Behavioral Dynamics on the Web: Learning, Modeling, and Prediction

Kira Radinsky, Technion—Israel Institute of Technology
Krysta M. Svore, Microsoft Research
Susan T. Dumais, Microsoft Research
Milad Shokouhi, Microsoft Research
Jaime Teevan, Microsoft Research
Alex Bocharov, Microsoft Research
Eric Horvitz, Microsoft Research

The queries people issue to a search engine and the results clicked following a query change over time. For example, after the earthquake in Japan in March 2011, the query *japan* spiked in popularity and people issuing the query were more likely to click government-related results than they would prior to the earthquake. We explore the modeling and prediction of such temporal patterns in Web search behavior. We develop a temporal modeling framework adapted from physics and signal processing and harness it to predict temporal patterns in search behavior using smoothing, trends, periodicities and surprises. Using current and past behavioral data, we develop a learning procedure that can be used to construct models of users' Web search activities. We also develop a novel methodology that learns to select the best prediction model from a family of predictive models for a given query or a class of queries. Experimental results indicate that the predictive models significantly outperform baseline models that weight historical evidence the same for all queries. We present two applications where new methods introduced for the temporal modeling of user behavior significantly improve upon the state of the art. Finally, we discuss opportunities for using models of temporal dynamics to enhance other areas of Web search and information retrieval.

Categories and Subject Descriptors: H.3.7 [**Information Storage and Retrieval**]: Digital Libraries; I.5.4 [**Pattern Recognition**]: Applications

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Behavioral Analysis, Predictive Behavioral Models

ACM Reference Format:
Radinsky, K., Svore, K., Dumais, S., Shokouhi, M., Teevan, J., Bocharov, A., and Horvitz, E. 2012. Behavioral Dynamics on the Web. ACM Trans. Inf. Syst. 0, 0, Article 0 (0), 37 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

The way that people use Web search engines changes over time. We explore the temporal dynamics of Web search behavior, investigating how we can model and predict changes in queries that people issue, the informational goals corresponding to the queries, and the search results that they access during Web search sessions.

In information retrieval, models that incorporate user behavior signals typically aggregate evidence over time and use it identically for all types of queries [Agichtein

Author's addresses: K. Radinsky, Computer Science Department, Technion—Israel Institute of Technology; Svore, K., Dumais, S., Teevan, J., Bocharov, A., and Horvitz, E., Microsoft Research, Redmond, WA; Shokouhi, M., Microsoft Research, Cambridge.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

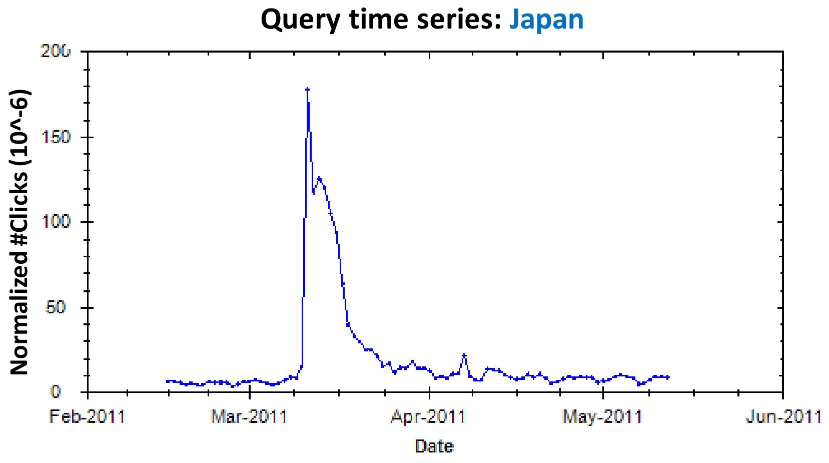


Fig. 1. A time series (2/14–5/25, 2011) for the query *japan* (normalized by overall #clicks logged on each day based on Bing query logs).

et al. 2006]. We learn to predict how the search behaviors of users change over time and use these predictive models to enhance retrieval. As an example, for a population of users, the frequency with which a query is issued and the number of times that search results are clicked for that query can change over time. In Figure 1, we show the *total clicks*[1]. We can see a dramatic change in behavior associated with this query following the Japanese earthquake on March 11th, 2011. The number of clicks surges for the query for a period of time, and then slowly decays. Likewise, the URLs that people choose to click on following the same query may vary over time, indicating a change in what people consider as relevant for that query. Figure 2 shows the change in click frequency for several popular URLs following the query *japan* around the time of the earthquake. The frequencies of access of some URLs (e.g., a US government site about Japan) mirror changes in query frequency, while others (e.g., a site for children to learn about Japan) do not change with query and click frequency.

We model and predict these different kinds of search dynamics, focusing in particular on predicting query and click frequency. Although the timing of the initial peak for the query *japan* may be hard to predict, once it is reached, the subsequent behavior can be predicted. There are many other cases in which search behaviors are easy to predict. In Figure 3, the query *halloween* exhibits periodic trends, and *android* undergoes an increasing trend in popularity, but the query *justin bieber* fluctuates with no obvious trend. Using time-series modeling, we can estimate these different trends and periodicities and predict future values of the frequency of queries and clicks. A major challenge is to select the appropriate model for predicting the dynamics. We present here a novel algorithm for selecting the best time-series model for each behavior, we refer to as the *dynamics model learner* (DML). Our model considers numerous factors for this selection, ranging from the long-term shape of the time-series to query-dependent features.

Learned models of what people search for, and then access via clicking on displayed results, can be used to improve the search experience. We shall consider two search-related applications: result ranking and query suggestion. When users information

[1]We define *total clicks* for a query q as the total number of times that any URL returned by the search engine in response to it is clicked.

