Search Result Prefetching on Desktop and Mobile

RYEN W. WHITE and FERNANDO DIAZ, Microsoft Research
QI GUO, Google

Search result examination is an important part of searching. High page load latency for landing pages (clicked search 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 (primarily bandwidth and battery) 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. In this article, we introduce this challenge and present methods to address it in both desktop and mobile settings. Our methods leverage searchers' cursor movements (on desktop) and viewport-based viewing behavior (on mobile) on search engine result pages (SERPs) 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 (prefetch top-ranked results), (ii) past SERP clicks from all searchers for the query (prefetch popular results), or (iii) past SERP clicks from the current searcher for the query (prefetch results that the searcher prefers). Our promising findings have implications for the design of search support in desktop and mobile settings that makes the search process more efficient.

CCS Concepts: • **Information systems** → **Web searching and information discovery**; **Users and interactive retrieval;**

Additional Key Words and Phrases: Search result prefetching, attention modeling, mouse cursor, viewport, mobile

ACM Reference Format:
Ryen W. White, Fernando Diaz, and Qi Guo. 2017. Search result prefetching on desktop and mobile. ACM Trans. Inf. Syst. 35, 3, Article 23 (May 2017), 34 pages.
DOI: http://dx.doi.org/10.1145/3015466

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 (so-called landing pages) 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) [Arapakis et al. 2014; Schurman and Brutlag 2009], the relationship between latency and the loading of landing pages is less well understood, even though waiting for landing pages to load can add delays to the search process. As shown in previous research on latencies in interactions with computer systems [Shneiderman 1984] and interactions with Web pages [Nielsen 1999]

An earlier version of parts of this work appears in Diaz, F., Guo, Q., and White, R.W. (2016). Search result prefetching using cursor movement. *Proc. SIGIR*, pages 609–618.
Authors' addresses: R. White, Microsoft Research, Redmond, WA; F. Diaz, Microsoft Research, New York City, NY; Q. Guo, Google, Mountain View, CA. Corresponding author email: ryenw@microsoft.com.

DOI: http://dx.doi.org/10.1145/3015466

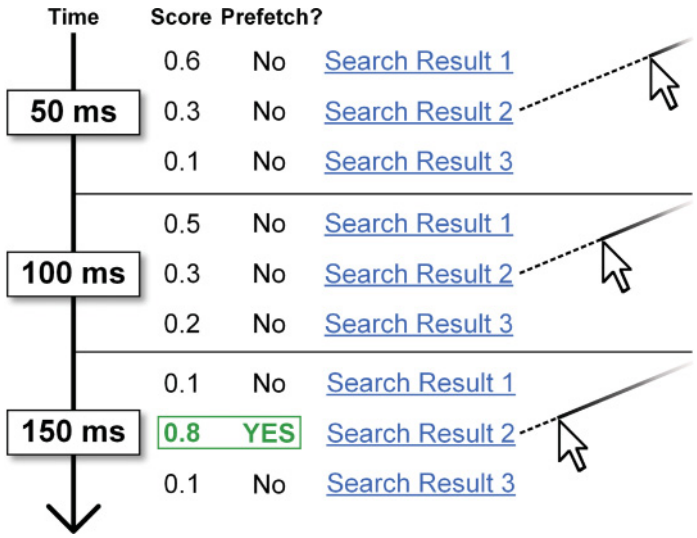


Fig. 1. Example of cursor-based prefetching. The model estimate of whether the searcher will select one of the top three results over time (at 50ms, 100ms, 150ms) is shown alongside each link (as *Score*). These estimates change as a function of the searcher's cursor movements on the SERP. The cursor 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 pre-defined threshold (0.8 for Search Result 2), that result is prefetched in anticipation of future selection by the searcher.

and, most recently, with search result pages [Crescenzi et al. 2016] in particular, such latency has a significant negative impact on the overall user experience. Methods to proactively fetch the content of particular search results before they are selected could benefit searchers in terms of time saved, while balancing the costs involved in mistakenly downloading content that is never viewed; we define this as the *search result prefetching challenge*.

To address the search result prefetching challenge, the prefetching system needs to predict which result the searcher will select next. In previous research, such predictions are usually made via static estimates learned from historic usage data at the population or individual level [Fagni et al. 2006; Fan et al. 1999; Jonassen et al. 2012; Lempel and Moran 2003; Yang et al. 2001]. For example, search engines prefetch the top result for queries where there is little variation in intent (primarily navigational queries [Agichtein and Zheng 2006]). However, these methods are only applied for small sets of queries where the dominant intent is clearly defined and observable via prior click patterns. Recent research on prefetching for video browsing has shown that effective dynamic models can be learned from a combination of mouse cursor and eye-gaze interactions [Lee et al. 2015]. The reliance on gaze data in particular raises doubts about the scalability of those methods. In contrast, we present scalable methods based solely on cursor data (on desktop) or viewport data (on mobile) to dynamically update 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 in anticipation of selection by the searcher.

On desktop, as illustrated in Figure 1, our method leverages the mouse cursor movements on the SERP (which can now be collected in a scalable manner [Buscher et al. 2012; Huang et al. 2011]), contextualized by the query, the results, and the searcher's historic activity to make *real-time predictions*. If we can correctly prefetch early enough, then we can save people significant time—this is especially important for those

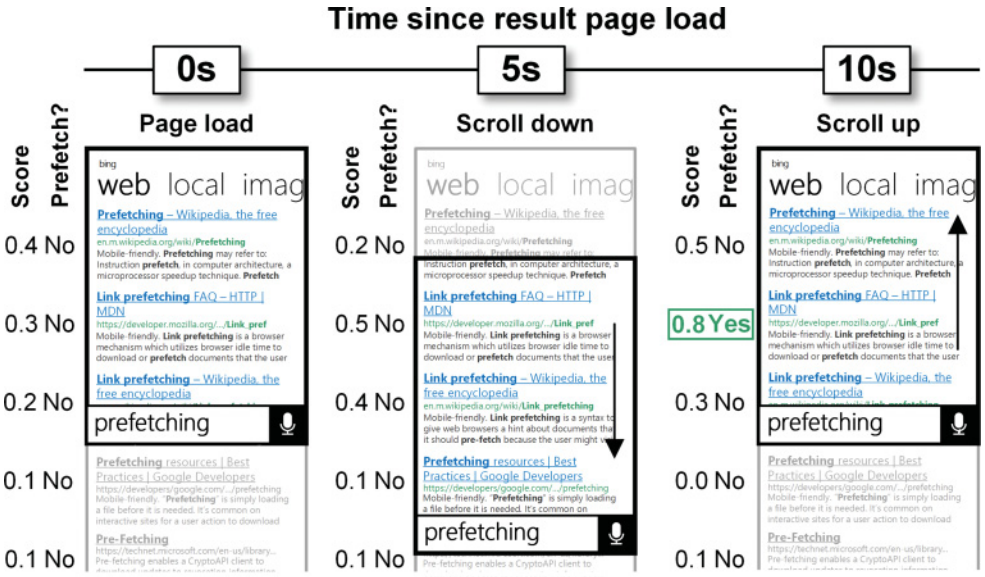


Fig. 2. Example of viewport-based prefetching for mobile search. The viewport position is in the schematic is reported at 5s intervals over the first 10s after the SERP loads. The searcher scrolls down after 5s and scrolls back up after 10s. A prefetching score is computed for each of the results at each time interval. When that prefetching score reaches a threshold (τ), set to 0.8 in this case), the second result is prefetched.

accessing the Web on low-bandwidth connections, who may need to be more selective about the results that they view.

Research in mobile prefetching seeks to balance the utility of the prefetched content against wasted bandwidth and battery, both of which are more pertinent in mobile settings than on the desktop [Higgins et al. 2013; Mohan et al. 2013; Wang et al. 2015a]. On mobile, instead of relying on cursor movements (which are unavailable given the focus on touch-based interactions), we present viewport-based prefetching as a way to estimate searcher attention on mobile SERPs and learn models to prefetch clicked landing pages. Just as searchers need to hover with their cursor in a desktop setting prior to clicking on a result, the result needs to be visible in the viewport to be selected. Since the viewport on mobile settings can be small, it offers a focused view on the SERP and can provide a good indication of searcher attention. Prior work has shown that viewport duration and eye-gaze duration are highly correlated [Lagun et al. 2014]. Figure 2 presents an example of viewport-based prefetching in operation. As the searcher scrolls to examine the SERP, the result captions that are visible in the viewport change (including some that are re-examined during the impression as the viewport moves up and down the SERP), and the model updates its estimates of which results will be selected. When the model score for a particular result reaches a given threshold (determined *a priori* based on desired performance characteristics, for example, in terms of precision and recall), the landing page is prefetched.

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 from our method);
—Develop machine-learned models that leverage a rich array of features to prefetch search results pre-click;

—Propose novel prefetching methodologies that span desktop and mobile settings, and leverage mouse cursor and viewport signals, to prefetch search results;
—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 search result prefetching for search system design and for society generally (e.g., enabling more rapid information access for those on slower Internet connections).

The remainder of this article is structured as follows. Section 2 describes related work in areas such as prefetching, the impact of response latency on searcher engagement, and monitoring interaction behavior, especially with SERPs. Section 3 motivates our research by demonstrating the potential impact of prefetching in a search context. 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. Section 6 presents the data used in training and testing our prefetching models, and characteristics of that data are provided in Section 7. In Section 8, we describe our experimental setup, including datasets and evaluation. In Section 9, we present the results of our experiments, demonstrating the effectiveness of our method compared to three strong baselines. Section 10 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 11 and present opportunities for future work.

2. RELATED WORK

A number of research areas are relevant to the work presented in this article: (i) the impact of latency on interactions with computers and search engines, (ii) methods to reduce latency via prefetching content, and (iii) methods for mining and modeling search interactions. We describe related work in each of these areas in turn before focusing on the contribution and benefit of our research over previous work.

2.1. Latency Effects

In the human-computer interaction community, there has been a significant amount of research on the impact of system response times on the quality of search interactions [Nielsen 1999]. Rapid responses to user instructions (i.e., less than a second) are preferred over delayed action and can increase user productivity [Miller 1968; Shneiderman 1984; Nielsen 1993]. Shneiderman [1984] reviewed the literature on computer response time and recommended that computers should respond immediately, in part based on limitations in human short-term memory [Miller 1968]. In online settings, latencies in responding to a hyperlink selection are based on factors such as Web browser performance, Internet connection speed, and the nature of the content requested (e.g., plain text versus rich media). Web download speeds are an important aspect of the user experience [Nielsen 1999]. A number of studies have considered tolerable Web page load times (PLTs) [Cohen and Kaplan 2000; Nah 2004], including the psychological impact of varying PLT [Ramsay et al. 1998].

Search engines have made significant infrastructure investments to reduce their response times [Dean and Barroso 2013]. Studies have shown that even small increases in latency (e.g., increasing the load time of the SERP by as little as 100ms) can lead to lower searcher engagement that persists over time, even once the latency has been reduced [Schurman and Brutlag 2009]. For example, Google conducted online experiments where they intentionally injected server-side delays, ranging from 100 to 400ms,

into the search results to observe the impact on people's behavior. They found that increasing the load time of the SERP by as little as 100ms decreased the number of searches per person. These differences increased over time and persisted even after the experiment ended [Schurman and Brutlag 2009]. Arapakis et al. [2014] showed that response time increases of 500ms were noticeable by searchers and reduced click-through rates. Beyond search result examination, Google recently integrated site speed as a ranking feature [Singhal and Cutts 2010], in recognition of the importance of page load time to searchers. Rather than leveraging latency as a signal, result prefetching methods can be useful, especially in low-bandwidth settings, where access to content may be delayed [Fan et al. 1999]. Recent work on "slow search" discusses the cost-benefit tradeoffs in retrieving search results quickly and what could be accomplished given more time [Teevan et al. 2014]. Beyond just making search faster (or slower), there are other reasons why page load latency matters, for example, to address network bandwidth constraints [Fan et al. 1999].

