# Search Result Prefetching Using Cursor Movement

Fernando Diaz[*]
fdiaz@microsoft.com
Microsoft Research

Qi Guo[†]
qiguo@google.com
Google

Ryen W. White
ryenw@microsoft.com
Microsoft Research

## ABSTRACT

Search result examination is an important part of searching. High page load latency for landing pages (clicked results) can reduce the efficiency of the search process. Proactively prefetching landing pages in advance of clickthrough can save searchers valuable time. However, prefetching consumes resources that are wasted unless the prefetched results are requested by searchers. Balancing the costs in prefetching particular results against the benefits in reduced latency to searchers represents the search result prefetching challenge. We present methods that leverage searchers' cursor movements on search result pages in real time to dynamically estimate the result that searchers will request next. We demonstrate through large-scale log analysis that our approach significantly outperforms three strong baselines that prefetch results based on (i) the search engine result ranking, (ii) past clicks from all searchers for the query, or (iii) past clicks from the current searcher for the query. Our promising findings have implications for the design of search support that makes the search process more efficient.

## 1. INTRODUCTION

Search result selection is a core part of the Web search experience. Following the generation of a search engine result page (SERP) comprised of candidate results, searchers must select results of interest and wait for them to load. While the latency in SERP generation has been well studied and shown to impact measures of the search experience (e.g., higher SERP generation latency leads to higher dissatisfaction and reduced SERP engagement) [2, 38], the relationship between latency and the loading of landing pages is less well understood. Despite a lack of research on this topic in the Web search community, general research on latencies in interactions with computer systems [39] and interactions with Web pages in particular [32] suggests that it has a significant impact on the overall experience. Methods to proactively fetch the content of particular search results before they are selected could benefit searchers, while balancing the costs involved in downloading content that is never viewed; we define this as the *search result prefetching challenge.*

[*]Author order alphabetical.

[†]Work conducted while at Microsoft.

*SIGIR '16, July 17-21, 2016, Pisa, Italy*
© 2016 ACM. ISBN 978-1-4503-4069-4/16/07...$15.00
DOI: http://dx.doi.org/10.1145/2911451.2911516

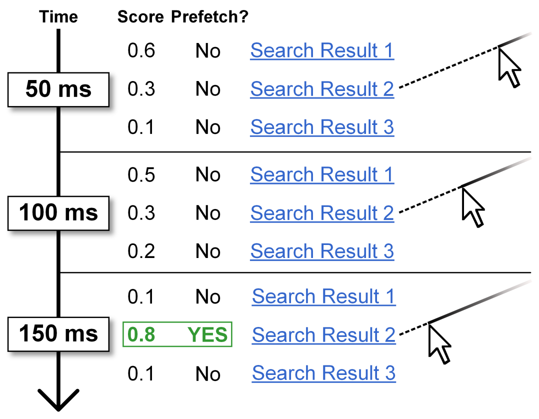


**Figure 1:** Example of cursor-based prefetching. The model estimate of whether the searcher will select one of the top three results over time (at 50ms, 100ms, 150ms) is shown alongside each link (as *Score*). These estimates change as a function of the searcher's cursor movements. The cursor trail and the trajectory towards the second result are highlighted in the figure with solid and dashed lines respectively. When the model score for a result reaches a certain threshold (0.8 for Search Result 2), that result is prefetched.

To address this challenge, the prefetching system needs to predict which result the user will select next. In previous research, such predictions are usually made via static estimates learned from historic usage data [13, 14, 22, 26, 45]. For example, search engines prefetch the top result for queries where there is little variation in intent (primarily navigational queries [1]). However, these methods are only applied for small sets of queries where the dominant intent is clearly defined and observable via prior click patterns.

In contrast, we propose to address this challenge by dynamically updating the estimates of likelihood of each result being selected during interaction with a SERP; when the system is confident that a link will be selected, the page is prefetched (Figure 1). Our method leverages the mouse cursor movements on the SERP (which can now be collected in a scalable manner [8, 20]), contextualized by the query, the results, and the searcher's historic activity to make *real-time predictions.* If we can correctly prefetch early enough, we can save people significant time–this is especially important for those accessing the Web on low-bandwidth connections, who may need to be more selective about the results that they view. Indeed, on average, our methods save searchers 650ms per query for the 65% of queries where we correctly prefetch the clicked result.

We make the following contributions with our research:

- Introduce the search result prefetching challenge and estimate the scope of its potential impact on searchers and their search experience (in terms of the fraction of query volume and average time savings per query of our method);
- Develop machine-learned models that leverage a rich array of features to prefetch search results pre-click;
- Experiment with large-scale logs and demonstrate gains over strong baselines, including variations for different query types. We also identify the important feature classes in the learned models via ablation experiments, and;
- Present implications of result prefetching for search system design and for society (e.g., enabling more rapid information access for those on slower Internet connections).

The remainder of this paper is structured as follows. Section 2 describes related work in areas such as prefetching, the impact of response latency on user engagement, and mouse cursor monitoring. Section 3 motivates our research by demonstrating the potential impact of our prefetching system. Section 4 presents a formal description of the search result prefetching challenge. Section 5 describes the prefetching approach, including the data used, the features generated, and the models that result. In Section 6, we describe our experimental setup, including datasets and evaluation. In Section 7, we present the results of our experiments, demonstrating the effectiveness of our method compared to baselines. Section 8 discusses our findings, their limitations given the data and the problem setting, and their implications for the design of search systems. We conclude in Section 9 and present opportunities for future work.

## 2. RELATED WORK

The following areas are relevant: (a) latency effects in human-computer interaction and SERP generation, (b) methods to reduce latency during search, and (c) using cursor movements collected in laboratory settings and at scale.

### 2.1 Latency Effects

Since the early days of human computer interaction, researchers have studied the influence of system response time on the success, speed, and satisfaction of interactions [31, Chapter 5]. These studies have found that rapid responses (i.e., less than a second) are preferred and can increase user productivity [31]. Shneiderman [39] reviewed the literature on computer response time and recommended that computers should respond immediately, in part based on limitations in human short-term memory [27]. Web page download time is affected by factors such as browser performance, Internet connection speed, local network traffic, load on the remote host, and the structure and format of requested content. Download speeds are an important aspect of the user experience [32]. Studies have examined the interplay between page load time and the user experience, including tolerable wait times [30] and their psychological impact [36].

Intensive research and engineering efforts have been devoted to achieving low latency in large, complex computing systems such as search engines [10]. Modern Web search engines deliver results rapidly because fast results are perceived as being of higher quality and searchers engage more with them. For example, Google conducted online experiments where they intentionally injected server-side delays, ranging from 100 to 400 milliseconds, into the search results to observe changes to people's behavior. They found that increasing the load time of the SERP by as little as 100 milliseconds decreased the number of searches per person. These differences increased over time and persisted even after the experiment ended [38]. In similar experiments, Bing added server delays ranging from 50 to 2000 milliseconds. They observed decreases in queries and clicks, and an increase in time to click, with larger effects with more delay [38]. Arapakis et al. employed user studies and large-scale log analysis, and showed that response latencies of 500ms were noticeable by searchers [2]. Recognizing the importance of speed to searchers, Google added site speed (i.e., how quickly a Website responds to requests) as a relevance signal in search ranking [40]. Recent work on slow search discusses the cost-benefit tradeoffs in retrieving search results quickly, and what could be accomplished given more time [41]. Beyond just making search faster (or slower), there are other reasons why page load latency matters, e.g., to address network bandwidth constraints [14].