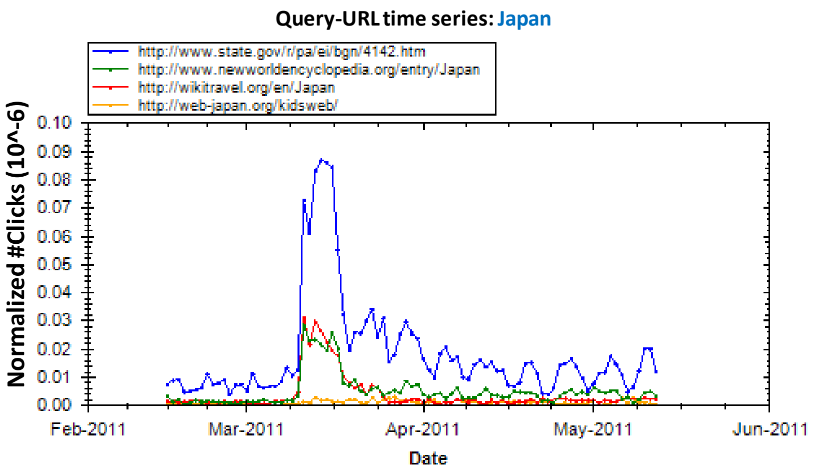


Fig. 2. Time series (2/14–5/25, 2011) for sample clicked URLs for query *Japan* (normalized by total #clicks on each day based on Bing query logs).

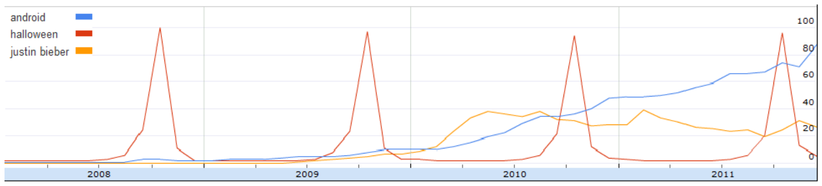


Fig. 3. Different classes of temporal trends in query frequencies.

needs change over time, the ranking of results should also change to accommodate these needs. Consider the example presented in Figure 4, which shows the frequency of URL clicks for the query *easter* at different times during the year. A user's information need several weeks before the Easter holiday (in mid-March) is likely to identify the exact date of the holiday; indeed, the URL `when-is-easter.html` is clicked more often than other Easter-related pages during mid-March. A few days before Easter, people issuing this query appear to become more interested in planning activities for the holiday, and logs of clickthroughs show that sites such as `holidays.kaboose.com` are clicked more often than other pages during this period of time. During Easter itself, people seem to be more interested in the religious meaning and customs of the holiday, questions that are answered by pages such as `answers.com/topic/easter`. To account for changes in what people click on following a query like *easter*, we explore time-aware ranking mechanisms using two ranking scenarios. The first predicts user click behavior for a given day using only this prediction to rank URLs. The second approach uses temporal user behaviors as features (along with content-based features) in a learning-to-rank algorithm. We find that weighting user behavior for each query and URL pair based on temporal dynamics significantly improves the accuracy of the ranked results in both types of ranking scenarios across several types of queries.

We also explore the application of our methods for time-aware query auto-suggestion (QAS), also called *auto completion*. Consider the example presented in Figure 5. On February 13th 2012—a day before *Valentine's day* — Google suggested *verizon wireless* as a top candidate for *v* (underlined) and did not suggest any Valentine-related

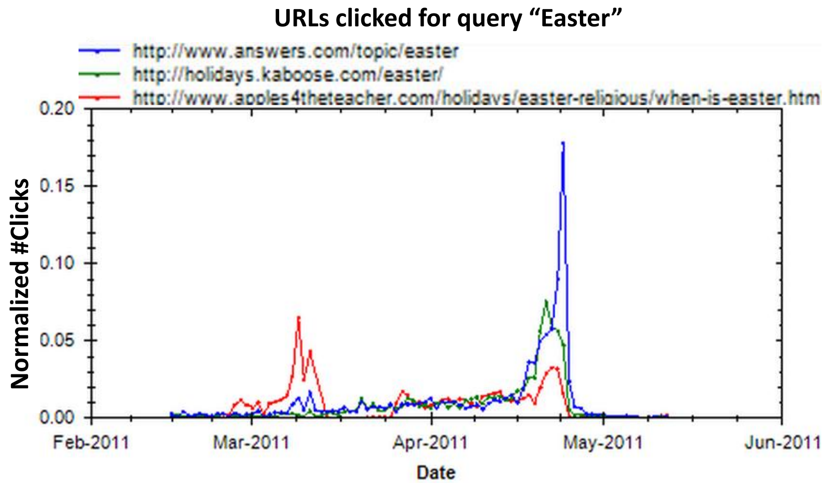


Fig. 4. Clicked-URLs behavior (2/14–5/25, 2011) for the query *easter* (normalized by the total #clicks on each day and the position of the URL).

candidates in the QAS ranking (the top plot in Figure 5).[2] The query frequency trends in the bottom plot of Figure 5 however clearly show that *valentines day* is a more relevant suggestion during this time period. We use the same time-series models to predict the query frequency and demonstrate that modeling the temporal profile of queries can improve the ranking of auto-suggestion candidates.

In summary, the key contributions described in this paper are as follows:

— We highlight the rich opportunities to study the dynamics of Web search behavior and explore several time-series models to represent and predict different aspects of search behavior over time. We discuss how these models can be learned from historical user-behavior data, and develop algorithms tailored to address several aspects of the dynamics of population behavior on the Web, including *trend, periodicity, noise,* and *surprise*.
— We present a new learning algorithm, which we refer to as the *dynamics model learner* (DML), that determines the appropriate model to use for predicting behavior, based on features extracted from large-scale logs of Web search behavior over time. We show that DML performs better than more traditional model selection techniques.
— We perform empirical evaluation of our approaches over large-scale logs of real-world user behavior (obtained from Bing), providing evidence for the value of temporal modeling techniques for capturing the dynamics observed in populations of users on the Web.
— We present applications of our temporal modeling methods to improve ranking and query suggestions, and provide evidence for the superiority of temporal modeling in both applications.