To address latency in query responses, researchers have designed caches to rapidly serve results [Baeza-Yates et al. 2007], including ways to leverage historic search behavior [Fagni et al. 2006; Jonassen et al. 2012; Lempel and Moran 2003]. 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 for popular queries [Agichtein and Zheng 2006]. Since repeat visits to the same result is common [Teevan et al. 2011], search engines have focused on identifying and promoting sites from historic access data that are likely to be clicked; reducing the time for searchers to locate these results on SERPs. For example, the Bing search engine already uses new browser capabilities to prefetch results that are highly likely to be selected [Psaroudakis and Khambatti 2013]. This enhances the search experience by reducing overall latency in ways extending beyond SERP generation.

Studies of search latency have focused on desktop search settings. However, latency effects may be even more acute in mobile search settings. As we show in Section 3, there is a significant increase in PLT for landing pages accessed on mobile devices, meaning that prefetching methods may be more useful on these devices. In this article, we present prefetching models that perform effectively in both desktop and mobile settings.

2.2. Prefetching

Researchers have studied Web page prefetching in general (see Domenech et al. [2012] for a good survey). The challenge of reducing latency has been addressed in the context of search engines, for example, in caching the results for frequent queries [Baeza-Yates et al. 2007] or leveraging Web browser capabilities to prefetch definitive results for common queries [Agichtein and Zheng 2006]. Other methods have used surfing patterns in the aggregate [Padmanabhan and Mogul 1996; Yang et al. 2001; Pitkow and Pirolli 1999] or individually [Fan et al. 1999] to prefetch pages that are likely to be selected. Padmanabhan and Mogul [1996] use N-hop Markov models based on surfing patterns for improving prefetching strategies for Web caches. Fan et al. [1999] leverage a user's historic surfing activity and a Markov predictor tree to predict future resource requests. Pitkow and Pirolli [1999] proposed longest subsequence models instead of Markov models. Sarukkai [2000] used Markov models to predict the next Web page accessed. Yang et al. [2001] 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. [2009] 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 [Horvitz 1998] proposed decision-theoretic methods for the ideal use of idle time for computational problem solving.

Continual computational methods can be applied for selective (utility-directed) content prefetching, including partial prefetching of specific Web page elements, while balancing associated costs and benefits.

Many of the existing prefetching methods proposed in the previous paragraph rely on access to significant amounts of data reflecting searcher activity over time. As such, these methods are static and, because they rely on sufficient volume in previous activities, only cover a subset of the resources that people access. This may be reasonable for popular queries/pages or active searchers but less applicable for other scenarios with less data (e.g., tail queries). The methods that leverage browsing data rely 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 [Blanco et al. 2010]. Recent work on prefetching based on dynamic signals such as interactions with resources yields much greater coverage of sought content but leverages signals such as gaze tracking [Lee et al. 2015], which is inaccessible to mobile search systems and impractical to apply at scale in both desktop and mobile settings (although recent advances attention modeling in mobile and desktop settings suggest that may be changing [Lagun et al. 2014; Papoutsaki et al. 2016]). In our earlier work [Diaz et al. 2016], we demonstrated the feasibility of cursor-based prefetching in desktop settings. In this article, we extend that work to also consider mobile settings, where cursor movement signals are unavailable yet prefetching may be even more valuable.

Despite the lack of access to such rich interaction signals, prefetching of content on mobile devices has still been well studied in light of the potential user benefits. Their focus is on the tradeoffs between time saved from prefetching and the costs associated with energy and cellular data usage [Higgins et al. 2013; Parate et al. 2013]. Components in these models include sequence modeling [Parate et al. 2013], bulk prefetching [Mohan et al. 2013], and content analysis and scheduling [Higgins et al. 2013; Wang et al. 2015a]. The tradeoff between energy usage and performance in Web page loading (independent of prefetching) has also been the focus of detailed study [Thiagarajan et al. 2012; Bui et al. 2015]. The most related work in this area focuses on training predictive models using large amounts of log data [Lymberopoulos et al. 2012]. However, despite its popularity, research in this area has not focused on the search result prefetching challenge on mobile devices, as we do in this article. Our work extends this research by introducing dynamic prefetching signals, allowing finer-grained predictions with much stronger performance. Furthermore, because our study focuses on Web search, we adopt signals from the search domain that are unavailable in the more general setting.

2.3. Monitoring Interaction Behavior

Recently, there has been an increase in the use of rich models of interaction behavior to better understand searchers' interests, intentions, and attention on SERPs [Guo and Agichtein 2010a; Lagun et al. 2014] and beyond [Guo and Agichtein 2012]. These signals can be used to disambiguate searcher intentions [Guo and Agichtein 2010a], or estimate search relevance [Huang et al. 2012; Lagun et al. 2014]. Many of these methods rely on cursor movements as the core signal of searcher attention, especially at scale [Huang et al. 2011], given that it has been shown to correlate with eye gaze [Guo and Agichtein 2010b; Rodden et al. 2008]. Beyond controlled settings, recent work has shown that such methods can be deployed at scale online [Buscher et al. 2012; Huang et al. 2011]. This facilitates a better understanding of search behavior [Buscher et al. 2012] and enables predictions of SERP examination activity [Diaz et al. 2013], improved relevance estimation [Huang et al. 2012], ranking [Lagun et al.

2014], and predicting searcher satisfaction [Liu et al. 2015] based on common patterns in cursor signals. From these 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. Endpoint prediction methods have used simple regression based on peak movement velocity calibrated to the searcher [Asano et al. 2005], normative kinematic laws that use a velocity-over-distance profile [Lank et al. 2007], Kalman filters [Aydemir et al. 2013], neural networks [Biswas et al. 2013], inverse control theory [Ziebart et al. 2012], or kinematic template matching, which treats velocity profiles as two-dimensional stroke gestures [Pasqual and Wobbrock 2014]. Simple versions of target prediction leverages distance from the mouse cursor [Lane et al. 2005] or consider angles between the movement vector and vectors for the target positions [Murata 1998]. More sophisticated methods build probabilistic models based on previous clicks [Ziebart et al. 2012], consider whether users have entered into the corrective sub-movement phase of their click [Aydemir et al. 2013], or use kinematic template matching [Pasqual and Wobbrock 2014]. Related research in the HCI community has used mouse movement data for user attention inference [Xu et al. 2016]. Although there are similarities between that 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 [Joachims et al. 2005] and cognitive biases [White 2013]). 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. Note that previous research in click prediction modeling (e.g., Richardson et al. [2007], Dupret and Piwowarski [2008], Chapelle and Zhang [2009], Guo et al. [2009], and Wang et al. [2015b]) focuses on the aggregated click distribution at the query-level instead of the dynamic/real-time prediction for individual search events proposed in this article.

Cursor movements are unavailable in mobile settings, where the interaction is primarily touch and voice based. As such, other signals such as the viewport may be particularly useful. Huang and Diriye [2012] proposed the viewport as an estimate of searcher attention in mobile settings. Huang et al. [2012] showed that scrolling (viewport position adjustment) was a useful feature in models to estimate result attractiveness and satisfaction. Buscher et al. [2012] showed that scrolling was a useful feature in clustering searchers based on their SERP examination patterns. Although based on cursor movements rather than scrolling, ViewSer [Lagun and Agichtein 2011] operates under a similar principle that monitoring exposed parts of SERPs can provide useful signals of interests and intentions. Focusing specifically on mobile devices, research has shown that viewport monitoring can yield useful insights regarding the relevance of search results [Guo et al. 2013] and searcher satisfaction [Lagun et al. 2014; Williams et al. 2016]. Shokouhi and Guo [2015] recently used viewport duration (and clickthrough data) to infer pseudo-relevance labels to rank cards in proactive recommendation scenarios.

2.4. Contributions over Previous Work

Our research extends previous work in a number of ways. First, we focus on prefetching for individual search events, based on real-time user behavior during each visit to individual SERPs. Most prefetching methods rely on static predictions and

transition probabilities learned from historic data. Second, many of the studies of end-point prediction (e.g., based on mouse cursor movements) focus on carefully controlled studies in laboratory settings. In contrast, we operate in a non-controlled environment 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 viewport-based behavior, our approach can better adapt to the current search situation and less common informational queries—and increase searcher efficiency and reduce resource waste compared to three strong baselines. Many current prefetching methods within Websites and search engines focus on providing this support for popular pages only. Fourth, 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., Asano et al. [2005]). Finally, we break out our findings in different ways and demonstrate that there are classes of information requests and searchers for which our proposed dynamic prefetching methods perform well.

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 (both on desktop and mobile), we analyzed one week of logs from the Internet Explorer Web browser from the start of June 2015. PLT was defined as the time between the page being requested and it fully loading in the Web browser. To help control for variations in information needs between mobile and desktop, we focused on the set of unique queries appearing in the dataset used for our prefetching analyses (described later) and computed the PLT for the clicked landing pages for those queries on Bing.com for queries issued from both desktop and mobile settings. Landing page load time was made available through event logging in Internet Explorer. We focused on Bing since that was the engine used in the remainder of our analysis. We did not control for URL directly, as many websites have mobile variants and we wanted a realistic sense of latency on different devices.

The cumulative distribution function for PLT on mobile and desktop between 200 and 5000ms is shown in Figure 3. Pages with PLTs <200ms are assumed to be cached and hence excluded from this analysis. The mean and median PLTs are 1282ms and 672ms, respectively, on desktop and 1470ms and 771ms, respectively, on mobile devices. If we consider the 75th percentile for PLT (a commonly-used latency threshold, marked in the figure), then there is a clear difference between desktop (950ms) and mobile (1350ms) (highlighted in red in Figure 3). Searchers in mobile settings would benefit more from prefetching. Searchers start to notice PLT delays at 500ms [Arapakis et al. 2014] (or even earlier [Barreda-Ángeles et al. 2015]), suggesting that if we could make an accurate prediction about which results to prefetch, then we could noticeably improve the search experience for approximately 60% of desktop SERP clicks and approximately 70% of mobile SERP clicks.

Beyond the overall distribution, we also wanted to better understand the distribution of PLTs at the searcher level. Focusing on searchers with at least 100 landing page visits in the 1-week period (to provide sufficient data), we find that the average percentage of landing-page PLTs of 500ms or more (first computed per searcher and then averaged across all searchers) is 55.6% on desktop (standard deviation = 19.2%) and 65.2% on mobile (standard deviation = 21.1%). In total, 16.6% of desktop searchers and 37.6%

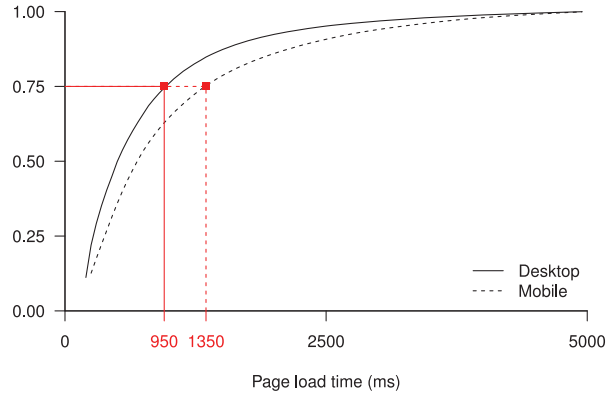


Fig. 3. Cumulative distribution function for PLT on mobile vs. desktop landing pages (spanning the range 200–5000ms). Median values for desktop and mobile settings are marked using solid and dashed lines, respectively.

of mobile searchers were dramatically affected by high PLTs (defined as 75% or more of their landing pages loading in 500ms or longer). Searchers with consistently higher latencies were disproportionally located in developing countries (e.g., Eritrea, Chad, Ethiopia), in remote locations (e.g., Falkland Islands, British Indian Ocean Territory), in states in the U.S. with slower wired and cellular Internet connections (e.g., Alaska), or are visiting less popular websites that may not have the infrastructure to support rapid access. These findings motivate the development of technology to help people access information more efficiently.