### 2.2 Reducing Latency

To address latency in query responses, researchers have designed caches to rapidly serve results [5], including ways to leverage historic search behavior [13, 22, 26]. These methods limit the set of documents searched in response to queries, incurring increased infrastructure costs. Search engines already try to reduce time to click by promoting popular results [1]. Since repeat visits to the same result is common [43], search engines have promoted results that are likely to be selected; reducing the time for searchers to locate these results on SERPs. The Bing search engine already uses new browser capabilities to prefetch results that are highly likely to be selected [35]. This can enhance the search experience by reducing latency beyond SERP creation.

Moving beyond search engine support for latency reduction, Padbanabham and Mogul [33] use N-hop Markov models based on surfing patterns for improving pre-fetching strategies for Web caches. Fan *et al.* [14] leverage a user's historic surfing activity and a Markov predictor tree to predict future page accesses. Yang et al. [45] mined frequent sequences from Website access logs and employed them to derive association rules that could be used in prefetching decisions. More recently, White et al. [44] used recent search interactions and other contextual signals (e.g., incoming hyperlinks) to predict searchers' future topical interests given a Web page. Research on *continual computation* [18] proposed decision-theoretic methods for the ideal use of idle time for computational problem solving. These methods have been applied for selective (utility-directed) content prefetching, including the partial prefetching of specific Web page elements, while balancing associated costs and benefits.

To be successful, all of these methods rely on popularity data and/or the sequence of visited Web pages. This may be reasonable for popular queries/pages or active users, but less applicable for other scenarios with less data (e.g., tail queries). The methods that leverage browsing data relies on being able to track sequences of Web page visits, which can require significant infrastructure, especially at scale across many Websites. It is also not clear the extent to which such an approach applies in search scenarios, where the results returned for a query are dynamically generated and the set of prefetching candidates may change over time [6].

## 2.3 Leveraging Cursor Movements

We are focused on dynamically prefetching search result content once a query has been issued. Retrieving the top result for all queries or all results for all queries incurs a high cost in terms of resource consumption (primarily network bandwidth but also battery power on mobile devices). We hypothesize that by monitoring searcher attention in real-time on SERPs we can perform better prefetching than these crude methods. Lee et al. [25] performed prefetching of video materials via models learned from eye-gaze and cursor movements. They did so in a laboratory setting with access to gaze tracking technology. Recent research has shown that there is a strong association between gaze and cursor position [16, 20], especially around the time of a click [19]. This can be used to predict attention and intentions from cursor movement data [16, 19] in laboratory environments. Beyond controlled settings, recent work has shown that such methods can be deployed at scale online [8, 20]. This facilitates a better understanding of search behavior [8], and enables predictions of SERP examination activity [11], improved relevance estimation [19], and ranking [23] based on common patterns in cursor signals. From this data we learn models to predict which results will be clicked *in real time*, and evaluate our models in a natural setting.

Human-computer interaction (HCI) researchers have predicted aspects of cursor movement termination, focusing on endpoint prediction (i.e., predicting the terminal location of the cursor) and target prediction (i.e., deciding between multiple targets). Within HCI, these have traditionally been used in applications to expedite the acquisition of targets.

Simple versions of target prediction leverages distance from the mouse cursor [24] or consider angles between the movement vector and vectors for the target positions [29]. More sophisticated methods build probabilistic models based on previous clicks [46], or using kinematic template matching [34]. Although there are similarities between this work and ours, experiments in this area are generally conducted in carefully-controlled, artificial environments. On SERPs, there are many results to choose from, there are many aspects of the page competing for searcher attention in addition to the results (e.g., advertisements, related searches), and there are preconceived biases that affect where people click in the result list, irrespective of content (e.g., positional biases [21]). All of these factors make the task of real-time click prediction on SERPs quite challenging, especially if searchers' clicks disagree with the query's aggregate click distribution.

## 2.4 Contributions Over Previous Work

Our research extends previous work in a number of ways. First, we focus on prefetching in dynamic environments where the list of targets can change over time, even for the same query. Most prefetching methods rely on static predictions and transition probabilities learned from historic data. Second, many of the studies of end-point prediction focus on carefully controlled studies in laboratory settings. In contrast, we operate in a non-controlled environment with cursor movements with multiple targets, potential distractions, and biases that can affect behaviors irrespective of the relevance of the results retrieved by the search engine. Third, by leveraging cursor movements and other features our approach can better adapt to the current search situation and less common informational queries. Many current prefetching methods within Websites and search engines focus only on providing this support for popular pages. Finally, since we propose a model that is learned offline from many searchers' behavior, there is no need for searchers to calibrate the model to accommodate their search activity before the first use (as is the case in other models, e.g., [4]).

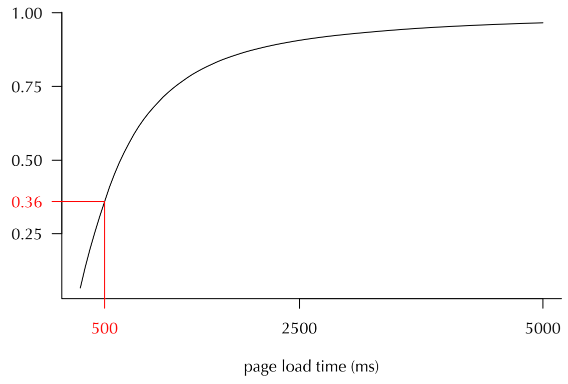


**Figure 2:** Cumulative distribution function of landing page load times for result clicks on SERPs.

## 3. MOTIVATION

Before proceeding, it is important to quantify the potential gains from prefetching in a Web search setting. If all landing pages were to load instantly then there would be no benefit from prefetching. Obviously this is not the case given the computational differences in the machines serving and accessing online content, and the network transport pipeline. To better understand the potential of prefetching in Web search settings we analyzed six months of search-click logs from millions of consenting users of Internet Explorer in 2013. From these data, we obtained a distribution of page load times for hyperlink clicks on SERPs from a popular Web search engine. The mean and median landing page load times (PLTs) were 1282ms and 672ms respectively. The empirical cumulative distribution of PLTs from 200ms to five seconds or more is shown in Figure 2. Given that searchers start to notice PLT delays at 500ms [2] (or even earlier [3]), the figure suggests that, if we could make an accurate prediction about which results to prefetch, we could noticeably improve the search experience for at least 64% of SERP clicks (as highlighted in red in Figure 2). These findings serve as a strong motivation for developing the prefetching model introduced and evaluated in this paper.

## 4. PROBLEM DEFINITION

Let $\mathcal{U}$ be a set of algorithmic results. Each result is associated with a specific SERP region or *area of interest* (AOI), comprising the region spanning the result title, snippet, and URL. Given that the SERP was presented to the searcher at time 0 and the searcher clicks on $u^* \in \mathcal{U}$ at time T, we would like to fetch $u^*$ at some point before T. Decision-making is online: starting at time 0, the system observes a sequence of interactions (cursor movements in our case) which might inform its decision-making. Once a decision to fetch has been made, the system may not fetch another page until the searcher clicks on $u^*$. This budget represents a conservative estimate of the cost, in terms of time and bandwidth, of prefetching the result, and considers all clicks equally, e.g., for simplicity the size of the page is not considered in the evaluation, even though prefetching a large/media-rich document costs more than prefetching a smaller resource.