2. RELATED WORK

Several lines of research are related to modeling and predicting peoples' Web search behavior over time [Lau and Horvitz 1998]. We begin our discussion of related work with a review of studies that have characterized temporal search behavior dynamics

[2]The query was issued from the United States with personalization disabled.

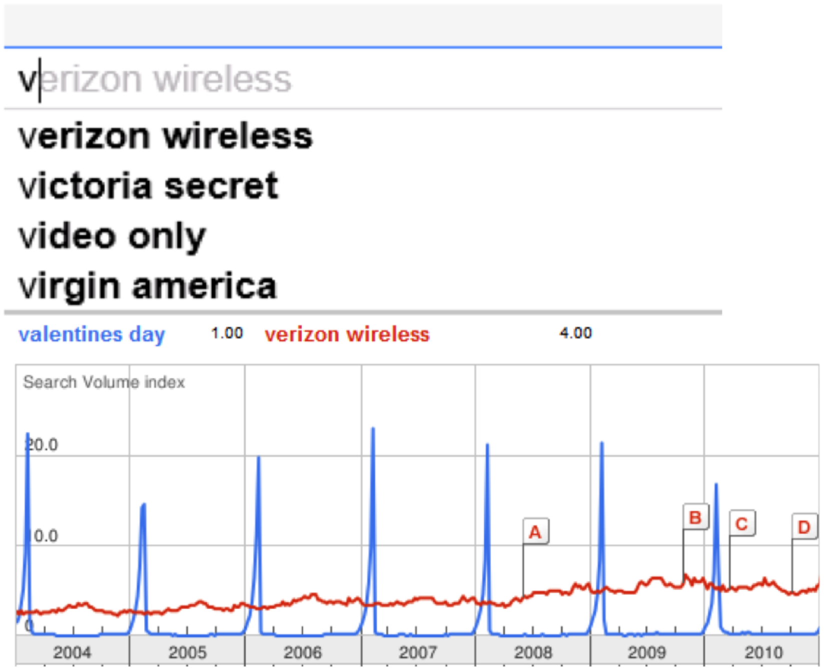


Fig. 5. (Top) The auto-suggestion candidates ranked by Google on the 13th of February 2012, a day before Valentine's Day in the USA. (Bottom) Query frequencies for *valentines day* vs. *verizon wireless* since 2004.

on the Web. We next summarize previous research that has used temporal evidence for ranking, mostly focusing on content dynamics rather than behavioral dynamics. Finally, we describe related work in automatic query suggestion.

2.1. Web Search Behavioral Dynamics

The variations of query volume over time have been studied extensively in prior work. For example, some researchers have examined changes in query popularity over time [Wang et al. 2003] and the uniqueness of topics at different times of the day [Beitzel et al. 2004]. Some studies [Jones and Diaz 2007] identified three general types of temporal query profiles: atemporal (no periodicities), temporally unambiguous (contain a single spike), and temporally ambiguous (contain more than one spike). They further showed that query profiles were related to search performance, with atemporal queries being associated with lower average precision. Kulkarni et al. [2011] explored how queries, their associated documents, and the intents corresponding to the queries change over time. The authors identify several features by which changes in query popularity can be classified, and show that presence of these features, when accompanied by changes in result content, can be a good indicator of change in the intent behind queries. Others [Chien and Immorlica 2005; Radinsky et al. 2011; Wang et al. 2007] used temporal patterns of queries to identify similar queries or words.

Vlachos et al. [2004] were among the first to examine and model periodicities and bursts in Web queries using methods from Fourier analysis. They also developed a method to discover important periods and to identify query bursts. Shokouhi [2011] identified seasonal queries using time-series analysis.

Shimshoni et al. [2009] studied the predictability of search trends using time-series analysis. Kleinberg et al. [Kleinberg 2002; 2006] developed general techniques for summarizing the temporal dynamics of textual content and for identifying bursts of terms within content.

Researchers have also examined the relationship between query behavior and events. Radinsky et al. [2008] showed that queries reflect real-world news events, and Ginsberg et al. [2009] used queries for predicting H1N1 influenza outbreaks. Similarly, Adar et al. [2007] identified when changes in query frequencies lead or lag behind mentions in both traditional media and blogs. From a Web search perspective, breaking news events are a particularly interesting type of evolving content. Diaz [2009] and Dong et al. [2010b] developed algorithms for identifying queries that are related to breaking news and for blending relevant news results into core search results. König et al. [2009] studied click prediction for news queries by analyzing the frequency and location of keywords in a corpus of news articles. Information about time-varying user behavior was also explored by Koren [2009], who used matrix factorization to model user biases, item biases, and user preferences over time.

Although much has been done to understand user Web search behavior over time, few efforts have sought to construct underlying models of this behavior and then used these models to predict future behavior. We present the construction of models for behaviors over time that can explain observed changes in the frequency of queries, clicked URLs, and clicked query-URL pairs.

2.2. Using Temporal Dynamics for Ranking

Agichtein et al. [2006] were the first to show that user behavior data could significantly improve ranking. In that work, the user behavior was represented as the simple average of behaviors over time, independent of query, URL clicks, or interactions between the two.

Researchers have examined how temporal attributes can be used to improve ranking using various kinds of content analysis. Dakka et al. [2008] defined a class of time-sensitive news queries, and suggested an approach that identifies important time intervals for those queries and augments the weight of those documents for ranking. Metzler et al. [2009] investigated a subset of temporal "year queries" – i.e., queries that often include the addition of terms representing a year such as *SIGIR 2012*, and modified the language model of the document so that that years found in the document are weighted more heavily. Similarly, Efron and Golovchinksy [2011] and Li and Croft [2003] added temporal factors into models of language likelihood and relevance for re-ranking results based on the publication date of documents. Elsas and Dumais [2010] incorporated the dynamics of content changes into document language models to improve relevance ranking, showing that there is a strong relationship between the amount of change and term longevity and document relevance. Efron [2010] considered term popularity in a document collection over time, to adjust term weights for document ranking. In summary, the prior studies cited above examine how general changes in content or specific content features (like dates) can be used to improve ranking.