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 is 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 (in our case, cursor movements on desktop, and viewport changes on mobile) that 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^*.

The formulation slightly differs between desktop and mobile. On desktop, this budget represents a conservative estimate of the cost, in terms of time and bandwidth, of prefetching the result, and considers all clicks equally, for example, 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. On mobile, where latency and bandwidth usage need to be more strictly constrained [Nicolaou 2013], we consider these factors directly in the evaluation of the prefetching model.

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 on desktop or the fraction of result AOI that is visible in the viewport on mobile). Our model computes this relevance score continually during the searcher's interaction with the SERP. If the score exceeds a predetermined threshold τ (chosen based on the desired operating characteristics in terms of precision and recall), then

the landing-page content is fetched. In this section, we will describe the features and models that we developed to address 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 or changes the viewport (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; fraction of AOI that is visible in the viewport). Features are computed every time the cursor position is sampled (i.e., after 250ms have passed or the cursor has moved at least eight pixels) on desktop and each time the content displayed in the viewport is updated (i.e., every time the viewport moves at least 20px (a customizable parameter)) for each of the result AOIs, since the goal is to predict the result AOI that will be selected. We stress that all of our features are lightweight and available at runtime. Even aggregate features such as click frequency and average PLT can be embedded into SERP content, and do not need to be computed in real time by the prefetching model.

Previous research has shown that modeling differences in cursor movements with the normative behavior for each searcher can help better estimate document relevance [Guo and Agichtein 2012] on desktop, due to the larger variability in cursor movements, while the benefit of modeling searcher variability is limited for viewport behavior on mobile [Guo et al. 2013]. As such, for the desktop setting, we include the normalized version for each feature for each searcher using the deviations from average for the ⟨searcher, feature⟩ pair. Searcher deviation is defined as the difference between the feature value at the current cursor position and the average feature value for that searcher computed over all their historic actions. Searcher deviation variants are included for *Dynamic Global* and *Dynamic Local* features in our desktop setting.

The features of our models on desktop and mobile are summarized in Tables I and II, respectively.

5.1.1. Static Global Features. Static global features capture properties of the impression, which do not change throughout the query impression. Page features describe visual aspects of the layout such as whether it includes an advertisement or a set of related searches/query suggestions. Query features describe historically observed behavioral properties of the query such as its frequency and its *click entropy*, a measure of the randomness of clicks over results [Dou et al. 2007]. These historic features were computed using a sample of query and click logs from the Microsoft Bing search engine, spanning a time period of 1.5 years prior to the timeframe used for training and testing our learned models.

5.1.2. Dynamic Global Features. Previous work has observed that search activity may be context dependent. For example, Huang et al. demonstrate that the correlation between gaze and cursor position may depend on how much time has elapsed since the page loaded [Huang et al. 2012]. Our dynamic global features aim to capture these within-impression changes in context.

On desktop, we 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 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

Table I. Features Used in the Desktop Prefetching Model. Coordinates Are Relative to the Upper-Left Corner of the SERP. Distances, Coordinates, and Areas Are Measured in Pixels. A Set of Features Is Generated for Each AOI $x \in \mathcal{X}$ with Respect to the Current Cursor Position c_t as Well as the History of Cursor Positions $\mathcal{C}_t = \{c_0, \ldots, c_t\}$. Global Features Are Shared Across All AOIs. Local Features Include Information about the Candidate AOI x

Feature name	Description
Static Global	
ads($\mathcal{X}$)	$\mathcal{X}$ has advertisement
querySugg($\mathcal{X}$)	$\mathcal{X}$ contains a query suggestion (related search)
H(q)	q click entropy computed over 1.5 years prior to study
freq(q)	q frequency
Dynamic Global	
x(c_t)	horizontal position of c_t
y(c_t)	vertical position of c_t
dist(c_t, c_{t-1})	distance in pixels between c_t and c_{t-1}
nonhyper($\mathcal{C}_t$)	number of clicks in $\mathcal{C}_t$ that are not on hyperlinks, for example, for text selections in result snippets
maxY($\mathcal{C}_t$)	maximum vertical position of $\mathcal{C}_t$
maxAOI($\mathcal{C}_t$)	maximum AOI rank of $\mathcal{C}_t$
cursordist($\mathcal{C}_t$)	total cursor distance of $\mathcal{C}_t$
time(c_t, c_0)	total time of impression
time(c_t, c_{t-1})	time difference between c_t and c_{t-1}
Static Local	
area(x)	area of x (in pixels)
width(x)	width of x (in pixels)
height(x)	height of x (in pixels)
rank(x)	rank position of x
x(x)	horizontal position of x
y(x)	vertical position of x
card(x)	whether x has special image (e.g., a brand logo) and/or additional result information such as deep links
answer(x)	whether x is a vertical result (weather, stock, etc.)
Dynamic Local	
vis(x, v_t)	x intersects current viewport
hover(x, c_t)	c_t is over x
reading($x, \mathcal{C}_t$)	reading behavior on x (i.e., following text with cursor [Rodden et al. 2008])
dist(x, c_t)	distance of c_t to center of x
angle(x, c_t)	angle between direction vector of c_t to center of x. AOI with least deviation receives 1, otherwise 0.
proximity(x, c_t)	proximity changes of c_t with respect to x. 2 = moving away from x, 1 = moving toward x, 0 = same distance from x.
speed(x, c_t)	speed of c_t toward x
accel(x, c_t)	acceleration of c_t toward x
jerk(x, c_t)	jerk of c_t toward x
ontarget(x, c_t)	c_t on track to visit x. Project least-squares line through $\{c_{t-5}, \ldots c_t\}$. Return 1 if it intersects x, otherwise 0
xdist(x, c_t)	horizontal distance between c_t and x
ydist(x, c_t)	vertical distance between c_t and x
dwell(x, c_t)	dwell time of c_t in x
titledwell(x, c_t)	dwell time of c_t in result title of x

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, while 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

Table II. Features Used in the Mobile Prefetching Model. A Set of Features Is Generated for Each AOI $x \in \mathcal{X}$ with Respect to the Current Viewport Position v_t as Well as the History of Viewport Positions $\mathcal{V}_t = \{v_0, \ldots, v_t\}$. Global Features Are Shared across All AOIs. Local Features Include Information about the Candidate AOI x

Feature name	Description
Static Global	
ads($\mathcal{X}$)	$\mathcal{X}$ has advertisement
querySugg($\mathcal{X}$)	$\mathcal{X}$ contains an query suggestion (related search)
H(q)	q click entropy computed over 1.5 years prior to study
freq(q)	q frequency
Dynamic Global	
time(v_t, v_{t-1})	time difference between v_t and v_{t-1}
dist(v_t, v_{t-1})	distance in pixels between v_t and v_{t-1}
speed(v_t, v_{t-1})	speed of movement
time(v_t, v_0)	total time of impression
numVis(v_t, $\mathcal{X}$)	number of items in $\mathcal{X}$ intersecting with v_t
fracVis(v_t, $\mathcal{X}$)	fraction of $\mathcal{X}$ intersecting with v_t
width(v_t)	width of v_t
height(v_t)	height of v_t
x(v_t)	horizontal position of v_t
y(v_t)	vertical position of v_t
maxY($\mathcal{V}_t$)	maximum vertical position of $\mathcal{V}_t$
maxAOI($\mathcal{V}_t$)	maximum AOI rank of $\mathcal{V}_t$
scrolldist($\mathcal{V}_t$)	total scroll distance of $\mathcal{V}_t$
upscrolls($\mathcal{V}_t$)	number of scrolls up
downscrolls($\mathcal{V}_t$)	number of scrolls down
scrolls($\mathcal{V}_t$)	number of scrolls in any direction
Static Local	
area(x)	area of x (in pixels)
width(x)	width of x (in pixels)
height(x)	height of x (in pixels)
rank(x)	rank position of x
x(x)	horizontal position of x
y(x)	vertical position of x
answer(x)	whether x is a vertical result (weather, stock, etc.)
CTR(x)	clickthrough rate of x
avgPLT(x)	average page load time (PLT) of x
sdPLT(x)	standard deviation PLT of x
size(x)	page size of x (in bytes)
Dynamic Local	
vis(x, v_t)	x intersects v_t
frac(v_t, x)	fraction of x within v_t
titleVis(x, v_t)	title of x is in v_t
visArea(x, v_t)	raw area of x intersecting with v_t
frac(x, v_t)	fraction of v_t intersecting with x
dur(x, v_t)	duration of x on screen (ms)
dist(x, v_t)	distance of x to v_t
relDist(x, v_t)	x is above or below the v_t
numVis(x, $\mathcal{V}_t$)	number of times x has been visible

[Rodden et al. 2008] (which could be an indicator of interest in a landing page), a binary feature captures whether two consecutive left-right movements of cursor are observed.

On mobile, the features include simple time features as well as global properties of the viewport that are not related to any specific AOI on the SERP, such as scroll position and speed. Also included are general measures of engagement (e.g., the depth to which the result list is examined, the number of AOIs that are visible in the viewport at any

particular time), similar to those used in the measurement of searcher satisfaction [Lagun et al. 2014; Williams et al. 2016].

5.1.3. Static Local Features. Static local features provide two types of information about the result AOI: attractiveness and download impact. Attractiveness features refer to visual properties related to searcher attention and clickthrough, such as the AOI position and dimensions [Diaz et al. 2013]. On mobile, the size of the page and the historic PLT may be especially important given bandwidth limitations. As such, we implemented features related to size and historic latency to incorporate that information directly into the prediction process. Page size reflects the number of bytes of the target document taken from a crawl of the page before our study began. Since requesting a page may require executing and rendering supporting JavaScript source, images, and other content (the impact of which may not be fully reflected in the size of the page), we also included a feature capturing the historic page load time measured for search engine users. A page's average load time in seconds is derived from 6 months of Internet Explorer log data from a time period preceding our experiments. In the event that either value is missing, we use the mean feature value instead.

5.1.4. Dynamic Local Features. On desktop, we 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, and 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 of the cursor trajectory (i.e., draw a least-squares line through last five cursor movements and return a Boolean value with whether it intersects each AOI). We also compute the dwell time of 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. We also capture whether the AOI is visible in the viewport. If the AOI is hidden from view, then it cannot be selected.

On mobile, we capture the relationship between a target AOI and the viewport position. These features include binary variables indicating the visibility of the AOI or its title as well as scalar variables indicating how much of the AOI is visible, the total time for which the AOI is visible, and the number of distinct occasions on which the AOI is visible in the viewport (i.e., where it is completely hidden and then reappears as the result of scrolling). Since a target AOI must be visible in the viewport to be selected, we hypothesize that this information about the viewport-AOI relationship will be highly predictive of a click and hence be useful for prefetching purposes.

