of the Layout Definition Problem for Web Search Results. In *OTM Workshops*, 2010.

[4] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[5] O. Chapelle and Y. Zhang. A Dynamic Bayesian Network Click Model for Web Search Ranking. In *Proc. WWW*, 2009.

[6] M. C. Chen, J. R. Anderson, and M. H. Sohn. What Can a Mouse Cursor Tell Us More? Correlation of Eye/Mouse Movements on Web Browsing. In *CHI Extended Abstracts*, 2001.

[7] W. Chen, Z. Ji, S. Shen, and Q. Yang. A Whole Page Click Model to Better Interpret Search Engine Click Data. In *Proc. AAAI*, 2011.

[8] L. B. Chilton and J. Teevan. Addressing People's Information Needs Directly in a Web Search Result Page. In *Proc. WWW*, 2011.

[9] N. Craswell, O. Zoeter, M. Tylor, and B. Ramsey. An Experimental Comparison of Click Position-Bias Models. In *Proc. WSDM*, 2008.

[10] G. E. Dupret and B. Piwowarski. A User Browsing Model to Predict Search Engine Click Data from Past Observations. In *Proc. SIGIR*, 2008.

[11] E. Frank and M. Hall. A Simple Approach to Ordinal Classification. In *Proc. ECML*, 2001.

[12] F. Guo, C. Liu, and Y. M. Wang. Efficient Multiple-Click Models in Web Search. In *Proc. WSDM*, 2009.

[13] Q. Guo and E. Agichtein. Beyond Dwell Time: Estimating Document Relevance from Cursor Movements and other Post-click Searcher Behavior. In *Proc. WWW*, 2012.

[14] Q. Guo, S. Yuan, and E. Agichtein. Detecting Success in Mobile Search from Interaction. In *Proc. SIGIR*, 2011.

[15] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.*, 11(1), 2009.

[16] J. Huang. On the Value of Page-Level Interactions in Web Search. In *Proc. HCIR*, 2011.

[17] J. Huang, R. W. White, G. Buscher, and K. Wang. Improving Searcher Models Using Mouse Cursor Activity. In *Proc. SIGIR*, 2012.

[18] J. Huang, R. W. White, and S. Dumais. No Clicks, No Problem: Using Cursor Movements to Understand and Improve Search. In *Proc. CHI*, 2011.

[19] T. Joachims. Optimizing Search Engines using Clickthrough Data. In *Proc. KDD*, 2002.

[20] D. Lagun and E. Agichtein. ViewSer: Enabling Large-Scale Remote User Studies of Web Search Examination and Interaction. In *Proc. SIGIR*, 2011.

[21] C. Liu, F. Guo, and C. Faloutsos. BBM: Bayesian Browsing Model from Petabyte-scale Data. In *Proc. KDD*, 2009.

[22] V. Navalpakkam and E. F. Churchill. MouseTracking: Measuring and Predicting Users' Experience of Web-based Content. In *Proc. CHI*, 2012.

[23] R. Srikant, S. Basu, N. Wang, and D. Pregibon. User Browsing Models: Relevance versus Examination. In *Proc. KDD*, 2010.

[24] Y. Yue, R. Patel, and H. Roehrig. Beyond Position Bias: Examining Result Attractiveness as a Source of Presentation Bias in Clickthrough Data. In *Proc. WWW*, 2010.