## 5. ALGORITHMS

We regard the search result prefetching challenge as a ranking problem. We model probability of clicking on a result as a function of static features of the result (e.g., position) as well as dynamic features of the result (e.g., proximity of the cursor to the result AOI). Our model computes this relevance score throughout the searcher's interaction with the SERP. If the score exceeds a threshold $\tau$, then the landing page content is fetched. In this section, we will describe the features and model that we developed for this challenge.

### 5.1 Features

Our features can be divided into *static* and *dynamic* based on whether they are the same across all cursor movements for a query impression (static) or change as the searcher moves the cursor (dynamic). Each group can be further divided into *global* and *local*. Global features of the SERP are the same over all AOIs (e.g., SERP includes an advertisement). Local features refer to properties unique to each result AOI (e.g., Euclidean distance between cursor and AOI). Features are computed every time the cursor positon is sampled (i.e., after 250ms have passed or the cursor has moved at least eight pixels) for each of the result AOIs, since the goal is to predict the result AOI that will be selected.

Previous research has shown that modeling differences with the normative behavior for each searcher can help better estimate document relevance [17]. As such, we include the normalized version for each feature for each searcher using the deviations from average for the $\langle \text{searcher}, \text{feature} \rangle$ pair. User deviation is defined as the difference between the feature value at the current cursor position and the the average feature value for that searcher computed over all their historic actions. User deviation variants are included for *Dynamic Global* and *Dynamic Local* features (Table 1).

#### 5.1.1 Static Global Features

*Query Features* comprise query frequency and query click entropy (as defined in [12]). Searchers may interact differently for different query classes, e.g., navigational queries vs. informational queries, as shown previously [9].

*SERP Features* capture the presence of non-target elements such as advertisements and related searches, which may influence the click behavior on the target organic search results. Binary features for SERP has advertisements and SERP has related searches are included.

#### 5.1.2 Dynamic Global Features

*Global Cursor Features* capture the dynamics of cursor movements up to the cursor sample in question. Features include velocity, acceleration, jerk (i.e., rate of acceleration change) and changes from previous cursor sample. These also include features that capture where the cursor was located on the SERP, such as its horizontal and vertical coordinates, the maximum vertical coordinate reached by the

**Table 1:** Features used in prefetching model. Coordinates are relative to upper-left corner of the SERP. Distances, coordinates, and areas measured in pixels. Features with maxima or totals (denoted *) are computed since SERP load.

| Feature name | Description |
|---|---|
| Static Global | |
| Query frequency | Frequency of the query |
| Query click entropy | Result click entropy of the query (as in [12]) |
| SERP has advertisements | SERP has advertisement(s)? (binary) |
| SERP has related searches | SERP has related searches? (binary) |
| Dynamic Global | |
| Cursor xcoord | X-coordinate of cursor |
| Cursor ycoord | Y-coordinate of cursor |
| Cursor delta | Distance cursor moved since it was last sampled |
| Num non hyperlink clicks | Number of cursor clicks on SERP that are not on hyperlinks, e.g., for text selections |
| Cursor is reading | Reading behavior (i.e., following text with cursor [37]) |
| Cursor max ycoord* | Max y coordinate of cursor |
| Cursor max AOI rank* | Max AOI rank of cursor |
| Cursor total distance* | Total cursor move distance |
| Cursor total time* | Total time moved |
| Cursor time delta | Time since cursor position was last sampled |
| Static Local | |
| AOI area | Area of result AOI |
| AOI width | Width of result AOI |
| AOI height | Height of result AOI |
| AOI rank | Rank position of result AOI |
| AOI xcoord | X-coordinate of result AOI |
| AOI ycoord | Y-coordinate of result AOI |
| AOI has card | AOI has special image (e.g., a brand logo) and/or additional result information such as deep links? (binary) |
| AOI has answer | AOI has answer? (binary) |
| Dynamic Local | |
| AOI is visible | AOI in viewport? (binary) |
| AOICursor hover | Cursor over AOI? (binary) |
| AOICursor distance | Euclidean distance from the cursor position to the center of the AOI |
| AOICursor angle | Angle between direction vector center of result AOI. AOI with least deviation receives 1, otherwise 0 |
| AOICursor proximity | Proximity changes of cursor with respect to AOI. 2 = moving away from AOI, 1 = moving toward AOI, 0 = same distance from AOI |
| AOICursor speed | Speed of cursor movement toward AOI |
| AOICursor acceleration | Acceleration toward AOI |
| AOICursor jerk | Rate of cursor acceleration change toward AOI |
| AOICursor on target | On track to visit AOI. Project least squares line through last five cursor movements. Return 1 if it intersects the AOI, otherwise 0 |
| AOICursor xdistance | Horizontal distance between cursor and AOI |
| AOICursor ydistance | Vertical distance between cursor and AOI |
| AOICursor dwell | Cursor dwell time in AOI |
| AOICursor title dwell | Cursor dwell time in result title of AOI |

cursor and maximum AOI rank that the cursor is observed passing over, and the number of non-hyperlink clicks the impression has received up to the cursor sample. The rationale for these features is to capture the different stages of the cursor movements. A directed, rapid movement may mean that the searcher has found content of potential value; slow, undirected movements may suggest that they are still searching. Determining the maximum vertical position of the cursor offers insight into the number of search results considered. Finally, to explicitly model reading behavior using cursor as an aid [37] (which could be an indicator of interest in a landing page), a binary feature captures whether two consecutive left-right cursor movements are observed.

#### 5.1.3 Static Local Features

*AOI Features* characterize rhe target AOIs (in our case, each search result). For each AOI, specific features include the rank position of the AOI, horizontal and vertical coordinates, width, height, area of the AOI, and the type of the AOI (e.g., whether AOI is an aggregated result). The rationale here is that behavior is highly influenced by the presentation of the AOIs, for example, higher ranked AOIs may be more likely to receive clicks while bigger AOIs and AOIs with cards (e.g., brand logos) may be more likely to attract searcher attention and result in clicks [11].

#### 5.1.4 Dynamic Local Features

*Local Cursor Features* capture the interaction between cursor movements and each AOI, in particular, to capture if the cursor is moving towards the AOI for a potential click. Features include the overall, vertical, horizontal distances, between the AOI and the cursor, the angle between the moving direction of the cursor and the AOI, and, in turn, the proximity between the two (2 = moving away from AOI, 1 = moving toward AOI, 0 = same distance from AOI), as well as whether the AOI is on target per the cursor trajectory (i.e., draw a least squares line through last five cursor movements and return true/false on whether it intersects each AOI). We also compute the dwell time the cursor hover on the AOI and the title of the AOI, respectively, as they may be strong indicators of a potential click on the AOI.

*View Features* capture whether the AOI is visible in the viewport. Invisible AOIs cannot be selected by searchers.