5.2. Model

We are interested in modeling the relationship between our set of pre-click features (Section 5.1) and the searcher result selection (click) event. 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 [Burges 2010]. 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 or viewport 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 or viewport sample, we featurize and score each $u \in \mathcal{U}$ and if the score of one or more results is above τ, we fetch u with the highest score. If no page is scored above this threshold, then we wait for the next interaction event (cursor move or viewport move) and re-evaluate the set of results. Since this threshold allows the system designer to manipulate model performance per desired operating characteristics, we present results for various threshold values in our experiments.

6. LOGGING SEARCHER BEHAVIOR

In this section, we describe the method we employed to collect the data used to train and test our prefetching models. To record searcher interactions with SERPs at scale without the need to install any browser plugins, we used an efficient and scalable approach [Buscher et al. 2012]. JavaScript-based logging functions were embedded into the hypertext markup language (HTML) source code of the Bing SERP. The datasets differed for desktop and mobile settings. We describe them separately in this section.

6.1. Desktop

To obtain a detailed understanding of searcher interactions with SERPs shown in desktop settings, 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 dimensions and vertical position. We optimized our implementation and ran 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 250ms. 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 and not adversely affect the user experience with delays associated with log data capture and data uploads. Since cursor tracking was relative to the document, we captured cursor alignment to SERP content regardless of how the searcher reached that position (e.g., 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, that is, the vertical coordinate of the uppermost visible pixel of the SERP in the browser viewport. This coordinate was checked 3 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 searcher's browser, we recorded the positions and sizes of AOIs. We use the method from Buscher et al. [2012] to identify and record the exact position of AOIs on SERP loading. The specific AOIs recorded were as follows: (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 4. 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 of all AOIs shown in Figure 4, we focus only on predicting clicks on 1 of the 10 result AOIs.

Fig. 4. Segmentation of a SERP in the desktop setting 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. A similar segmentation was performed for mobile SERPs.

6.2. Mobile

To obtain a detailed understanding of interactions with the SERP, we recorded (among other things, for example, result clickthrough) information on viewport position and size, scrolling, and the bounding boxes of several AOIs on the SERP. The method used for recording this SERP layout data was similar to that used in the desktop setting. The search results are displayed in a dedicated application on the phone where the results from Bing.com are shown in a SERP. The width and height of the viewport in pixels at SERP load time were logged. The viewport dimensions could not be adjusted. Overall, over 60% of the viewports had dimensions of approximately 430 $\times$ 520 pixels, illustrating the constrained view on SERPs offered by mobile devices.

The viewport sampling rate is impacted by two factors: (i) the frequency with which the Web browser sends the scroll update event (which is browser specific) and (ii) if the top of the viewport changes by at least 20px since the previous sample (this is a configurable parameter). We also recorded the current scroll position, that is, the vertical coordinate of the uppermost visible pixel of the SERP in the browser viewport, each time the searcher swiped up or down on the touch device. When the position of the viewport changed, the visible part of the SERP also changed. Tracking positional information of AOIs and the viewport allowed us to compute features for prefetching such as the distance between each AOI and the viewport at any point in time, in addition to other features such the viewport speed during scrolling, relative to each result.

7. BEHAVIORAL CHARACTERISTICS

We now describe the characteristics of searcher behavior reflected in the datasets, broken out by the two settings: desktop and mobile.

7.1. Desktop

In this section, we characterize the features proposed in Section 5.1 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 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,[1] 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. We report analysis across millions of cursor samples from 186k query impressions which were part of the external experiments.

The results are summarized in Table III, 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, that is, 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 significantly differ among the two groups. Interestingly, the queries with fewer clicks have higher frequencies, which may be due to the more likely presence of answer results—this hypothesis is further supported by the higher value of the *has answer* feature in the *StaticLocal* group.

Dynamic Global: Almost all the features in this group significantly differ 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 significantly differ 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 area(x), which makes intuitive sense, as larger AOIs may be more likely to attract people's attention. Other examples include the rate of AOI having the card attribute (e.g., additional information such as brand logo and/or deep links) that may increase both people's confidence in document quality as well as attractiveness, resulting in more chance of clickthrough. In contrast, having the answer directly in the AOI, as discussed earlier, reduces the chance of the AOI being clicked due to "good abandonment" [Huang et al. 2011]. Also, as expected, the clicked AOI tends to have lower rank (demonstrated by both lower rank(x) and y(x)).

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 ontarget(x, c_t).

Overall, there is strong evidence that there are statistically significant differences in the feature values when there are clicks versus when no clicks are present. The

[1]Since the style of the SERP has not been changed significantly since then (e.g., knowledge cards/direct answers were already prevalent on Bing in 2011 and 2012), we believe the findings from this dataset remain representative of current behavior. Nevertheless, we plan to collect new data from live experiments to confirm whether these findings still hold.

Table III. Descriptive Statistics of the Proposed Features for the Two Classes of Cursor Samples with Clicked AOI and Un-Clicked AOI. The Differences between the Two Classes Are Statistically Significant for the Majority of the Features Based on Welch's *t*-Test ($p < 0.05$, Adjusted for Family-wise Error Rates Using a Bonferroni Correction), Except for Features That Are Noted with *. The Values in the Table Denote the Mean Average and the Standard Deviation (in Parentheses). Also Shown in the Table Is the Cohen's *d* Value (Effect Size). Large Effects ($d \geqslant 0.8$) Are Marked in Bold

	Mean (Standard deviation)		
Feature name	Unclicked	Clicked	Cohen's d
Static Global			
freq(q)	79K (408K)	74K (389K)	0.0128
H(q)	1.697 (1.218)	1.630 (1.186)	0.0564
ads($\mathcal{X}$)	0.330 (0.470)	0.322 (0.467)	0.0171
querySugg($\mathcal{X}$)	0.739 (0.439)	0.795 (0.403)	–0.1384
Dynamic Global			
x(c_t) (pixels)	457 (277)	468 (272)	–0.0404
y(c_t) (pixels)*	334 (224)	334 (228)	0.0000
dist(c_t, c_{t-1}) (pixels)	89 (123)	87 (121)	0.0165
nonhyper($\mathcal{C}_t$)	0.122 (0.010)	0.103 (0.010)	**1.9000**
maxY($\mathcal{C}_t$) (pixels)	396 (246)	397 (253)	–0.0040
maxAOI($\mathcal{C}_t$)	3.10 (2.25)	3.21 (2.10)	–0.0522
cursordist($\mathcal{C}_t$) (pixels)	773 (918)	765 (918)	0.0087
time(c_t, c_0) (secs)*	47 (54)	47 (55)	0.0000
time(c_t, c_{t-1}) (secs)	6.17 (15.70)	5.93 (13.70)	0.0174
Static Local			
area(x) (pixels^2)	58510 (35340)	75177 (51117)	–0.3300
width(x) (pixels)	682 (195)	632 (86)	0.5337
height(x) (pixels)	89 (56)	120 (81)	–0.3873
rank(x)	5.75 (2.72)	2.57 (2.27)	**1.3873**
x(x) (pixels)	148 (75)	177 (39)	–0.7021
y(x) (pixels)	515 (414)	354 (261)	0.5968
card(x)	0.002 (0.043)	0.192 (0.394)	–0.4933
answer(x)	0.375 (0.484)	0.136 (0.343)	0.6817
Dynamic Local			
vis(x, v_t)	0.527 (0.499)	0.755 (0.430)	–0.5261
hover(x, c_t)	0.002 (0.107)	0.319 (0.466)	–0.6953
reading($x, \mathcal{C}_t$)	0.081 (0.273)	0.074 (0.261)	0.0268
dist(x, c_t) (pixels)	558 (331)	367 (283)	0.6694
angle(x, c_t)	4.21 (65.41)	16.39 (11.23)	–0.6883
proximity(x, c_t)	1.28 (0.84)	1.23 (0.61)	0.0804
speed(x, c_t)	4.77 (36.47)	7.03 (39.04)	–0.0581
accel(x, c_t)	−0.104 (96.88)	0.667 (98.62)	–0.0078
jerk(x, c_t)*	2.45 (1010.79)	3.42 (977.76)	–0.0010
ontarget(x, c_t)	0.110 (0.372)	0.136 (0.455)	–0.0576
xdist(x, c_t) (pixels)	316 (270)	299 (265)	0.0641
ydist(x, c_t) (pixels)	389 (312)	155 (176)	**1.2697**
dwell(x, c_t) (secs)	0.08 (0.64)	1.03 (2.50)	–0.3883
titledwell(x, c_t) (secs)	0.03 (0.37)	0.34 (1.46)	–0.2170

results suggest that prefetching models learned from these features may be useful for the important task of predicting which search result will be selected. Given the large sample sizes, even small differences in the means of the feature values are likely to be statisticially signficant. To address this, we also report the effect size in Table III, computed via Cohen's *d*. Thresholds for *d* correspond to small (0.2), medium (0.5), and large (0.8) effects [Cohen 1977]. There are especially large effects for features such as the rank position of the result AOI, the number of non-hyperlink clicks, and the *y*-distance between the cursor and the result AOI. Before describing the prefetching

Table IV. Characteristics of Viewport Activity. The Values Represent the Average or Percentage Across All Impressions in Our Dataset. The Values in Parentheses Denote the Standard Deviations

Behavioral characteristic	Value
Number of captions shown at one time	3.059 (1.473)
Percentage of SERPs with scrolling	58.65%
Average scroll depth	4.586 (3.757)

models, we first present some statistics on interaction and selection behaviors in the mobile setting.

7.2. Mobile

We analyzed viewport activity in a week of Bing.com logs from June 1–7, 2015, from a sample of 34M query impressions. Table IV reports a number of features that are important in understanding how people use the viewport to inspect mobile search results. The average scroll depth denotes the average of the maximum rank positions visible in the viewport. Over the course of the query impression, searchers decide between the results in the retrieved SERP, but at any point in time there are only a few options available. As mentioned earlier, since visibility in the viewport is a prerequisite of selection, the presence of a result in the viewport at any time could be valuable for predicting future clicks. Table IV also reveals that searchers scroll on 60% of the SERPs and on average consider an additional one to two results beyond what is visible in the initial viewport.

Beyond general statistics about how people employ viewports on SERPs, there may be differences in how people consider the result captions that were presented on the SERP. There are a number of ways that examinination can be defined. We focus on these definitions: (i) views and clicks, (ii) total time for which the caption is visible in the viewport and *arrival time* (i.e., the time from SERP load until the caption is first visible in the viewport), and (iii) the distribution of captions that were considered before a click, all as a function of rank position. Figure 5 presents the distributions down to the eighth rank position (when automatic SERP pagination is initiated in the Bing mobile search application).

The results from the figures mirror those from previous studies on eye-gaze tracking [Joachims et al. 2005; Cutrell and Guan 2007a] and cursor tracking [Huang et al. 2011]. Most of the top-ranked captions are examined by the searcher for some amount of time. The top result is only viewed around 90% of the time because there may be top of page advertisements or recourse hyperlinks that push the top result caption outside of the viewport on SERP load. The click distribution also follows a similar trend to that reported in previous work [Joachims et al. 2005]. We observe that searchers spend considerably more time inspecting the top-ranked results (3–5s on average) and they examine the results from top to bottom (both noted previously [Cutrell and Guan 2007a; Joachims et al. 2005; Huang et al. 2011]). Note that the visible time for the caption (i.e., the total time that any amount of the caption text is visible in the viewport) is likely an overestimate of the time that searchers spend examining the caption (and is approximately 2–3 times that reported in eye-gaze tracking studies on the desktop [Cutrell and Guan 2007a]), especially since multiple result captions are often visible simultaneously.