Another line of research centers on methods for improving the ability of search engines to rank recent information effectively. Diaz [2009] studied how news can be integrated into search results by examining changes in query frequency, the popularity of query terms in the news collection, and query click feedback on presented news articles. Dong et al. [2010b] used Twitter data to detect and rank fresh documents. Similarly, Dong et al. [2010a] identified queries that are time-sensitive, developed features that represent the time of Web pages, and learned a recency-sensitive ranker. Dai et al. [2011] presented a ranking optimization with temporal features in documents, such as the trend and seasonality of the content changes in title, body, heading, anchor, and page or link activities. In summary, this line of research focuses on the desirability of providing fresh results for some kinds of queries.

To the best of our knowledge, temporal characteristics of queries and clickthrough behavior have not yet been used to improve general ranking of documents. In this work, we use time-series models to represent the dynamics of search behavior over time and show how this can be used to improve ranking and query suggestions.

2.3. Query Auto-Suggestion

In addition to using temporal models for ranking, we also use them to improve query auto-suggestion (QAS). Previous work on query auto-suggestion can be grouped into two main categories. The first group (also referred to as *predictive* auto-completion [Chaudhuri and Kaushik 2009]) uses information retrieval and NLP techniques to generate and rank candidates on-the-fly as the user enters new words and characters [Darragh et al. 1990; Grabski and Scheffer 2004; Nandi and Jagadish 2007]. For instance, Grabski and Scheffer [2004], and Bickel et al. [2005] studied sentence completion based on lexicon statistics of text collections. Fan et al. [2010] ranked auto-suggestion candidates according to a generative model learned by Latent Dirichlet Allocation (LDA) [Blei et al. 2003]. White and Marchionini [2007] developed a real-time query expansion system that produces an updated list of candidates based on the top-ranked documents as the user types new words in the search box.

In the second group of QAS techniques — including the one presented here — candidates are pre-generated and stored in tries and hash tables for efficient *lookup*. The list of candidate suggestions is updated by new lookups with each new input from the user. The filtering of candidates is typically based on exact prefix matching. Recently, Chaudhuri and Kaushik [2009] and Ji et al. [2009] proposed flexible fuzzy matching models that are tolerant to small edit-distance differences between the query (prefix) and candidates.

In the context of Web search, the most conventional approach is to rank candidate query suggestions according to their past popularity. Bar-Yossef and Kraus [2011] referred to this approach as *MostPopularCompletion* (MPC):

$$\boldsymbol{MPC}(\mathcal{P}) = \arg\max_{q \in \mathcal{C}(\mathcal{P})} w(q), \quad w(q) = \frac{f(q)}{\sum_{i \in \mathcal{Q}} f(i)}. \tag{1}$$

where, $f(q)$ denotes the number of times the query q occurs in a previous search log $\mathcal{Q}$. They propose a context-aware technique in which the default static scores for the candidates are combined with contextual scores based on recent session history to compute the final ranking. Under the MPC model, the candidate scores do not change as long as the same query log $\mathcal{Q}$ is used. We also take MPC [Bar-Yossef and Kraus 2011] as our QAS ranking baseline and show that it can be improved significantly by considering the temporal characteristics of queries.

Our approach is distinct from prior work in several important ways. We use time-series analysis to learn how to differentially weight historical click data for queries, URLs, and query-URL pairs (extending the work by Radinsky et al. [2012]). This enables recent behavior to be weighted more highly for some queries (or URLs or query-URL pairs) but not others; and similarly for periodic behaviors to be important for some queries but not others; etc. Second, we extend previous research on ranking fresh results by addressing the more general challenge of finding the most appropriate results, even if they are not recent. We extend earlier work by Agichtein et al. [2006] on search ranking by harnessing time-series analyses to learn the most appropriate weighting of previous behaviors rather than simply averaging previous behavior. Finally, we show that the temporal profiles of queries created by our time-series models can be used to improve the ranking of query auto-suggestion candidates that have historically been based on static measures of popularity (extending the early results by Shokouhi and Radinsky [2012] with further experiments and insights).

3. TEMPORAL MODELING OF WEB SEARCH BEHAVIOR

We now present a modeling technique based on state-space models that we use to capture the dynamics of Web behavior. Of special importance in modeling Web search behavior are global and local trends, periodicities, and surprises. We summarize in this section a general theory behind this model and discuss several modeling techniques we use to capture these important aspects. We conclude the section with a discussion of how the models can be used.

3.1. Model Framework: State-Space Models

The state-space model (SSM) is a mathematical formulation frequently used in work on systems control [Durbin and Koopman 2008] to represent a physical system as a set of input, output, and state variables related by first-order differential equations. It provides an easy and compact approach for analyzing systems with multiple inputs and outputs. The model mimics the optimal control flow of a dynamic system, using some knowledge about its state variables. These models allow for great flexibility in the specification of the parameters and the structure of the problem, based on some knowledge of the problem domain that provides information about the relations (e.g., linear) among the parameters. We use upper-case letters for matrices and lower-case for scalars. The linear space state model (with additive single-source error) defines a system behavior by the following two equations:

$$Y_t = W(\theta)X_t + \epsilon_t, \tag{2}$$
$$X_{(t+1)} = F(\theta)X_t + G(\theta)\epsilon_t, \tag{3}$$

where Y_t is the observation at time t, X_t is the state vector, ϵ_t is a noise series, and $W(\theta), F(\theta), G(\theta)$ are matrices of parameters of the model. For a longer prediction range h (also referred to as the *prediction horizon*), it is usually assumed that $Y_{t+h} = Y_t$. For simplicity, in the following sections we present equations for $h = 1$. We shall assume (as commonly assumed in representations of dynamics in natural systems) that ϵ_t are independent and identically distributed following a Gaussian distribution with variance σ^2 and mean 0. Equations (2) and (3) are called the measurement and transition equations, respectively. To build a specific SSM, a structure for the matrices $W(\theta), F(\theta), G(\theta)$ is selected, and the optimal parameters θ and σ and an initial state X_0 are estimated.