### 5.2 Model

We train a regression model to predict which result will be clicked. The training procedure builds an ensemble of decision trees based on gradient boosting [7]. This technique has been shown to provide state-of-the-art performance for various applications. In the context of learning to rank, we treat each cursor sample as a query and each $u$ as a document. The objective of the model is to predict $u^*$. At test time, for each cursor sample, we featurize and score each $u \in \mathcal{U}$. If the score of one or more results is above $\tau$, we fetch $u$ with the highest score.

## 6. METHODS

We now define the methods employed in training and evaluating our prefetching models, starting with the log data.

### 6.1 Data

To record searcher interactions with the Bing SERP at scale without the need to install any browser plugins, we used an efficient and scalable approach [8]. JavaScript-based logging functions were embedded into the HTML source code of the SERP. To obtain a detailed understanding of user interactions with the SERP, we recorded information on mouse cursor movements, clicks, scrolling, text selection events, focus gain and loss events of the browser window, as well as bounding boxes of several AOIs on the SERP and the browser's viewport size. We optimize our implementation and run large-scale live experiments to ensure no significant delay in PLT for SERPs with this logging enabled.

The JavaScript function for logging mouse cursor positions checked the cursor's horizontal and vertical coordinates relative to the top-left corner of the SERP every 250 milliseconds. Whenever the cursor moved more than eight pixels away from its previously logged position, its new coordinates were sent to the remote Web server. Eight pixels correspond to approximately the height of half a line of text on the SERP. We used this approach rather than recording every cursor movement since we wanted to minimize the data gathered and transmitted so as to not adversely affect the user experience with delays associated with log data capture and data uploads to the remote server. Since cursor tracking was relative to the document, we captured cursor alignment to SERP content regardless of how the user reached that position (e.g., by scrolling or keyboard).

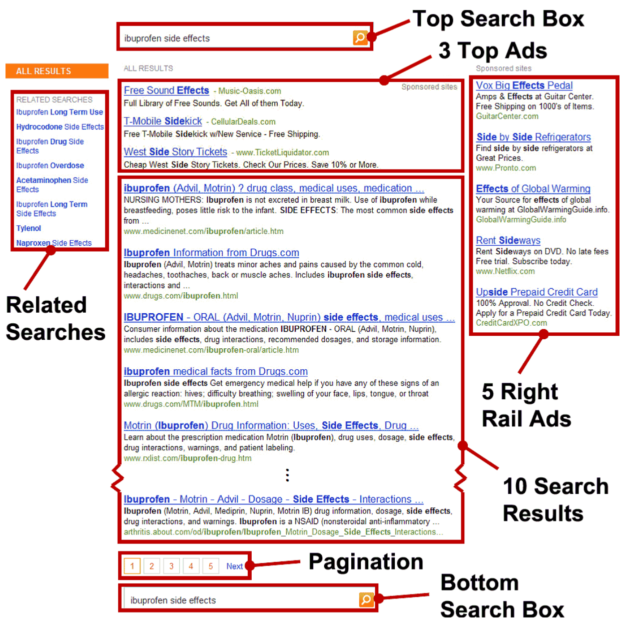
Mouse clicks were recorded using the JavaScript `onMouseDown` event handling method. The backend server received log entries with location coordinates for every mouse click, including clicks that occurred on a hyperlink as well as those that occurred elsewhere on the SERP (even on white space containing no content). To identify clicks on hyperlinks and differentiate them from clicks on inactive page elements, we logged unique hyperlink identifiers embedded in the SERP.

The width and height of the browser viewport in pixels at SERP load time were also logged. This told us which AOIs were visible. Browser window resizing during SERP interaction was not accounted for. We also recorded the current scroll position, i.e., the vertical coordinate of the uppermost visible pixel of the SERP in the browser viewport. This coordinate was checked three times per second and was recorded whenever it had changed by more than 40 pixels compared to the last logged scrolling position. This corresponds to approximately the height of two lines of text.

Simply logging the text of what was displayed on the SERP is insufficient for reconstructing its layout since SERPs vary per query (depending on whether vertical results are shown, etc.), font sizes, and other browser preferences.

To reconstruct the exact SERP layout as it was rendered in the user's browser, we recorded the positions and sizes of AOIs. We use the method from [8] to identify and record the exact position of AOIs on SERP loading. The specific AOIs recorded were: (i) top and bottom search boxes, (ii) left rail and its contained related searches, search history, and query refinement areas, (iii) mainline results area and its contained result entries, including advertisements and answers, and (iv) right rail. Some of these AOIs are visualized in Figure 3. For each AOI bounding box, we determined and logged the coordinates of its upper left corner as well as its width and height in pixels. Using this information, we map cursor positions and clicks to AOIs. Although we record the position all AOIs shown in Figure 3, we focus only on predicting clicks on one of the ten result AOIs.

The final, processed dataset consists of batches of instances



ibuprofen side effects

ALL RESULTS

RELATED SEARCHES
Ibuprofen **Long Term Use**
**Hydrocodone** Side Effects
Ibuprofen **Drug** Side Effects
Ibuprofen **Overdose**
**Acetaminophen** Side Effects
Ibuprofen **Long Term** Side Effects
**Tylenol**
**Naproxen** Side Effects

ALL RESULTS

Sponsored sites

Free Sound **Effects** - Music-Oasis.com
Full Library of Free Sounds. Get All of them Today.

T-Mobile **Side**kick - CellularDeals.com
Free T-Mobile **Side**kick w/New Service - Free Shipping.

West **Side** Story Tickets - www.TicketLiquidator.com
Cheap West **Side** Story Tickets. Check Our Prices. Save 10% or More.

**ibuprofen** (Advil, Motrin) ? drug class, medical uses, medication ...
NURSING MOTHERS: **Ibuprofen** is not excreted in breast milk. Use of **ibuprofen** while breastfeeding, poses little risk to the infant. **SIDE EFFECTS**: The most common **side effects** from ...
www.medicinenet.com/**ibuprofen**/article.htm

**Ibuprofen** Information from Drugs.com
**Ibuprofen** (Advil, Motrin) treats minor aches and pains caused by the common cold, headaches, toothaches, back or muscle aches. Includes **ibuprofen side effects**, interactions and ...
www.drugs.com/**ibuprofen**.html

**IBUPROFEN** - ORAL (Advil, Motrin, Nuprin) **side effects**, medical uses ...
Consumer information about the medication **IBUPROFEN** - ORAL (Advil, Motrin, Nuprin), includes **side effects**, drug interactions, recommended dosages, and storage information.
www.medicinenet.com/**ibuprofen**-oral/article.htm

**ibuprofen** medical facts from Drugs.com
**Ibuprofen side effects** Get emergency medical help if you have any of these signs of an allergic reaction: hives; difficulty breathing; swelling of your face, lips, tongue, or throat
www.drugs.com/MTM/**ibuprofen**.html

Motrin (**Ibuprofen**) Drug Information: Uses, **Side Effects**, Drug ...
Learn about the prescription medication Motrin (**Ibuprofen**), drug uses, dosage, **side effects**, drug interactions, warnings, and patient labeling.
www.rxlist.com/**ibuprofen**-drug.htm

⋮

**Ibuprofen** - Motrin - Advil - Dosage - **Side Effects** - Interactions ...
**Ibuprofen** (Motrin, Advil, Mediprin, Nuprin, Motrin IB) drug information, dosage, **side effects**, drug interactions, and warnings. **Ibuprofen** is a NSAID (nonsteroidal anti-inflammatory ...
arthritis.about.com/od/**ibuprofen**/**Ibuprofen**_Motrin_Dosage_**Side**_**Effects**_Interactions...