Given a click on a result at a rank position, we examined the relative positions of the other results that searchers may have considered (denoted in terms of a difference in rank position from the clicked result: negative means after the clicked result, positive means before it (as in Joachims et al. [2005])). To do this, we focused on

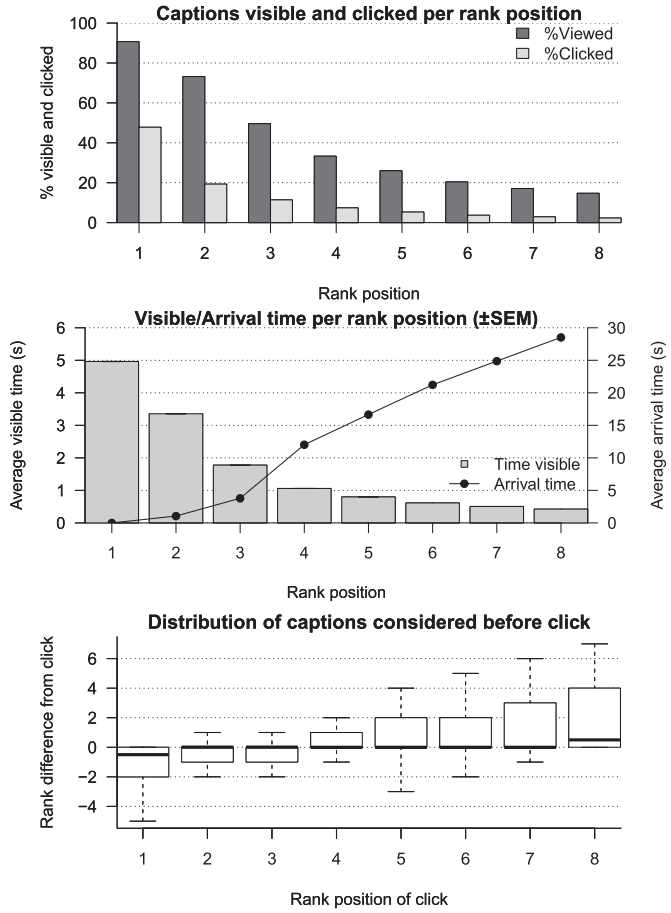


Fig. 5. Distributions per rank position of (i) result captions visible and clicked, (ii) visible time and arrival time (standard error of the means are included but are too small to be visible given the large sample sizes), and (iii) box-and-whisker plot with relative result positions considered before click.

captions that are visible in the viewport for at least 500ms. The box-and-whisker plot in Figure 5 shows that for clicks on the top three results people often considered proximal results (likely because multiple results are visible in the viewport), but for results at rank position 4 and above, it is more common to consider only the results above the clicked result. Since searchers examine results sequentially, they may stop when they encounter a result of interest (and not consider the next result like has been noted in the prior gaze and cursor tracking studies referenced above, since it requires additional scrolling effort). Focusing on click decisions in more detail, we find that (i) there is a fair positive correlation between consideration time and the presence of a click (point-biserial correlation (r_{pb}) = 0.377, $p < 0.001$) and (ii) there seems to be sufficient time before the click (i.e., median time-from-visible-to-click = 3948ms), when searchers are considering a specific result to fully prefetch it (i.e., median PLT (from earlier) = 771ms).

Overall, the evidence suggests that analyzing viewport dynamics can offer detailed information into how people examine SERPs in mobile settings. The results align well with how people examine results in desktop settings. Aspects of viewport behavior,

such as scroll depth and time spent viewing a caption, may make useful features for prefetching purposes. We explore this later in the article. Beyond specific patterns, the viewport also provides data on which results *could* be clicked that helps scope the set of results to be considered for prefetching purposes.

8. EXPERIMENTAL SETUP

We now describe experiments to measure the effectiveness of our prefetching models.

8.1. Data

We selected a random set of 100k searchers (and their 1M queries) (desktop) and 100k searchers (and their 500k queries) (mobile) from our datasets described earlier.

For both mobile and desktop data, the final, processed dataset consists of batches of instances of the form: $\langle \sigma, t, u, \phi, y \rangle$ where σ is a unique identifier for this impression, encoding information about the searcher and query, t denotes the timestamp (relative to 0) of the measurement, u is a unique identifier for the AOI, ϕ 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 σ. In the mobile setting, we take the additional step of modeling prediction decisions within the context of bandwidth constraints. That is, incorrectly downloading a very large page is worse than incorrectly downloading a smaller page. The target for each instance is the bandwidth-weighted quality of the link, defined as follows. The clicked link has the highest grade. The grades of the unclicked links are binned into below-average page size, average page size, and above-average page size. These four grades provide an ordinal target for our regression model. We leave alternative grading methods for future work. We regress against the quality target for each instance in the training set using the ensemble of gradient-boosted decision trees as noted above.

We focus on impressions with exactly one detected click in both settings[2] and at least five detected cursor positions in the desktop setting to provide a sufficient number of mouse movements from which to perform analysis (corresponding to 73.8% of all query impressions on desktop). 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 falls back to the strongest baseline comprising the original search result ranking (as it is part of our model), and our gains over this baseline would just be slightly diluted by including this small fraction of traffic but findings and conclusions would remain the same.

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.

8.2. Training

Given the large sample sizes involved, a single train-test split will yield reliable results. 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 τ. Instead of optimizing for a fixed τ, we present precision-recall curves to demonstrate performance at different operating points. When training our model, we fixed the following hyperparameters: number of trees to 500, number of leaves per tree to 70, minimum samples per leaf to 2000, and learning rate to 0.1. We did not modify these parameters using a validation set and believe that performance may be improved with some tuning.

[2]Comparisons on impressions without clicks are less relevant since they do not impact the user experience.

8.3. Evaluation

We focus the 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 τ, we evaluate our algorithm when given at least ℓ milliseconds to fetch a page before the click. So, for example, if ℓ is 500ms and the searcher took 2000ms to investigate the page before clicking, then we can observe the searcher's behavior for 1500ms 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 ℓ, the true positive (TP) is defined as the prefetching decision made for the link that was actually clicked at least ℓ time later. A true negative (TN) occurs when the system accurately predicts that no clicks on any links occurred during the impression. Prefetching an unclicked target link is considered false positive (FP) while prefetching made with leadtime longer than ℓ is considered as late positive (LP). A false negative (FN) is an impression where the model did not select any link, even though the searcher 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 $\frac{1}{|\mathcal{U}|}$.

We consider two metrics capturing aspects unique to the mobile setting. The first metric, *latency*, measures the amount of time the searcher will wait after the click. In the event of the system correctly prefetching the clicked result, the searcher incurs a latency of 0; otherwise the latency is proportional to the clicked page size. In practice, we normalize this value to a range between 0 or 1 to allow comparison across impressions. As such, the latency is also equivalent to the miss rate. The second metric, *bandwidth fallout*, measures the amount of data fetched during the impression over the data explicitly requested by the searcher. In the event of the system correctly prefetching the clicked result, the searcher incurs a bandwidth fallout of 0; otherwise the bandwidth fallout is proportional to the sum of the size of any incorrectly fetched pages. To allow comparison between impressions, we normalize the landing-page sizes relative to the maximum landing-page size in the candidate set and normalize a query impression's bandwidth fallout to lie between 0 and 1.

Latency and bandwidth fallout empirically trade off in most situations. A system that prefetches no pages will achieve a maximum latency of 1 and a minimum bandwidth fallout of 0. Conversely, if a system prefetches every page, then it will achieve a minimum latency of 0 and a maximum bandwidth fallout of 1.[3] Our threshold τ allows us to control where the system operates between low bandwidth/high latency and high bandwidth/low latency. We have observed empirically that latency monotonically improves and bandwidth fallout monotonically degrades as we reduce the operating threshold. As such, in our experiments, we measure system performance at a range of values for τ.

Resource and timing constraints limit the amount of prefetching that a system can perform before the searcher clicks. In order to model this, we restrict our system to prefetching at most one page. As a result, our bandwidth fallout ranges between 0, when the system correctly prefetches the clicked page, and 1, when the system incorrectly prefetches the largest landing page on the SERP. The latency metric remains unchanged.

[3]However, such a system though may incur latency due to bandwidth saturation or request ordering.

8.4. 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 [Agichtein and Zheng 2006; Fagni et al. 2006; Jonassen et al. 2012; Lempel and Moran 2003]. The baselines are defined as follows.

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 searchers) (denoted $p(click_{all})$): Selects the most popular clicked URL for the query. This baseline is a simplified proxy of the previous work in click prediction modeling (e.g., Richardson et al. [2007], Dupret and Piwowarski [2008], Chapelle and Zhang [2009], Guo et al. [2009], and Wang et al. [2015b]), which focuses on aggregated prediction of clicks (e.g., query, result pair) instead of the dynamic prediction for individual search event described in this article. Nevertheless, this group is a natural baseline to illustrate the value of the proposed dynamic predictions over the state-of-the-art static predictions. 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 searcher) (denoted $p(click_{searcher})$): Applies the personal navigation algorithm [Teevan et al. 2011] 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.

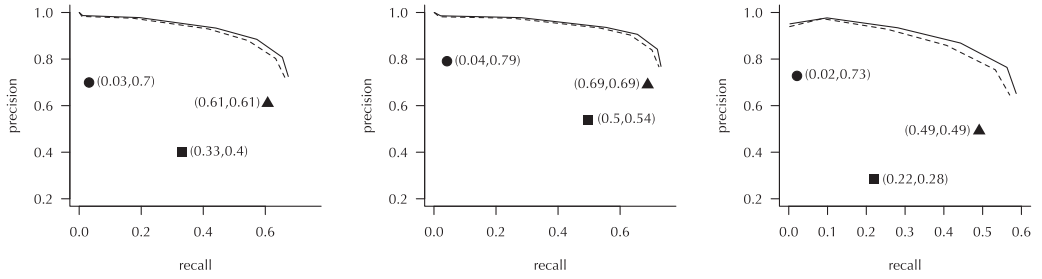
9. RESULTS

We now present our experimental results.

9.1. Desktop

We present the results of our prefetching experiments for different leadtimes in the precision-recall curves in Figure 6. 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 V for a high-precision model (with a high value of τ) and a high-recall model (with a low value of τ).

9.1.1. Effect of Query Type. One might suspect that prefetching decisions for navigational 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 Teevan



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

Fig. 6. Click precision and recall. The solid line indicates performance as a function of the score threshold τ for a leadtime ℓ of 500ms. The dashed line indicates the performance for a leadtime ℓ of 5s. The points indicate the performance of baselines prefetching based on rank position (▲), the probability of click (all searchers, $p(click_{all})$) (■), and probability of click (current searcher, $p(click_{searcher})$) (●).

Table V. 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), $p(click_{searcher})$ (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}
$p(click_{searcher})$	0.697^{ar}	0.032
high recall	0.723^{aur}	0.670^{aurP}
high precision	0.877^{aurR}	0.567^{au}

et al. [2008]). Figure 6(a) 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 Teevan et al. [2008]). 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 6(b)) demonstrate the lower performance of all runs, including our model. Nevertheless, the model-based approach significantly improves over the baselines ($p < 0.05$).