The SSM representation encompasses all linear time-series models used in practice. We show in this section how the SSM can be applied to model query and click frequency in Web search. We model the state X_t using a trend component and a seasonal (or periodicity) component, as is often done for modeling time series [Hyndman et al. 2008]. The trend represents the long-term direction of the time series. The seasonal component is a pattern that repeats with a known periodicity, such as every week or every year. We first present models for user search behavior using just historical smoothing, and then present models with trend, periodicity, and their combination. Finally, we explore models that incorporate notions of unexpected observations or *surprises*. Each model is represented by setting the scalars of the matrices $W(\theta), F(\theta), G(\theta)$.

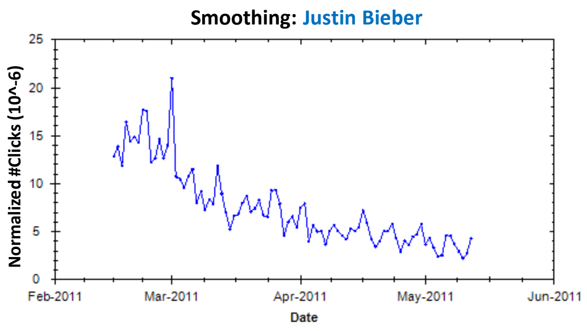


Fig. 6. Query exhibiting behavior where historical data has little relevance for predicting future clicks (normalized by overall #clicks on each day based on Bing query logs).

3.2. Modeling with Smoothing (SMT)

For the query *justin bieber* in Figure 6, simple averaging of past frequencies may give too much weight to the relatively high popularity of the query before April, and hence if used for forecasting, it is likely to overestimate the future popularity.

Therefore models that simply average historical data, and then extrapolate a constant value as a prediction, may perform poorly for predicting the future. The simple moving average technique, also called the simplest Holt-Winters model [Holt 2004], is a technique for producing an exponentially decaying average of all past examples, thus giving higher weights to more recent events. The model is represented by the following equation,

$$y_{t+1} = \alpha \cdot x_t + (1 - \alpha) \cdot y_t.$$

Here, for $y = x_0$, solving the recursive equation as follows,

$$y_{t+1} = \alpha x_t + \ldots + \alpha(1-\alpha)^{k-1} x_{t-k} + \ldots + (1-\alpha)^t x_0$$

produces a prediction y_{t+1} that weights historical data based on exponentially decay according to the time distance in the past. The parameter α is estimated from the data (see Section 3.7). Converting to SSM notation, let $l_t = y_{t+1}$ where l_t is the level of the time series at time t and $\epsilon_t = x_t - y_t$. The measurement and transition equations can be defined as:

$$\begin{aligned} Y_t = y_t = l_{t-1}, \\ X_t = l_t = l_{t-1} + \alpha\epsilon_t. \end{aligned} \tag{4}$$

In this case, $W = (1), F = (1), G = (\alpha)$.

3.3. Modeling Trends (TRN)

In many cases, such as the query *harold camping* in Figure 7, using only one coefficient to discount previous data does not have the expressiveness to capture the dynamics of the system. The figure shows the query-click behavior for the query *harold camping*, who predicted that the end of the world would commence on May 21st, 2011. The growing interest in this prediction in the proximity of this date shows a clear local growth trend in the time series. In such cases, where the time series exhibits a local trend, simple smoothing of historical data cannot accurately capture the dynamics of

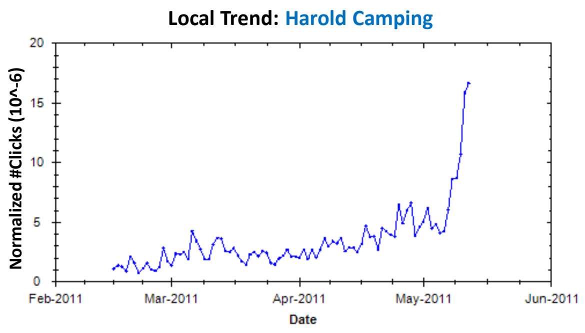


Fig. 7. Query exhibiting behavior with local trend (normalized by overall #clicks on each day based on Bing query logs).

interest in the topic. A potential solution to this problem is the addition of a trend component b_t to the previously described model:

$$\begin{aligned} y_t &= l_{t-1} + d \cdot b_{t-1} + \epsilon_t, \\ l_t &= l_{t-1} + b_{t-1} + \alpha\epsilon_t, \\ b_t &= b_{t-1} + \beta^*(l_t - l_{t-1} - b_{t-1}), \end{aligned} \tag{5}$$

where l_t is called the level of the time series at time t, d is the *damping factor*, and b_t is the *estimation of the growth of the series* at time t, which also can be written as

$$b_t = b_{t-1} + \alpha\beta^*\epsilon_t = b_{t-1} + \beta\epsilon_t.$$

Let $X_t = (l_t, b_t)'$, then:

$$Y_t = (1 \;\; d)\, X_{t-1} + \epsilon_t,$$
$$X_t = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} X_{t-1} + \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \epsilon_t.$$

In this case, $W = (1 \;\; d)\,, F = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, G = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$.

The parameters α, β are estimated from the data (see Section 3.7).