Sponsored sites

Vox Big **Effects** Pedal
Amps & **Effects** at Guitar Center. Free Shipping on 1000's of Items.
GuitarCenter.com

**Side** by **Side** Refrigerators
Find **side** by **side** refrigerators at Great Prices.
www.Pronto.com

**Effects** of Global Warming
Your Source for **effects** of global warming at GlobalWarmingGuide.info.
GlobalWarmingGuide.info

Rent **Side**ways
Rent **Side**ways on DVD. No late fees Free trial. Subscribe today.
www.Netflix.com

Up**side** Prepaid Credit Card
100% Approval. No Credit Check. Apply for a Prepaid Credit Card Today.
CreditCardXPO.com

1 2 3 4 5 Next

ibuprofen side effects

**Figure 3:** Segmentation of a search engine results page (SERP) by areas of interest (AOI). Each of the 10 search result captions (title, snippet, URL) is regarded as its own AOI. These 10 result AOIs are our click prediction targets.

of the form: $\langle \sigma, t, u, \phi, y \rangle$ where $\sigma$ is a unique identifier for this impression, encoding information about the user and query, $t$ denotes the timestamp (relative to 0) of the measurement, $u$ is a unique identifier for the AOI, $\phi$ represents all of the feature values computed as discussed in Section 5.1, and $y$ is a boolean variable indicating whether $u$ was clicked during the session $\sigma$. We focus on impressions with exactly one detected click [1] and at least five detected cursor positions (73.8% of all query impressions). Running experiments with a smaller number of minimum cursor positions resulted in no performance differences relative to our baselines. This is because when cursor data is missing (<10% of all query impressions), our model effectively fall back to the strongest baseline of original search result ranking (as it is part of our model), and our gains over the baseline would just be slightly diluted by including this small fraction of traffic but findings and conclusions would remain the same.

We recorded the SERP interactions from the Microsoft Bing Web search engine. Log data were gathered from searchers in the control groups (i.e., with no experimental treatments on frontend or backend) of multiple experiments on the search engine in the U.S. English geographic locale, run between May 2011 and June 2012, during external experiments on small fractions of user traffic. For the duration of the experiments, all queries from searchers in the control groups (which is a representative sample of the overall search traffic) are recorded along with their cursor movements.

## 6.2 Training

We selected a random set of 100K users (and their 1M queries) from our dataset. We split the set by searcher and time with 80% for training and 20% for testing. That is, all tuples belonging to the same searcher–and therefore impression–were in the same split. The prediction targets are set to 4 for clicked hyperlinks and 0 for hyperlinks that were not clicked. Our model outputs a score for each hyperlink at each cursor movement, stopping and prefetching a link when its score exceeds a threshold $\tau$. Instead of optimizing for a fixed $\tau$, we present precision-recall curves to demonstrate performance at different operating points.

[1]The comparison on impressions without a click is not very relevant since they do not impact the user experience.

## 6.3 Evaluation

We focus evaluation on prefetching algorithmic search results. As discussed in Section 3, the page load time can vary dramatically. We therefore want to test our algorithm under various regimes. For a given score $\tau$, we evaluate our algorithm when given at least $\ell$ milliseconds to fetch a page before the click. So, for example, if $\ell$ is 500 ms and the user took 2000 ms to investigate the page before clicking, then we can observe the user's behavior for 1500 ms before losing our chance to get any benefit from prefetching. We compute precision and recall for $\ell \in \{500, 5000\}$ to demonstrate regimes of normal and severely-limited bandwidth. For a given $\ell$, the true positive ($TP$) is defined as the prefetching decision made for the link that was actually clicked at least $\ell$ time later. A negative ($TN$) occurs when the system accurately predicts that no clicks on any links occurred during the impression. The prefetching on an unclicked target link is considered false positive ($FP$) while prefetching made with lead time shorter than $\ell$ is considered as late positive ($LP$). A false negative ($FN$) is an impression where the model did not select any link, even though the user eventually clicked one. We evaluate the performance of our models (and the baselines described in the next subsection) using precision and recall. With the above definitions, the precision is then defined as $TP/(TP+FP)$ while recall is defined as $TP/(TP+LP+FN)$. Notice that a random prefetching system will achieve precision of $\frac{1}{|\mathcal{U}|}$.

## 6.4 Baselines

We employed three strong baselines: (i) the original search engine ranking, (ii) historic clicks from all searchers for the query, and (iii) historic clicks from the current searcher for the query. These baselines, in particular the latter two, are similar to prefetching methods proposed in prior research, in which, prefetching is determined by static estimates of likelihood of future content access from historic usage data [1, 13, 22, 26]. The baselines are defined as:

*Search engine ranking:* This baseline always prefetches the top-ranked result for the query as returned by the Bing search engine at the time the logs were collected. Note that this baseline is expected to be very strong since commercial search engines rank results by leveraging a variety of sources of evidence, including content and historic usage data, and the top-ranked search result often receives most clicks.

*Historic clicks (all users)* (denoted $p(click_{all})$): Selects the most popular clicked URL for the query. Multi-year click logs from a separate data source (same search engine, but not the cursor-tracking flights) were used to compute the probability of selecting a particular URL given the current query. The separate dataset was much larger than the set of data collected during the online cursor tracking experiments. This enabled broader query coverage and more reliable click predictions. The result that is most likely to be clicked based on this historic data was prefetched if it appeared in the current result ranking. To improve coverage, all URLs were normalized to remove trailing slashes, lowercase, and collapse https and http protocols.

*Historic clicks (current user)* (denoted $p(click_{user})$): Applies the personal navigation algorithm [43] to prefetch the result that was visited by the current searcher historically for the current query. Specifically, if the searcher has visited the same result for the previous two instances of the query, then that result will be prefetched for the current (third) instance of the query if it appears in the result ranking. URLs were normalized as with the previous baseline.

## 7. RESULTS

We now present our experimental results, starting with the descriptive statistics of the cursor movement data.

### 7.1 Descriptive Statistics

In this section, we characterize the features proposed in the previous subsection for building the prefetching models. Since our goal is to prefetch clicked results, we wanted to understand whether certain features could indicate future clicks. We report analysis across 41.6 million cursor samples from 186k unique query impressions used for training our model. The results are summarized in Table 2 and the discussions are organized by the feature group. We focus on the two classes of cursor samples with clicked AOI and unclicked AOI to understand the changes of feature values across the two groups. Features in the two global groups are impacting at the impression-level, i.e., features that impact overall clickthrough for the entire impression would, in turn, impact clickthrough for individual AOIs on the SERP. In contrast, the features in the two local groups impact directly regarding the AOI in question.

*Static Global*: All the features in this group are significantly different among the two groups. Interestingly, the queries with less clicks have higher frequencies, which may be due to the more likely presence of answer results – this hypothesis is further supported by the averaged higher value of the *has answer* feature in the *StaticLocal* group.