9.1.2. Feature Ablation. We present the feature ablation experiments in Table VI, 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 V. 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

Table VI. The Effectiveness of Prefetching Models Trained Suppressing the Specified Feature Groups against the Full Model Trained with All Features the Score Threshold τ of 3 and a Leadtime ℓ of 500ms. * Represents Statistically Significant *Decreases* in Performance with Respect to Using All Features Using a Student's *t*-Test ($p < 0.05$)

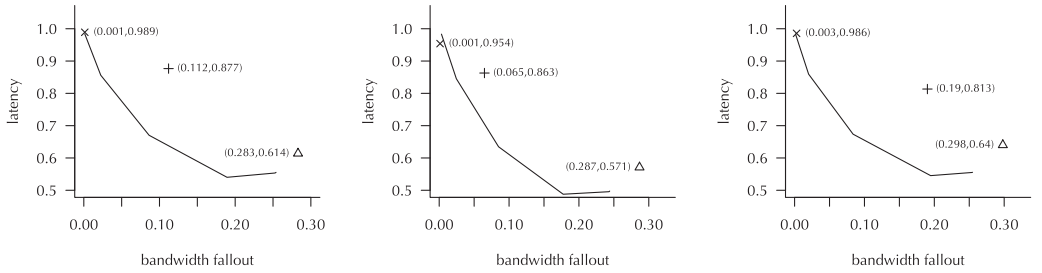
Suppressed	Precision	Recall
—	0.877	0.567
Static Global	0.865*	0.566
Static Local	0.875	0.531*
All Static	0.819*	0.547*
Dynamic Global	0.877	0.563
Dynamic Local	0.876	0.464*
All Dynamic	0.831*	0.432*
Searcher Deviation	0.880	0.563

Table VII. Feature Importance in Desktop Setting. The Ten Most and Least Important Features and Weights. Weights Are Normalized to Be in Unit Range with Respect to the Most Important Feature (hover(x, c_t) from the *Dynamic Local* Class). "Dev" Denotes That Features That Are Based on the Deviations from the Normative Behavior for the Current Searcher

Feature name	Class	Importance
hover(x, c_t)	DL	1.0000
card(x)	SL	0.4433
rank(x)	SL	0.3738
maxAOI$(\mathcal{C}_t)$	DG	0.2288
freq(q)	SG	0.0773
answer(x)	SL	0.0718
ydist(x, c_t)	DL	0.0691
H(q)	SG	0.0659
y(x)	SL	0.0411
height(x)	SL	0.0410
⋮	⋮	⋮
x(c_t) dev	DG	0.0008
dist(c_t, c_{t-1}) dev	DG	0.0007
vis(x, v_t)	DL	0.0006
proximity(x, c_t)	DL	0.0002
speed(x, c_t) dev	DL	0.0001
accel(x, c_t) dev	DL	0.0001
jerk(x, c_t)	DL	0.0001
jerk(x, c_t) dev	DL	0.0001
ontarget(x, c_t)	DL	0.0001
reading$(x, \mathcal{C}_t)$	DG	0.0000

of the AOIs that may be missed in general SERP-level analysis. For the high-precision model, searcher deviation features appear to add no value over the other features, in combination.

9.1.3. Feature Importance. Finally, we wanted to understand feature importance. We present the most and least informative features in Table VII. 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



(a) all **(b)** navigational **(c)** informational

Fig. 7. Performance by query type. The line shows performance as a function of the score threshold τ. Points indicate the performance of baselines prefetching based on rank position ($\triangle$), the probability of click (all searchers, $p(click_{all})$) ($+$), and probability of click (current searcher, $p(click_{searcher})$) ($\times$). Historic click behavior is used to measure performance for navigational queries (7(b)) and informational queries (7(c)). *Lower values are better.* Note that, because of our prefetch budget and nonuniform misclassification costs, we may not observe bandwidth fallout values in our experiments, resulting in truncated curves.

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., Diaz et al. [2013]). 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.

9.2. Mobile

We now report the results of the mobile prefetching, first spanning all queries and then broken out by different query and searcher types. There is also an analysis of the contributions made by the different features, as well as the battery consumption, which is important in applying these models in a mobile setting.

9.2.1. All Queries. We present evaluation results over all queries in Figure 7(a). We note here that, unlike precision-recall curves, for our curves, *lower numbers are better*, and an ideal curve hugs the axes. Turning to our baselines, we see that prefetching based on rank is very effective at reducing latency for many queries. However, because this baseline aggressively prefetches for all queries, those cases where a result below the first position was clicked will result in bandwidth fallout. The $p(click_{all})$ baseline can provide some correction for highly ranked URLs with low historic clickthrough rate. However, because these statistics are unavailable for many tail queries, the net effect of this baseline is higher latency due to many decisions to not fetch any pages. This behavior is amplified for our personalized baseline where the bandwidth fallout is reduced further but at an increase of latency. Our algorithm, which adaptively models the probability of a searcher clicking a hyperlink, dominates all baselines for all metrics. All differences are statistically significant using a Student's paired t-test ($p < 0.05$) except when comparing latency with $p(click_{searcher})$, with which it is statistically indistinguishable. These results suggest that our model is indeed able to take advantage of the dynamic features to improve prediction performance.

Table VIII. Average Performance Broken Down by Historic Searcher Scrolling Behavior. Shallow Searchers Tend to Only Inspect the Top Seven Results; Moderate Searchers Tend to Inspect the Top 10 Results; and Deep Searchers Tend to Inspect More Than 10 Results. Performance Is Averaged over Operating Thresholds in Our Experiments. Lower Values Are Better

	Fallout	Latency
Shallow	0.142	0.639
Medium	0.152	0.676
Deep	0.164	0.718

9.2.2. By Query Type. We examined the performance of our algorithm on queries defined as navigational (i.e., with a click entropy less than or equal to one) and informational (i.e., with a click entropy equaling of exceeding two) [Teevan et al. 2008]; same thresholds as used in the desktop setting. When inspecting the performance on navigational queries (Figure 7(b)), we observe superior performance on both metrics over the rank baseline ($p < 0.05$), comparable latency with superior bandwidth to the $p(click_{all})$ baseline ($p < 0.05$), and comparable bandwidth with inferior latency to the $p(click_{searcher})$ baseline ($p < 0.05$). These results demonstrate that, for navigational queries, our algorithm may be less precise than $p(click_{searcher})$ but performs more robustly than other baselines for other operating points. The performance on informational queries (Figure 7(c)) reflects that of the full set of queries, with dominance over all baselines except for statistical parity with $p(click_{searcher})$ for the latency metric. A clearer dominance for informational queries reflects this class's stronger representation in the corpus (48.29%) compared to navigational queries (5.5%). This may also suggest that behavioral data are more valuable when there is more ambiguity regarding the searcher's intent, as is likely with informational queries.

9.2.3. By Searcher Type. We were also interested in whether there were any differences in model performance as a function of searcher type. One way to do this is to consider how far down the result ranking people explore. This is relevant given the reliance on the viewport for most of the features in the model. We used the data for searchers with 10 or more impressions in the month (June 2015) prior to the time period used for training and testing the prefetching model. For each searcher, we computed the average maximum depth in the result lists that they considered. We hypothesized that the model would perform better for searchers who did not explore deeply, since the set of results assigned a high prefetching score from viewport-based features would be smaller. We split searchers into three groups based on their typical examination behavior: (i) *shallow* (11.82%), searchers who typically examined captions for results up to rank 7; (ii) *moderate* (72.80%), searchers who typically examined captions for results beyond rank 7 but less than rank 10; and (iii) *deep* (15.38%), searchers who typically examined captions for results at rank 10 or beyond. The performance relative to baselines is comparable to that found in Figure 7, so we suppress these plots for space reasons. However, when inspecting the average performance over operating thresholds (Table VIII), we notice that performance degrades as the searchers considered tend to read deeper in the ranked list. One explanation for this is that examining more result captions creates more options for prefetching (i.e., more results with a higher prefetching score), leading to more noise in the predictions and resultant prefetching decisions.

9.2.4. Feature Analysis. In order to understand the behavior of our model, we can analyze the contribution of individual features. In Table IX, we present the most and least

Table IX. Feature Importance in Mobile Setting. The Ten Most and Least Important Features and Weights for Modeling Our Bandwidth-adjusted Latency Targets. Higher Values Denote Greater Evidential Weight. Weights Are Normalized to Be in Unit Range with Respect to the Most Important Feature (rank(x) from the *Static Local* Class)

Name	Class	Importance
rank(x)	SL	1.0000
size(x)	SL	0.5845
dur(x, v_t)	DL	0.3540
y(x)	SL	0.3415
height(x)	SL	0.2495
avgPLT(x)	SL	0.2201
sdPLT(x)	SL	0.2130
H(q)	SG	0.1916
freq(q)	SG	0.1775
time(v_t, v_0)	DG	0.1518
⋮	⋮	⋮
upscrolls($\mathcal{V}_t$)	DG	0.0268
width(v_t)	DG	0.0266
titleVis(x, v_t)	DL	0.0265
fracVis(v_t, $\mathcal{X}$)	DG	0.0210
frac(v_t, x)	DL	0.0104
frac(x, v_t)	DL	0.0102
scrolls($\mathcal{V}_t$)	DG	0.0077
visArea(x, v_t)	DL	0.0066
width(x)	SL	0.0026
vis(x, v_t)	DL	0.0000

important features used for predicting the link's regression target. As can be seen, the most important features are dominated by static local features. In particular, the top-ranked feature, rank(x), indicates that this baseline provides a great deal of information but, as seen in Figure 7, it is not enough alone to provide strong performance. The only dynamic local feature appearing in the top features captures the duration of the AOI in the viewport. It supports our hypothesis that visual attention is predictive of a click. Interestingly, another important feature (height(x)) also reflects the attractiveness of an AOI. Other features are important for expressing the bandwidth of the candidate result (size(x), avgPLT(x)), the query type (H(q), freq(q)), and the time in the impression (time(v_t, v_0)). Many of the less-important features are either relatively static in our dataset (e.g., width(x), width(v_t)) or are correlated with higher ranked features (e.g., vis(x, v_t), visArea(x, v_t)).

Looking at a feature's contribution to regression performance overemphasizes the accuracy of predicting the exact link grade. We can measure a feature's more direct impact on our performance metrics by conducting ablation experiments. In Figure 8, we present the effect on performance resulting from holding out feature groups. The features whose removal results in the most dramatic reduction in performance are the *Dynamic Global* features. The reduction in performance is dramatic and clear, supporting the hypothesis that behavioral information can improve the effectiveness of click prediction. The removal of *Dynamic Global* features also results in a drop in performance, although slightly less. The static features, on the other hand, seem to provide weak contribution to the performance. These results may seem inconsistent with Table IX, but it is important to note that the feature importance values

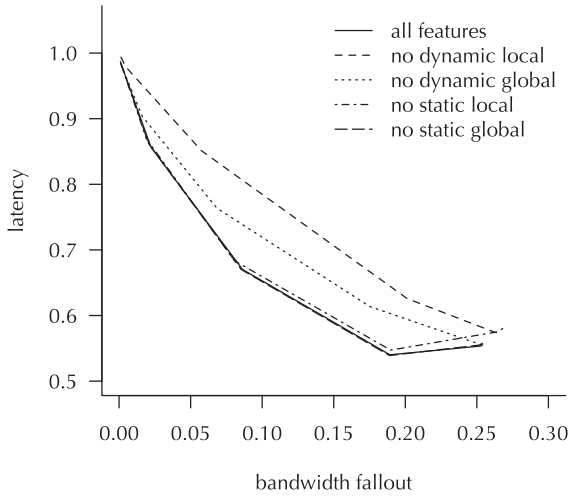


Fig. 8. Performance curves illustrating latency and fallout for the model using all features, and when removing each of the four feature groups in Table II. Removing the static global features has little impact on the performance of the model; the performance curve for no static global mostly overlaps with the curve for all features.

reflect the contribution to a regression objective, which is loosely related to our target evaluation metric(s). When we measure the performance more directly through ablation experiments, the feature contribution for the target evaluation metric is clearer.