3.4. Modeling Periodicity (PRD)

Figure 8 shows query-click behavior for the query *consumer report* that exhibits weekly periodicity. Similarly, the query *halloween* in Figure 3 exhibits annual periodicity. For such queries, predictions based only on local trends or smoothing of the data will perform badly during the peaks. A possible solution is the addition of a periodic or seasonal component s_t to the simple Holt-Winters model,

$$\begin{aligned} y_t &= l_{t-1} + s_{t-m} + \epsilon_t, \\ l_t &= l_{t-1} + \alpha\epsilon_t, \\ s_t &= \gamma^* \cdot (y_t - l_{t-1}) + (1 - \gamma^*)s_{t-m}, \end{aligned}$$

where m is the periodicity parameter that is estimated based on the data along with other parameters (see Section A.1 for details). In SSM notation, the equation system can be written as:

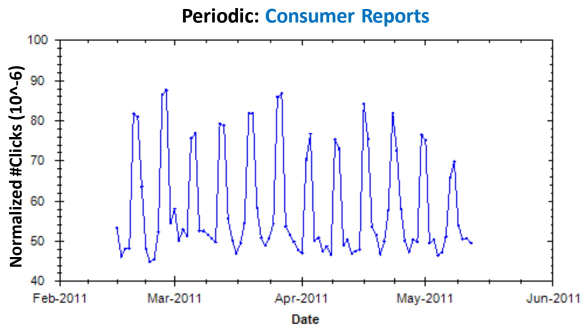


Fig. 8. Query exhibiting a periodic behavior (normalized by overall #clicks on each day based on Bing query logs).

$$\begin{aligned} y_t &= l_{t-1} + s_{t-m} + \epsilon_t, \\ l_t &= l_{t-1} + \alpha\epsilon_t, \\ s_{t-i} &= s_{t-i+1}, \\ &\dots, \\ s_t &= s_{t-m} + \gamma\epsilon_t, \end{aligned} \tag{6}$$

and for $X_t = (l_t, s_1, \dots, s_m)$, we can represent the parameters F, G, W in a form of the matrices similar to the Trend Holt-Winters model formulation. The parameters α, γ are estimated from the data (see Section 3.7).

3.5. Modeling Trends and Periodicity (TRN+PRD)

In the previous models, trend and periodicity were considered separately. However, for many queries, such as the query *vampire diaries* shown in Figure 9, the trend and periodicity components are mixed. In this case, the periodicity is unchanged but the frequency increases. The addition of trend and periodicity parameters produces the following model:

$$\begin{aligned} y_t &= l_{t-1} + d \cdot b_{t-1} + s_{t-m} + \epsilon_t, \\ l_t &= l_{t-1} + b_{t-1} + \alpha\epsilon_t, \\ b_t &= b_{t-1} + \beta\epsilon_t, \\ s_t &= s_{t-m} + \gamma\epsilon_t, \\ &\dots, \\ s_{t-i} &= s_{t-i+1}. \end{aligned} \tag{7}$$

The parameters α, β, γ are estimated from the data (see Section 3.7).

3.6. Modeling Surprises (SRP)

The Holt-Winters models assume the conversion of a set of parameters to a single forecast variable Y_t. However, in many real-world time series, the model itself changes over time and is affected by external disturbances caused by unmodeled, exogenous processes in the open world. For example, in Figure 10, we see a periodic query *fda* with a disturbance on March 4th, due to an external event concerning the announcement that some prescription cold products are unsafe. Inclusion of such outliers might

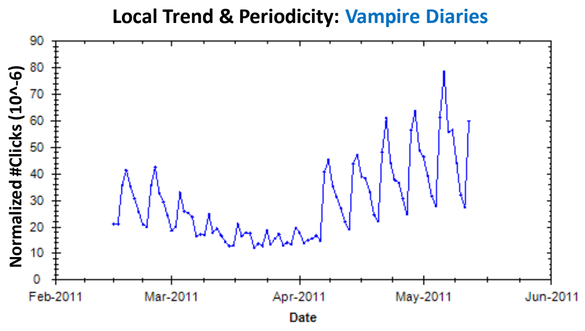


Fig. 9. Query exhibiting periodic behavior with local trend (normalized by overall #clicks on each day).

have a strong effect on the forecast and parameter estimation of a model. We wish to identify characteristics in the temporal patterns which are not adequately explained by the fitted model, and try to model them for a better estimation of Y_t. If these characteristics take the form of sudden or unexpected movements in the series, we can model them by the addition of *disturbances* that capture the occurrences of *surprises* from the perspective of the model.

A disturbance or a *surprise* is an event which takes place at a particular point in the series, defined by its location and magnitude. In a time series, the effect of a disturbance is not limited to the point at which it occurs, but also propagates and creates subsequent effects that manifest themselves in subsequent observations.

We augment the standard Holt-Winters model with the addition of two *surprise parameters*: m_t, which is a surprise measurement at time t, and k_t, which is the surprise trend at time t:

$$\begin{aligned} y_t &= l_{t-1} + d \cdot b_{t-1} + s_{t-m} + m_t + \epsilon_t, \\ l_t &= l_{t-1} + d \cdot b_{t-1} + \alpha\epsilon_t, \\ b_t &= b_{t-1} + k_t + \beta\epsilon_t, \\ s_t &= s_{t-m} + \gamma\epsilon_t, \\ &\dots, \\ s_{t-i} &= s_{t-i+1}. \end{aligned} \tag{8}$$

We discuss in Section A.2 methods for identifying the surprises k_t in a time series, and in Section 3.7 discuss how the parameters α, β, γ are estimated from the data.

3.7. Using the Models to Forecast

Once the model structure is specified, the distribution of the future values of the time series can be evaluated, given past history. That is, we learn an SSM for time series $Y_1, \dots, Y_n$ jointly with the internal states $X_0, \dots, X_n$, and residuals $\epsilon_0, \dots, \epsilon_n$. During prediction, future states X_{n+1} are generated using the state transition equation (Eq. 3) and, based on these, a distribution of the future values Y_{n+1} is generated using the measurement equation (Eq. 2). The final prediction can be generated by using the expected value of this distribution.