*Dynamic Global*: Almost all the features in this group are significantly different among the two groups except for cursor ycoord and cursor total time. The total cursor distance of cursor for the unclicked impressions is higher, as is the number of non-hyperlink clicks (often associated with text selections) and evidence of reading behavior, which suggest that people are exploring and more deeply engaged with examining the SERP (rather than clicking).

*Static Local*: All the features in this group are significantly different among the two classes. As we can see, certain types of AOIs are indeed more likely to attract more clicks. For example, the clicked class has larger *AOI area*, which makes sense, as larger AOI may be more likely to attract user attention. Other examples include the rate of AOI having the card attribute (e.g., additional information such as brand logo and/or deep links) which may increase both user confidence in document quality as well as attractiveness, resulting in more chance of clickthrough. In contrast, having answer directly in the AOI, as discussed earlier, reduces the chance of the AOI being clicked due to "good abandonment" [20]. Also, as expected, the clicked AOI tends to have lower rank (demonstrated by both lower *AOI rank* and *AOI ycoord*).

*Dynamic Local*: There are interesting differences in this group's features. Cursor hovers on the AOI are a significant indicator of an impending click, as is dwell time on the AOI (which is much higher when a click is observed). The speed and acceleration toward the AOI suggests that the searcher is performing a focused movement before the click. This is also supported by the higher value of *AOICursor ontarget*.

**Table 2:** Descriptive statistics of the proposed features for the two classes of cursor sample with clicked AOI and unclicked AOI. The differences between the two classes are statistically significant for the majority of the features based on Welch's $t$-test ($p < 0.05$) except for features that are noted with *. (s) denotes seconds, (px) denotes pixels.

| Feature name | Mean (Standard deviation) | |
|---|---|---|
| | Unclicked | Clicked |
| Static Global | | |
| Query frequency | 79K (408K) | 74K (389K) |
| Query click entropy | 1.697 (1.218) | 1.630 (1.186) |
| SERP has advertisements | 0.330 (0.470) | 0.322 (0.467) |
| SERP has related searches | 0.739 (0.439) | 0.795 (0.403) |
| Dynamic Global | | |
| Cursor xcoord (px) | 457 (277) | 468 (272) |
| Cursor ycoord (px)* | 334 (224) | 334 (228) |
| Cursor delta (px) | 89 (123) | 87 (121) |
| Num non hyperlink clicks | 0.122 (0.010) | 0.103 (0.010) |
| Cursor is reading | 0.081 (0.273) | 0.074 (0.261) |
| Cursor max ycoord (px) | 396 (246) | 397 (253) |
| Cursor max AOI rank | 3.10 (2.25) | 3.21 (2.10) |
| Cursor total distance (px) | 773 (918) | 765 (918) |
| Cursor total time (s)* | 47 (54) | 47 (55) |
| Cursor time delta (s) | 6.17 (15.70) | 5.93 (13.70) |
| Static Local | | |
| AOI area (px$^2$) | 58510 (35340) | 75177 (51117) |
| AOI width (px) | 682 (195) | 632 (86) |
| AOI height (px) | 89 (56) | 120 (81) |
| AOI rank | 5.75 (2.72) | 2.57 (2.27) |
| AOI xcoord (px) | 148 (75) | 177 (39) |
| AOI ycoord (px) | 515 (414) | 354 (261) |
| AOI has card | 0.002 (0.043) | 0.192 (0.394) |
| AOI has answer | 0.375 (0.484) | 0.136 (0.343) |
| Dynamic Local | | |
| AOI is visible | 0.527 (0.499) | 0.755 (0.430) |
| AOICursor hover | 0.002 (0.107) | 0.319 (0.466) |
| AOICursor distance (px) | 558 (331) | 367 (283) |
| AOICursor angle | 4.21 (65.41) | 16.39 (11.23) |
| AOICursor proximity | 1.28 (0.84) | 1.23 (0.61) |
| AOICursor speed | 4.77 (36.47) | 7.03 (39.04) |
| AOICursor acceleration | -0.104 (96.88) | 0.667 (98.62) |
| AOICursor jerk* | 2.45 (1010.79) | 3.42 (977.76) |
| AOICursor ontarget | 0.110 (0.372) | 0.136 (0.455) |
| AOICursor xdistance (px) | 316 (270) | 299 (265) |
| AOICursor ydistance (px) | 389 (312) | 155 (176) |
| AOICursor dwell (s) | 0.08 (0.64) | 1.03 (2.50) |
| AOICursor title dwell (s) | 0.03 (0.37) | 0.34 (1.46) |

### 7.2 Prefetching Experiments

We present the results of our prefetching experiments for different lead times in the precision-recall curves in Figure 4. As we can see from these curves, for comparable recall levels, we achieve substantial improvements over prefetching based on all cursor-agnostic methods. Although performance drops when we require a conservative five second leadtime, our algorithm still outperforms baselines by a significant margin. We present the results of significance tests in Table 3 for a high precision model (with a high value of $\tau$) and a high recall model (with a low value of $\tau$).

#### 7.2.1 *Effect of Query Type*

One might suspect that pre-fetching decisions for naviga-

**Table 3:** Comparison with baselines. Superscripts denote statistically significant improvements over a competing run using a Student's $t$-test ($p < 0.05$) with respect to rank (r), $p(click_{all})$ (a), $p(click_{user})$ (u), high precision model (P), or high recall model (R).

| Model | Precision | Recall |
|---|---|---|
| rank | $0.605^{a}$ | $0.602^{auP}$ |
| $p(click_{all})$ | 0.399 | $0.326^{u}$ |
| $p(click_{user})$ | $0.697^{ar}$ | 0.032 |
| high recall | $0.723^{aur}$ | $0.670^{aurP}$ |
| high precision | $0.877^{aurR}$ | $0.567^{au}$ |

**Table 4:** The effectiveness of prefetching models trained suppressing the specified feature groups against the full model trained with all features the score threshold $\tau$ of 3 and a leadtime $\ell$ of 500 ms. * represents statistically significant *decreases* in performance with respect to using all features using a Student's $t$-test ($p < 0.05$).

| Suppressed | Precision | Recall |
|---|---|---|
| - | 0.877 | 0.567 |
| Static Global | 0.865* | 0.566 |
| Static Local | 0.875 | 0.531* |
| All Static | 0.819* | 0.547* |
| Dynamic Global | 0.877 | 0.563 |
| Dynamic Local | 0.876 | 0.464* |
| All Dynamic | 0.831* | 0.432* |
| User Deviation | 0.880 | 0.563 |

tional queries, because they have a single target result, can be made without cursor information. To test this, we examined the performance of our algorithm on queries defined as navigational (i.e., with a click entropy less than or equal to one, as in [42]). Figure 4b demonstrates the higher performance of all methods, including baselines. The differences in performance are statistically significant ($p < 0.05$). We similarly investigated the performance of our model when evaluating only on informational queries (i.e., queries with a click entropy of two or more (again, as in [42]). These queries involve more thorough examination of the ranked list. This behavior can result from either multiple intents, poor retrieval performance, or higher recall intent. Because of the diversity of the click patterns, we suspect that our baselines will perform less well on these queries. The results (Figure 4c) demonstrate the lower performance of all runs, including our model. Nevertheless, the model-based approach significantly improves over the baselines ($p < 0.05$).