9.2.5. Battery Consumption. One limitation of our work concerns the potential to impact battery usage in mobile settings. Our model, albeit lightweight, both in features and functional form, was evaluated under the assumption that the decision trees could be evaluated with each viewport move. A possible concern with running the model in production is the continual computation of the prefetching scores for the search results. As implemented for this study, the prefetching model recomputes the scores for each search result each time the position of the viewport updates. Analysis of the test data shows that the median number of viewport sampling points per query impression is 17, and there is a median of 208 requests to the model (to score a result) in total per impression. Conservatively, we assume that running the model on each result consumes 100% of the smartphone CPU. Recent work has shown that a 1GHz smartphone CPU (a Google Nexus One device with a 1400mAh battery) running at 100% will consume about 0.0833μAh (microampere hours) per millisecond [Murmuria et al. 2012]. Our experiments for this article (run on a desktop PC with an Intel Xeon E5-1650 CPU running at 3.20GHz) showed that it takes 0.006ms to assign a prefetching score to a particular search result using our model. Even if it takes 1ms per result in a mobile setting with a less powerful processor, we still expect the battery drain to be minimal. For example, if we use the median numbers of requests (208), we will consume 0.0173mAh per query (i.e., $208 \times 0.0833\mu$Ah) or 0.001% of the 1400mAh battery. Newer phones have longer battery lives (e.g., 1810mAh for the iPhone 6) and are more efficient in how they use computational resources, so the percentage could be even lower. Alternative strategies could be employed if needed, such as incorporating energy consumption into our objective [Wang et al. 2010], more carefully downsampling decision points, or even offloading aspects of the computation to the cloud.

10. DISCUSSION AND IMPLICATIONS

We have introduced the search result prefetching challenge and real-time prefetching methods to address it based on cursor movements (for desktop settings) and viewport movements (for mobile settings). We have presented results demonstrating the use of dynamic behavior information for adaptive click modeling, evaluated in a mobile prefetching context. We show that this method performs more effectively than three non-adaptive baselines, operating at comparable latencies. These gains are substantial and could provide significant improvements to the user experience (reduced latency) at the cost of a slight increase in bandwidth (increased bandwidth fallout).

Our findings showed that we can achieve strong prediction accuracy. As with other studies [Buscher et al. 2012; Cutrell and Guan 2007b], 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 and searcher type (mobile setting only). Variability in interactions between searchers is high [Buscher et al. 2012] and differences in model performance per individual or cohort should be considered. Whether a query is navigational or informational [Broder 2002] is likely to be highly correlated with the probability of scrolling and hence could be useful to predict whether prefetching is necessary. Navigational queries have a lower query click entropy, since in the aggregate there is less variation in the results that are chosen. In contrast, the query click entropy for informational queries is higher, since a number of different results are likely to be selected. We capture this as a feature in the prefetching model so it can be taken into consideration during learning and also used for post-hoc segmentation of the analysis results. Query and/or searcher properties could also inform decisions about the selective application of cursor-based prefetching.

Our analysis indicates that our performance is strongest for queries whose relevant results are concentrated at the top of the ranking (i.e., navigational queries, shallow-browsing searchers). This reflects the strength of the rank signal for these situations but also points to an opportunity to improve modeling for other situations where the searcher is interacting more with the SERP. We believe that improvements should come both from more sophisticated modeling (including the adoption of temporal decays or temporal cutoffs during feature generation), as well as from richer signals, perhaps including proximity measurements from capacitive sensors on touch devices. Although the focus in this article has been on prefetching within SERPs, there is a good opportunity to employ these methods for prefetching purposes within other settings such as general website navigation. We also need to understand the utility of viewport-based prefetching on other types of device such as tablet computers, where searchers may still not be able to view the full SERP at any one time.

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 tradeoff 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.

It is worth noting that the task of search result prefetching shares some similarities with the task of general search personalization, whether short term or long term [Bennett et al. 2012]. In both sets of experiments, we used a state-of-the-art personalization baseline (e.g., personal navigation [Teevan et al. 2011]) that leverages searchers historic behavior for the current query. The superior performance of the cursor model to the personal navigation baseline in both cursor- and viewport-based prefetching suggests that there are additional signals available in real-time engagement with the SERP that add predictive value beyond the searchers past experience. Future work will include additional features and baselines, such as those that leverage within-session behavior, for example, White et al. [2010], for prefetching decisions, as well as evaluating our models in terms of predicting multiple clicks.

We also need to experiment with additional features and more sophisticated models that explicitly consider the costs and benefits of past prefetching decisions, the temporal sequencing of interactions or the intentionality of searcher interactions, for example, the degree of focus that is observed in cursor movement trajectories [Pasqual and Wobbrock 2014]. Features based on the historic distribution of clicks across the set of top-ranked results, for the current query, current searcher, current searcher-query pair, or independent of either, may also be useful. Previous work on click prediction modeling (e.g., Richardson et al. [2007], Dupret and Piwowarski [2008], Chapelle and Zhang [2009], Guo et al. [2009], and Wang et al. [2015b]) is also relevant and could be considered as an additional baseline, especially those that leverage past cursor signals [Huang et al. 2012]. One important difference is that click prediction models are static, based on aggregated historical data. Our prefetching model is dynamic, meaning that it adapts to searcher activity on the current SERP and can be applied to any query (not just those for which we have historic data, as is the case with traditional click prediction models).

Although the use of historic data may be affected by cold start issues (where we need sufficient data to compute reliable estimates), there may be some value in exploring whether this information could be used as a prior. We do employ a baseline that ranks search results by overall clickthrough rate and prefetches the result with the highest rate. To minimize the number of features that require historic data, we do not integrate the baseline features (clickthrough rate and personal navigation) into the full model, but we plan to experiment with doing this in future work. There may also be particular scenarios where static methods perform reasonably (e.g., navigational queries, as our results suggest), and it is interesting to explore the selective application of static and dynamic methods on a per-query basis or per-searcher basis. Further work is also needed to understand the impact of applying these model enhancements on battery consumption in practice—our initial explorations of suggest that the full impact in mobile settings will be minimal, but usability tests are required to fully understand the impact of search-result prefetching.

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 [Mishra et al. 2014] where time is of the essence. 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 [Feild et al. 2013]). 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 application of data from historic activity in prefetching decisions.

11. CONCLUSIONS AND FUTURE WORK

We introduce the search result prefetching challenge and present methods for the dynamic prefetching of 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 (on desktop) and viewport changing behavior (on mobile), can substantially improve the accuracy of real-time prefetching decisions and may improve searcher efficiency—saving them a considerable amount of time per query. With increasing reliance on mobile devices and per-request resource requirements, prefetching and other anticipatory techniques will become a critical component of responsive user experiences. In future work, we will experiment with more sophisticated models to better capture the temporal dynamics and personal nature of cursor movements and viewport-based interactions. While in theory our technique may generalize beyond SERPs, optimal performance may require task-specific models tailored to the specific application scenario in which they are deployed. In addition, we will extend our analysis to a diverse range of non-SERP Web pages, refine our model target to incorporate a more holistic set of resource desiderata, enrich our features to include new sensor information as it becomes available (e.g., physiological data associated with searcher interests [White and Ma 2017]), and extend our modeling methods to more directly handle the time series of SERP interactions.

REFERENCES

Eugene Agichtein and Zijian Zheng. 2006. Identifying "best bet" web search results by mining past user behavior. In *SIGKDD*. 902–908.

Ioannis Arapakis, Xiao Bai, and B. Barla Cambazoglu. 2014. Impact of response latency on user behavior in web search. In *SIGIR*. 103–112.

Takeshi Asano, Ehud Sharlin, Yoshifumi Kitamura, Kazuki Takashima, and Fumio Kishino. 2005. Predictive interaction using the Delphian desktop. In *UIST*. 133–141.

Gökçen Aslan Aydemir, Patrick M. Langdon, and Simon Godsill. 2013. User target intention recognition from cursor position using Kalman filter. In *UAHCI*. 419–426.

Ricardo Baeza-Yates, Aristides Gionis, Flavio Junqueira, Vanessa Murdock, Vassilis Plachouras, and Fabrizio Silvestri. 2007. The impact of caching on search engines. In *SIGIR*. 183–190.

Miguel Barreda-Ángeles, Ioannis Arapakis, Xiao Bai, B. Barla Cambazoglu, and Alexandra Pereda-Baños. 2015. Unconscious physiological effects of search latency on users and their click behaviour. In *SIGIR*. 203–212.

Paul N. Bennett, Ryen W. White, Wei Chu, Susan T. Dumais, Peter Bailey, Fedor Borisyuk, and Xiaoyuan Cui. 2012. Modeling the impact of short-and long-term behavior on search personalization. In *SIGIR*. ACM, 185–194.

Pradipta Biswas, GokcenAslan Aydemir, Pat Langdon, and Simon Godsill. 2013. In *Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data*. Vol. 7947. 112–123.

Roi Blanco, Edward Bortnikov, Flavio Junqueira, Ronny Lempel, Luca Telloli, and Hugo Zaragoza. 2010. Caching search engine results over incremental indices. In *SIGIR*. 82–89.

Andrei Broder. 2002. A taxonomy of web search. In *ACM SIGIR Forum*, Vol. 36. ACM, 3–10.

Duc Hoang Bui, Yunxin Liu, Hyosu Kim, Insik Shin, and Feng Zhao. 2015. Rethinking energy-performance trade-off in mobile web page loading. In *MobiCom*. 14–26.

Christopher J. C. Burges. 2010. *From RankNet to LambdaRank to LambdaMART: An Overview*. Technical Report MSR-TR-2010-82. Microsoft Research.

Georg Buscher, Ryen W. White, Susan Dumais, and Jeff Huang. 2012. Large-scale analysis of individual and task differences in search result page examination strategies. In *WSDM*. 373–382.

Olivier Chapelle and Ya Zhang. 2009. A dynamic Bayesian network click model for web search ranking. In *WWW*. ACM, 1–10.

Edith Cohen and Haim Kaplan. 2000. Prefetching the means for document transfer: A new approach for reducing web latency. In *INFOCOM*. 854–863.

Jacob Cohen. 1977. Statistical power analysis for the behavioral sciences (revised ed.). (1977).

Anita Crescenzi, Diane Kelly, and Leif Azzopardi. 2016. *Impacts of Time Constraints and System Delays on User Experience*. 141–150.

Edward Cutrell and Zhiwei Guan. 2007a. What are you looking for? An eye-tracking study of information usage in web search. In *SIGCHI*. 407–416.

Edward Cutrell and Zhiwei Guan. 2007b. What are you looking for? An eye-tracking study of information usage in web search. In *SIGCHI*. 407–416.

Jeffrey Dean and Luiz André Barroso. 2013. The tail at scale. *Commun. ACM* 56, 2 (2013), 74–80.

Fernando Diaz, Qi Guo, and Ryen W. White. 2016. Search result prefetching using cursor movement. (2016).

Fernando Diaz, Ryen W. White, Georg Buscher, and Dan Liebling. 2013. Robust models of mouse movement on dynamic web search results pages. In *CIKM*. 1451–1460.