The SSM family provides a predefined structure for forecasting, where the specific parameters of the model need to be evaluated from the data. We apply gradient descent [Snyman 2005] to optimize the model parameters based on training data. We assume here the following loss function, which is a common criterion for measuring forecast

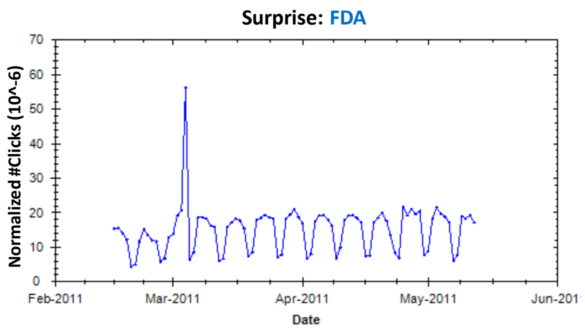


Fig. 10. Query exhibiting behavior with surprises.

error:

$$F = \sum_{t=1}^{T} \epsilon_t^2. \tag{9}$$

The initial values of the models, X_0, are set heuristically, and refined along with the other parameters. The seasonal component m is estimated from the data by *autocorrelation* analysis (see Section A.1 for details).

4. LEARNING THE RIGHT TEMPORAL MODEL

As described in Section 3, different temporal models can be employed to represent user behaviors over time. We first present a known method for selecting which model is the best based on the information criterion of the time series (Section 4.1), and then provide a novel method for inferring the model based on extended and domain-specific characteristics of Web behaviors (Section 4.2).

4.1. Bayesian Information Criterion

The models described thus far are based on curve fitting and we have shown via examples that there is no single model that adequately models trends, periodicities, and surprise. An alternative approach would be to *fit* several models, and train a classifier that selects the best predictive model.

Each model adds more parameters. When fitting the models, there is a high likelihood of more complex models fitting the data better. This might result in overfitting, especially when applying gradient descent. The Bayesian information criterion (BIC) [Schwarz 1978] resolves this problem by adding a penalty for the number of parameters in the model. That is, it presents a tradeoff between the accuracy and the complexity of the model. It is closely related to the Akaike information criterion (AIC), but the penalty term is larger in BIC than in AIC. The BIC criteria being optimized is defined as:

$$\mathrm{BIC} = -2 \cdot \log(L) + q \cdot \log(n), \tag{10}$$

where q is the number of parameters, n is the length of the time series, and L is the maximized likelihood function. For a Gaussian likelihood, this can be expressed as

$$\mathrm{BIC} = n \cdot \log(\sigma_e^2) + q \cdot \log(n), \tag{11}$$

where σ_e is the variance of the residual in the testing period (estimated from the test data). The model with the lowest BIC is selected to represent the time series and to issue point forecasts.

4.2. Dynamics Model Learner (DML)

The BIC criterion introduced in Section 4.1 takes only the model behavior on the time-series values into account. However, in our domain we have access to richer knowledge about search behavior. For example, we know what query was issued, the number of clicks on a given URL for that query, and so on. We shall now discuss how to use domain knowledge to further improve behavior prediction. We focus on learning which of the trained temporal SSM models is most appropriate for each object of interest, and then estimating parameters for the chosen model.

4.2.1. Going Beyond Time-series for Temporal Modeling. We start by formally motivating the algorithm and defining the learning problem. Let T be the discrete representation of time and O be the set of objects, and let $f_i : O \times T \rightarrow F(f_i)$ be a set of features. For example, O can be a set of URLs, $f_1(o, t)$ can be the number of times URL o was clicked at time t, and $f_2(o, t)$ can be the dwell time on the URL at time t.

In time-series analysis, a *single object* o is modeled over some period $t_1, \ldots, t_n$, and a model $C : T \rightarrow \mathbb{R}$ can be trained based on those historical examples. In order to forecast future trends (classes) at time t_{i+1}, the model is provided with an example of the object seen at training time t_i (for example, predicting how many times the URL o is clicked on tomorrow based on its past history).

In regression learning, *multiple objects* $O' \subset O$ are modeled simultaneously by a single model $C : O \rightarrow \mathbb{R}$. During prediction, the regression model may be given an example of an object o_i, that has not been seen before, to produce the prediction of its numeric value. Notice that no notation of time is considered.

The time-series approach is capable of making specific predictions about a specific object at a certain time, but does not consider information about other objects in the system and therefore cannot generalize based on their joint behaviors. Regression learning, on the other hand, generalizes over multiple objects, but does not use the specific information about the object it receives during prediction, and therefore does not usually use the information about how this specific object behaves over time.

We combine the two approaches into a unified methodology that first considers generalized information about other objects to choose a model of prediction and then uses the specific knowledge of the predicted object to learn the specific parameters of the model for the object. Formally, given a set of objects $O' \subset O$ over some period of time $t_0, \ldots, t_n$, we produce a model C that receives an object $o \in O$ (not necessarily $o \in O'$) over some period $t_0, \ldots, t_n$, and produces the prediction of its numeric value at time t_{n+1}.

4.2.2. DML Learning. In this section, we present the *dynamics model learner (DML)* algorithm for learning from multiple objects with historical data. Let $O' \subset O$ be a set of objects given as examples for training the learning model. Let $t_1, \ldots, t_{n+\sigma}$ be the times dedicated for training the model. For example, O' can be a set of queries for which we have user behavior information for the period of time $t_1, \ldots, t_{n+\sigma}$. We divide the objects into two sets — the learning set, $t_1, \ldots, t_n$, and the validation set $t_{n+1}, \ldots, t_{n+\sigma}$. For every temporal model described in Section 3, we train a model on the learning period, and check the mean squared error (MSE) over the validation period. Formally, let $o(t)$ be the behavior of object o at time t (e.g., how many times the query o was searched), then

$$MSE(o, t_1, \ldots, t_{n+\sigma}, m) = \frac{\sum_{t=t_{n+1}}^{t_{n+\sigma}} (o(t) - \widehat{o}(t))^2}{\sigma},$$

where $\widehat{o}(t)$ is the model m estimation at time t.