### 7.2.2 Feature Ablation

We present the feature ablation experiments in Table 4, where we remove one feature group at a time to examine the effectiveness of each of them in the presence of other feature groups. To do this, we use the high precision model, whose overall results are reported in Table 3. Static features, taken as a whole, contribute substantially (removing them results in a 6.6% drop in precision and a 3.5% drop in recall). This reflects the importance of visual layout and attractiveness in successful prefetching. Importantly, the degradation is not observed when suppressing global or local features alone, suggesting that static features perform best when using conjunctions of local and global features. Dynamic features, taken as a whole, also provide significant information (removing them results in a 5.2% drop in precision and a 23.8% drop in recall). The majority of this contribution comes from local features, suggesting that information about the AOI with respect to the cursor are critical to high performance. The local features are important in recall since they provide signals about each of the AOIs that may be missed in general SERP-level analysis. For the high precision model, user deviation features appear to add no value over the other features, in combination.

**Table 5:** Feature importance. Five most and least important features and weights. Weights are normalized to be in unit range with respect to the most important feature (AOICursor hover from the *Dynamic Local* class).

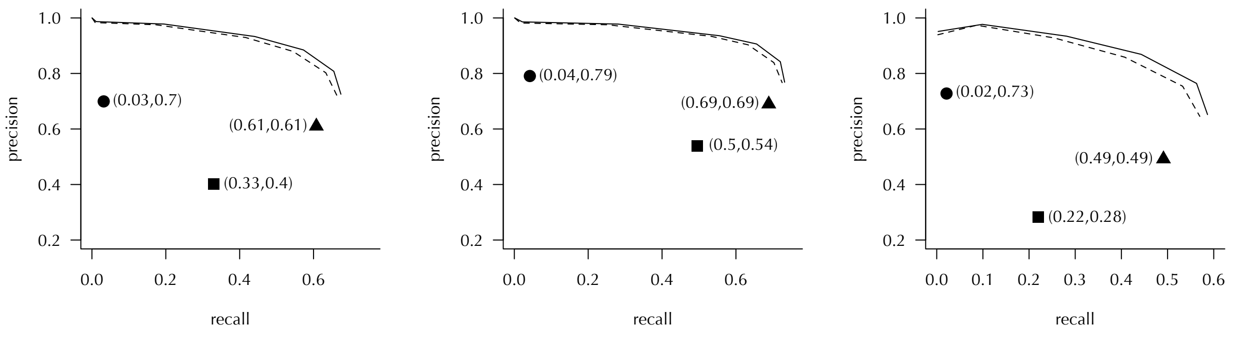
| Feature name | Importance |
|---|---|
| AOICursor hover | 1.000000 |
| AOI has card | 0.443321 |
| AOI rank | 0.373836 |
| Cursor max AOI rank | 0.228811 |
| Query frequency | 0.077288 |
| ⋮ | ⋮ |
| USERDEV AOICursor acceleration | 0.000066 |
| AOICursor jerk | 0.000062 |
| USERDEV AOICursor jerk | 0.000062 |
| AOICursor on target | 0.000061 |
| Cursor is reading | 0.000000 |

### 7.2.3 Feature Importance

Finally, we wanted to understand feature importance. We present the most and least informative features in Table 5. Unsurprisingly, hovering over an AOI is a strong signal that a click is imminent. Other *Dynamic Local* features, such as AOICursor ydistance, are also important but are not in the top five. While features such as the rank position of the result AOI and the distance between the cursor and the AOI might seem obvious, others are less so. The presence of a card suggests that the visual attractiveness of an AOI may result in a higher clickthrough rate. Variations in searcher attention as a function of caption attractiveness have been noted in previous studies (e.g., [11]). Conversely, query click entropy appears to help the model distinguish between more predictable behavior connected to navigational intentions and less predictable behaviors for informational intent. The less informative features involve more granular cursor movements (e.g., jerk, acceleration), suggesting that in order to perform effectively our prefetching model only requires a coarse model of search interaction behavior.

## 8. DISCUSSION AND IMPLICATIONS

We have introduced the search result prefetching challenge and real-time prefetching methods to address it based primarily on cursor movements. Our approach significantly outperforms three strong baselines leveraging result rankings and historic clicks to prefetch search results. Analysis of the differences between the prefetch time and click time reveals that on average, our method could save searchers around 650ms per query for the 65% of queries where our model correctly prefetches the clicked result. The average prefetch lead time is 6-7 seconds (depending on $\tau$), but we



**(a)** All **(b)** Navigational **(c)** Informational

**Figure 4:** Click precision and recall. The solid line indicates performance as a function of the score threshold $\tau$ for a leadtime $\ell$ of 500ms. The dashed line indicates the performance for a leadtime $\ell$ of 5s. The points indicate the performance of baselines prefetching based on rank position (▲), the probability of click (all users) (■), and probability of click (current user) (●).

cap our time savings to the median landing page load time from earlier (672ms). 650ms in time savings per accurate prediction is a sizable efficiency increase, especially at Web scale, where prefetching could benefit millions of searchers.

Our findings showed that we can achieve strong prediction accuracy. As with other studies [8, 9] we noted evidence of task effects (e.g., our models do better for navigational queries, where search behavior is more predictable). We break out our findings by query type, but not by searcher type. Variability in interactions between searchers is high [8] and differences in model performance per individual or cohort should be considered. This could inform decisions about the selective application of cursor-based prefetching.

There are some limitations in this study. First, we consider prefetching only one result per SERP. In the future, we plan to extend our method to handle prefetching of multiple results during SERP examination. Second, in our evaluation we represent cost as the action of prefetching a result. In practice, cost is more nuanced: there is a variable cost associated with prefetching, depending on the nature of the prefetched page (e.g., size, content types). Third, we have not fully explored the trade-off between client-side code optimization and its impact on prefetching performance. In the current implementation, we optimize our client-side code to ensure no significant delay on PLT for SERPs with the logging enabled. This includes lazy loading of the logging code, enforcing mouse sampling (i.e., every 250ms or 8px of movement), and optimizing our prediction latency in both feature extraction (i.e., constant time in the number of cursor samples) and model execution (i.e., using decision trees, which can be turned into heavily optimized binaries). However, we do not know whether these optimizations are truly optimal and plan to further explore this in future work. Finally, we focused on desktop devices (e.g., PCs) in this study. We will extend our methods to mobile devices (e.g., smartphones, tablets), where multi-touch interactions and viewport dynamics would replace the mouse cursor. Bandwidth may be more limited in mobile settings and result prefetching could prove to be more valuable therein.

Looking ahead, the ability to prefetch visited resources has a number of implications. People on slow network connections (e.g., in developing countries or remote locations in developed countries) can benefit from faster landing page loading, as would those engaged in time-critical tasks [28]. Accurately prefetching clicked results also allows search engines to offer enhanced interaction capabilities such as augmenting landing pages to better support transitions from SERPs (e.g., clickable snippets [15]). More broadly, our prefetching methods could help reduce latency in any Website, especially those with low traffic or large amounts of dynamically-generated content; both of which can hinder the use of historic activity in prefetching decisions.