Josep Domenech, Bernardo de la Ossa, Julio Sahuquillo, Jose A. Gil, and Ana Pont. 2012. A taxonomy of web prediction algorithms. *Expert Syst. Appl.* 39, 9 (2012), 8496–8502.

Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. 2007. A large-scale evaluation and analysis of personalized search strategies. In *WWW*. 581–590.

Georges E. Dupret and Benjamin Piwowarski. 2008. A user browsing model to predict search engine click data from past observations. In *SIGIR*. 331–338.

Tiziano Fagni, Raffaele Perego, Fabrizio Silvestri, and Salvatore Orlando. 2006. Boosting the performance of web search engines: Caching and prefetching query results by exploiting historical usage data. *ACM TOIS* 24, 1 (2006), 51–78.

Li Fan, Pei Cao, Wei Lin, and Quinn Jacobson. 1999. Web prefetching between low-bandwidth clients and proxies: Potential and performance. In *SIGMETRICS*. 178–187.

Henry Feild, Ryen W. White, and Xin Fu. 2013. Supporting orientation during search result examination. In *SIGCHI*. 2999–3008.

Fan Guo, Chao Liu, and Yi Min Wang. 2009. Efficient multiple-click models in web search. In *WSDM*. 124–131.

Qi Guo and Eugene Agichtein. 2010a. Ready to buy or just browsing? Detecting web searcher goals from interaction data. In *SIGIR*. 130–137.

Qi Guo and Eugene Agichtein. 2010b. Towards predicting web searcher gaze position from mouse movements. In *SIGIR*. 3601–3606.

Qi Guo and Eugene Agichtein. 2012. Beyond dwell time: Estimating document relevance from cursor movements and other post-click searcher behavior. In *WWW*. 569–578.

Qi Guo, Haojian Jin, Dmitry Lagun, Shuai Yuan, and Eugene Agichtein. 2013. Mining touch interaction data on mobile devices to predict web search result relevance. In *SIGIR*. 153–162.

Brett D. Higgins, Jason Flinn, T. J. Giuli, Brian Noble, Christopher Peplin, and David Watson. 2013. Informed mobile prefetching. In *EuroSys*. 155–168.

Eric Horvitz. 1998. Continual computation policies for utility-directed prefetching. In *CIKM*. 175–184.

Jeff Huang and Abdigani Diriye. 2012. Web user interaction mining from touch-enabled mobile devices. In *HCIR Workshop*.

Jeff Huang, Ryen White, and Georg Buscher. 2012. User see, user point: Gaze and cursor alignment in web search. In *SIGCHI*. 1341–1350.

Jeff Huang, Ryen W. White, Georg Buscher, and Kuansan Wang. 2012. Improving searcher models using mouse cursor activity. In *SIGIR*. 195–204.

Jeff Huang, Ryen W. White, and Susan Dumais. 2011. No clicks, no problem: Using cursor movements to understand and improve search. In *SIGCHI*. 1225–1234.

Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR*. 154–161.

Simon Jonassen, B. Barla Cambazoglu, and Fabrizio Silvestri. 2012. Prefetching query results and its impact on search engines. In *SIGIR*. ACM, 631–640.

Dmitry Lagun, Mikhail Ageev, Qi Guo, and Eugene Agichtein. 2014. Discovering common motifs in cursor movement data for improving web search. In *WSDM*. 183–192.

Dmitry Lagun and Eugene Agichtein. 2011. Viewser: Enabling large-scale remote user studies of web search examination and interaction. In *SIGIR*. 365–374.

Dmitry Lagun, Chih-Hung Hsieh, Dale Webster, and Vidhya Navalpakkam. 2014. Towards better measurement of attention and satisfaction in mobile search. In *SIGIR*. 113–122.

David M. Lane, S. Camille Peres, Aniko Sandor, and H. Albert Napier. 2005. A process for anticipating and executing icon selection in graphical user interfaces. *Int. J. Hum.-Comput. Interact.* 19, 2 (2005), 241–252.

Edward Lank, Yi-Chun Nikko Cheng, and Jaime Ruiz. 2007. Endpoint prediction using motion kinematics. In *SIGCHI*. 637–646.

Seungyup Lee, Juwan Yoo, and Gunhee Han. 2015. Gaze-assisted user intention prediction for initial delay reduction in web video access. *Sensors* 15 (2015), 14679–14700.

Ronny Lempel and Shlomo Moran. 2003. Predictive caching and prefetching of query results in search engines. In *WWW*. 19–28.

Yiqun Liu, Ye Chen, Jinhui Tang, Jiashen Sun, Min Zhang, Shaoping Ma, and Xuan Zhu. 2015. Different users, different opinions: Predicting search satisfaction with mouse movement information. In *SIGIR*. 493–502.

Dimitrios Lymberopoulos, Oriana Riva, Karin Strauss, Akshay Mittal, and Alexandros Ntoulas. 2012. PocketWeb: Instant web browsing for mobile devices. In *ACM SIGARCH Computer Architecture News*, Vol. 40. ACM, 1–12.

Robert B. Miller. 1968. Response time in man-computer conversational transactions. In *AFIPS*. 267–277.

Nina Mishra, Ryen W. White, Samuel Ieong, and Eric Horvitz. 2014. Time-critical search. In *SIGIR*. 747–756.

Prashanth Mohan, Suman Nath, and Oriana Riva. 2013. Prefetching mobile ads: Can advertising systems afford it? In *MobiSys*. 267–280.

Atsuo Murata. 1998. Improvement of pointing time by predicting targets in pointing with a PC mouse. *Int. J. Hum.-Comput. Interact.* 10, 1 (1998), 23–32.

Rahul Murmuria, Jeffrey Medsger, Angelos Stavrou, and Jeffrey M. Voas. 2012. Mobile application and device power usage measurements. In *SERE*. 147–156.

Fiona F. Nah. 2004. A study on tolerable waiting time: How long are web users willing to wait? *Behav. Inform. Technol.* 23, 3 (2004), 153–163.

Alex Nicolaou. 2013. Best practices on the move: Building web apps for mobile devices. *Queue* 11, 6, Article 30 (June 2013), 12 pages. DOI:http://dx.doi.org/10.1145/2493944.2507894

Jakob Nielsen. 1993. *Usability Engineering*. Morgan Kaufmann.

Jakob Nielsen. 1999. User interface directions for the web. *Commun. ACM* 42, 1 (1999), 65–72.

Venkata N. Padmanabhan and Jeffrey C. Mogul. 1996. Using predictive prefetching to improve world wide web latency. *SIGCOMM Comput. Commun. Rev.* 26, 3 (1996), 22–36.

Alexandra Papoutsaki, Nediyana Daskalova, Patsorn Sangkloy, Jeff Huang, James Laskey, and James Hays. 2016. WebGazer: Scalable webcam eye tracking using user interactions. In *IJCAI*.

Abhinav Parate, Matthias Bohmer, David Chu, Deepak Ganesan, and Benjamin M. Marlin. 2013. Practical prediction and prefetch for faster access to applications on mobile phones. In *UbiComp*. 275–284.

Phillip T. Pasqual and Jacob O. Wobbrock. 2014. Mouse pointing endpoint prediction using kinematic template matching. In *SIGCHI*. 743–752.

James Pitkow and Peter Pirolli. 1999. Mining longest repeating subsequences to predict world wide web surfing. In *USITS*. 1.

John Psaroudakis and Mujtaba Khambatti. 2013. A deeper look at task completion. Retrieved from blogs.bing.com/search/2013/10/14/a-deeper-look-at-task-completion.

Judith Ramsay, Alessandro Barbesi, and Jenny Preece. 1998. A psychological investigation of long retrieval times on the world wide web. *Interact. Comput.* 10, 1 (1998), 77–86.

Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: Estimating the click-through rate for new ads. In *WWW*. ACM, 521–530.

Kerry Rodden, Xin Fu, Anne Aula, and Ian Spiro. 2008. Eye-mouse coordination patterns on web search results pages. In *SIGCHI EA*. 2997–3002.

Ramesh R. Sarukkai. 2000. Link prediction and path analysis using Markov chains. *Comput. Netw.* 33, 1 (2000), 377–386.

Eric Schurman and Jake Brutlag. 2009. Performance related changes and their user impact. In *Velocity*.

Ben Shneiderman. 1984. Response time and display rate in human performance with computers. *Comput. Surv.* 16, 3 (1984), 265–285.

Milad Shokouhi and Qi Guo. 2015. From queries to cards: Re-ranking proactive card recommendations based on reactive search history. In *SIGIR*. 695–704.

Amit Singhal and Matt Cutts. 2010. Using site speed in web search ranking. Retrieved from googlewebmastercentral.blogspot.com/2010/04/using-site-speed-in-web-search-ranking.html.

Jaime Teevan, Kevyn Collins-Thompson, Ryen W. White, and Susan Dumais. 2014. Slow search. *Commun. ACM* 57, 8 (2014), 36–38.

Jaime Teevan, Susan T. Dumais, and Daniel J. Liebling. 2008. To personalize or not to personalize: Modeling queries with variation in user intent. In *SIGIR*. 163–170.

Jaime Teevan, Daniel J. Liebling, and Gayathri Ravichandran Geetha. 2011. Understanding and predicting personal navigation. In *WSDM*. 85–94.

Narendran Thiagarajan, Gaurav Aggarwal, Angela Nicoara, Dan Boneh, and Jatinder Pal Singh. 2012. Who killed my battery: Analyzing mobile browser energy consumption. In *WWW*. 41–50.

Chao Wang, Yiqun Liu, Meng Wang, Ke Zhou, Jian-yun Nie, and Shaoping Ma. 2015b. Incorporating non-sequential behavior into click models. In *SIGIR*. 283–292.

Lidan Wang, Jimmy Lin, and Donald Metzler. 2010. Learning to efficiently rank. In *SIGIR*. ACM, 138–145.

Yichuan Wang, Xin Liu, David Chu, and Yunxin Liu. 2015a. EarlyBird: Mobile prefetching of social network feeds via content preference mining and usage pattern analysis. In *MobiHoc*. 67–76.

Ryen W. White. 2013. Beliefs and biases in web search. In *SIGIR*. ACM, 3–12.

Ryen W. White, Peter Bailey, and Liwei Chen. 2009. Predicting user interests from contextual information. In *SIGIR*. 363–370.

Ryen W. White, Paul N. Bennett, and Susan T. Dumais. 2010. Predicting short-term interests using activity-based search context. In *CIKM*. ACM, 1009–1018.

Ryen W. White and Ryan Ma. 2017. Improving search engines via large-scale physiological sensing. In *SIGIR*. ACM, in press.

Kyle Williams, Julia Kiseleva, Aidan C. Crook, Imed Zitouni, Ahmed Hassan Awadallah, and Madian Khabsa. 2016. Detecting good abandonment in mobile search. In *WWW*. 495–505.

Pingmei Xu, Yusuke Sugano, and Andreas Bulling. 2016. Spatio-temporal modeling and prediction of visual attention in graphical user interfaces. In *SIGCHI*. ACM, 3299–3310.

Qiang Yang, Haining Henry Zhang, and Tianyi Li. 2001. Mining web logs for prediction models in WWW caching and prefetching. In *SIGKDD*. 473–478.

Brian Ziebart, Anind Dey, and J. Andrew Bagnell. 2012. Probabilistic pointing target prediction via inverse optimal control. In *IUI*. 1–10.

Received June 2016; revised September 2016; accepted November 2016