## 9. CONCLUSIONS AND FUTURE WORK

We presented a dynamic framework for prefetching Web pages in response to SERP interaction behavior. Previous models have focused on static prediction using historic data. We show that incorporating aspects of the SERP visual layout and dynamic on-SERP behavior, such as cursor movement, can substantially improve the accuracy of real-time prefetching decisions and may improve searcher efficiency. In future work, we will experiment with more sophisticated models to better capture the temporal dynamics and personal nature of cursor movements. While our technique generalizes beyond SERPs, optimal performance may require task-specific models. We will extend our analysis to a diverse range of non-SERP Web pages. Finally, we will adapt our models to mobile devices where viewport and touch interactions can be leveraged in place of cursor movements.

## 10. REFERENCES

[1] E. Agichtein and Z. Zheng. Identifying "best bet" web search results by mining past user behavior. *SIGKDD*, 902–908, 2006.

[2] I. Arapakis, X. Bai, and B.B. Cambazoglu. Impact of response latency on user behavior in web search. *SIGIR*, 103–112, 2014.

[3] M. Barreda-Ángeles et al. Unconscious physiological effects of search latency on users and their click behaviour. *SIGIR*, 203–212, 2015.

[4] T. Asano, E. Sharlin, Y. Kitamura, K. Takashima, and F. Kishino. Predictive interaction using the delphian desktop. *UIST*, 133–141, 2005.

[5] R. Baeza-Yates et al. The impact of caching on search engines. *SIGIR*, 183–190, 2007.

[6] R. Blanco et al. Caching search engine results over incremental indices. *SIGIR*, 82–89, 2010.

[7] C.J. Burges. From ranknet to lambdarank to lambdamart: An overview. Technical Report MSR-TR-2010-82, Microsoft Research, 2010.

[8] G. Buscher et al. Large-scale analysis of individual and task differences in search result page examination strategies. *WSDM*, 373–382, 2012.

[9] E. Cutrell and Z. Guan. What are you looking for?: An eye-tracking study of information usage in web search. *SIGCHI*, 407–416, 2007.

[10] J. Dean and L.A. Barroso. The tail at scale. *CACM*, 56(2):74–80, 2013.

[11] F. Diaz et al. Robust models of mouse movement on dynamic web search results pages. *CIKM*, 1451–1460, 2013.

[12] Z. Dou, R. Song, and J.-R. Wen. A large-scale evaluation and analysis of personalized search strategies. *WWW*, 581–590, 2007.

[13] T. Fagni et al. Boosting the performance of web search engines: Caching and prefetching query results by exploiting historical usage data. *TOIS*, 24(1):51–78, 2006.

[14] L. Fan et al. Web prefetching between low-bandwidth clients and proxies: Potential and performance. *SIGMETRICS*, 178–187, 1999.

[15] H. Feild, R.W. White, and X. Fu. Supporting orientation during search result examination. *SIGCHI*, 2999–3008, 2013.

[16] Q. Guo and E. Agichtein. Towards predicting web searcher gaze position from mouse movements. *SIGIR*, 3601–3606, 2010.

[17] Q. Guo and E. Agichtein. Beyond dwell time: estimating document relevance from cursor movements and other post-click searcher behavior. *WWW*, 569–578, 2012.

[18] E. Horvitz. Continual computation policies for utility-directed prefetching. *CIKM*, 175–184, 1998.

[19] J. Huang, R.W. White, and G. Buscher. User see, user point: Gaze and cursor alignment in web search. *SIGCHI*, 1341–1350, 2012.

[20] J. Huang, R.W. White, and S. Dumais. No clicks, no problem: using cursor movements to understand and improve search. *SIGCHI*, 1225–1234, 2011.

[21] T. Joachims et al. Accurately interpreting clickthrough data as implicit feedback. *SIGIR*, 154–161, 2005.

[22] S. Jonassen, L. Granka, and F. Silvestri. Prefetching Query Results and Its Impact on Search Engines. *SIGIR*, 631–640, 2012.

[23] D. Lagun et al. Discovering common motifs in cursor movement data for improving web search. *WSDM*, 183–192, 2014.

[24] D.M. Lane et al. A process for anticipating and executing icon selection in graphical user interfaces. *Int. J. Human-Computer Inter.*, 19(2):241–252, 2005.

[25] S. Lee, J. Yoo, and G. Han. Gaze-assisted user intention prediction for initial delay reduction in web video access. *Sensors*, 15:14679–14700, 2015.

[26] R. Lempel and S. Moran. Predictive caching and prefetching of query results in search engines. *WWW*, 19–28, 2003.

[27] R.B. Miller. Response time in man-computer conversational transactions. *AFIPS*, 267–277, 1968.

[28] N. Mishra et al. Time-critical search. *SIGIR*, 747–756, 2014.

[29] A. Murata. Improvement of pointing time by predicting targets in pointing with a pc mouse. *Int. J. Human-Computer Interaction*, 10(1):23–32, 1998.

[30] F.F. Nah. A study on tolerable waiting time: how long are Web users willing to wait? *Behaviour & Information Technology*, 23(3):153–163, 2004.

[31] J. Nielsen. *Usability Engineering.* Morgan Kaufmann, 1993.

[32] J. Nielsen. User interface directions for the web. *CACM*, 42(1):65–72, 1999.

[33] V.N. Padmanabhan and J.C. Mogul. Using predictive prefetching to improve world wide web latency. *SIGCOMM Comput. Commun. Rev.*, 26(3):22–36, 1996.

[34] P.T. Pasqual and J.O. Wobbrock. Mouse pointing endpoint prediction using kinematic template matching. *SIGCHI*, 743–752, 2014.

[35] J. Psaroudakis and M. Khambatti. A deeper look at task completion. blogs.bing.com/search/2013/10/14/a-deeper-look-at-task-completion, 2013.

[36] J. Ramsay, A. Barbesi, and J. Preece. A psychological investigation of long retrieval times on the world wide web. *Interacting with Computers*, 10(1):77–86, 1998.

[37] K. Rodden et al. Eye-mouse coordination patterns on web search results pages. *SIGCHI EA*, 2997–3002, 2008.

[38] E. Schurman and J. Brutlag. Performance related changes and their user impact. *Velocity*, 2009.

[39] B. Shneiderman. Response time and display rate in human performance with computers. *ACM Comput. Surv.*, 16(3):265–285, 1984.

[40] A. Singhal and M. Cutts. Using site speed in web search ranking. googlewebmastercentral.blogspot.com/2010/04/using-site-speed-in-web-search-ranking.html, 2010.

[41] J. Teevan et al. Slow search. *CACM*, 57(8):36–38, 2014.

[42] J. Teevan, S.T. Dumais, and D.J. Liebling. To personalize or not to personalize: Modeling queries with variation in user intent. *SIGIR*, 163–170, 2008.

[43] J. Teevan, D.J. Liebling, and G. Ravichandran Geetha. Understanding and predicting personal navigation. *WSDM*, 85–94, 2011.

[44] R.W. White, P. Bailey, and L. Chen. Predicting user interests from contextual information. *SIGIR*, 363–370, 2009.

[45] Q. Yang, H.H. Zhang, and T. Li. Mining web logs for prediction models in www caching and prefetching. *SIGKDD*, 473–478, 2001.

[46] B. Ziebart, A. Dey, and J.A. Bagnell. Probabilistic pointing target prediction via inverse optimal control. *IUI*, 1–10, 2012.