Let i be the index of the model with the lowest MSE on the test period for the object o. We construct a set of examples $E = \{\langle f_1(o,t), \ldots, f_n(o,t)\rangle, i | o \in O'\}$ — a vector representing an object and labeled with the index of the best-performing model for that object. We then use a learner (in our experiments, a decision-tree learner) along with the examples E to produce a classifier C. During prediction, C is applied on the object we wish to predict, o_{target}. The output $m = C(o_{target})$ represents the index of the most appropriate model for the object o_{target}. We train the model m using the behavior of o_{target} during $t_1, \ldots, t_{n+\sigma}$. A detailed algorithm is illustrated in Figure 11. An example of a learned decision tree is shown in Figure 12. In this figure, we see that the periodic model should be applied only on objects (queries) considered periodic (as defined in Section A.1), along with other characteristics. If the query is not periodic, the trend or the smoothing model should be applied, depending on the query shape (see Section 4.2.3). Thus, by using the DML model we learn to apply the best equations for modeling the time series.

```
Procedure DYNAMICS MODEL LEARNER($O'$, $Learner$)
  Train:
    Train temporal models $m_1, \ldots, m_M$ for every $o \in O'$
    $E = \{ \langle f_1(o,t), \ldots, f_n(o,t)\rangle,$
            $\arg\min_i MSE(o, t_1, \ldots, t_{n+m}, m_i) | o \in O'\}$
    Call Learner with $E$, and receive the
            hypothesis classifier $C$
  Prediction: Given unlabeled instance $o_{target}$ at time $t$
    chosenModel ← Evaluate $C$ on
            $\langle f_1(o_{target}, t), \ldots, f_n(o_{target}, t_i)\rangle$
    learntModel ← Learn chosenModel parameters
            using $o_{target}(t_1), \ldots, o_{target}(t_{n+m})$
    Return Evaluate learntModel on $o_{target}(t)$
```

Fig. 11. The procedure estimates the model type to learn based on past examples and their features. The model parameters are estimated and a prediction for the new object is performed.

4.2.3. DML Features. DML uses a set of features f_i about each object o. In this section, we discuss the specific features we use to train the DML model used in our experiments. We devise a total of 973 features (description of the features is available online [3]), and group them into three groups: aggregate features of the time series o, shape features of the time series o, and other domain-specific features such as the query class.

Aggregate Features. Features include the *average*, *minimum*, *maximum*, and *period* of the time series, e.g., the average of the query volume. Other features consider the dynamics of the series, e.g., the time series *periodicity* and *number of surprises*. We also consider the *size of the spikes* during the surprises (the magnitude of the disturbance).

[3] `http://www.technion.ac.il/~kirar/Datasets.html`

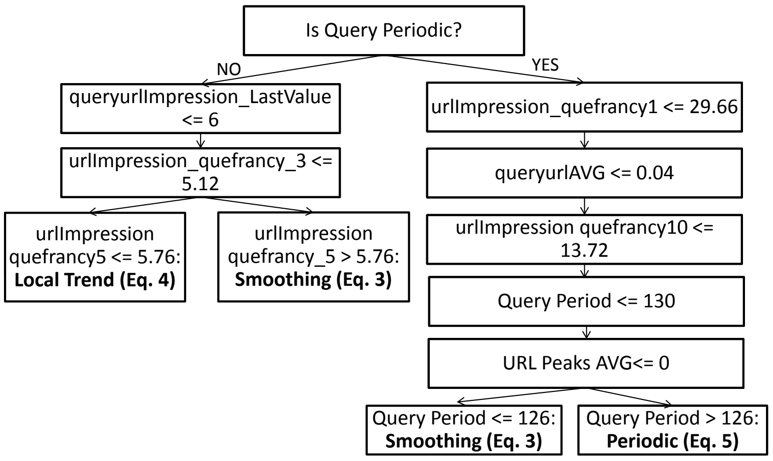

Is Query Periodic?
NO
YES
queryurlImpression_LastValue <= 6
urlImpression_quefrancy_3 <= 5.12
urlImpression quefrancy5 <= 5.76: Local Trend (Eq. 4)
urlImpression quefrancy_5 > 5.76: Smoothing (Eq. 3)
urlImpression_quefrancy1 <= 29.66
queryurlAVG <= 0.04
urlImpression quefrancy10 <= 13.72
Query Period <= 130
URL Peaks AVG<= 0
Query Period <= 126: Smoothing (Eq. 3)
Query Period > 126: Periodic (Eq. 5)

Fig. 12. A part of the learned dynamic model.

Shape Features. Shape features represent the shape of the time series. Our goal is to produce a representation that is not sensitive to shifts in time or to the magnitude of the differences. Formally, for a time series $y[n] = x[n]$, we are looking for a representation that will be equivalent to series of the form $y[n] = x[n - h]$ and $y[n] = A \cdot x[n]$, for any shift h and any scalar A. Homomorphic signal processing is a solution that satisfies these conditions. Intuitively, application of the Finite Fourier transform (FFT) operator on a time series transforms it to what is called the *spectral domain*, i.e., produces a vector x of size k, where every x_k represents the k-th *frequency* of the time series.

$$x_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}}$$

This procedure eliminate shifts, as both $y[n] = x[n - h]$ and $y[n] = x[n]$ have similar frequencies. Application of a log function on the resulting frequency vector and an additional FFT operator transforms it to what is called the *cepstral domain* [Childers et al. 1977], and the values of x in this product are called *quefrencies*. The byproduct of these procedures is that series of the form $y[n] = A \cdot x[n - h]$ and $y[n] = x[n]$ are transformed to the same vector X in the ceptral domain. In speech recognition [Bogert et al. 1967], the values of $x_1, \ldots, x_{13}$ in the cepstral domain are considered a good representation of the series. We consider these features to represent the shape of the series.

Domain-Specific Features. We also consider a set of temporal and static features that are domain specific. For the query-click time series we consider the total *number of clicked URLs*, and the *query-click entropy* at time t, which is defined as:

$$QueryClickEntropy(q, t) = -\sum_{i=1}^{n} p(click_t(u_i, q)) \log p(click_t(u_i, q)), \qquad (12)$$

where $u_1, \ldots, u_n$ are the clicked URLs at time t, and $p(click_t(u_i, q))$ is the percentage of clicks on URL u_i for query q at time t, among all clicks on query q. If all users click on the same URL for query q, then $QueryClickEntropy(q, t) = 0$. We consider an aggregated version of query-click entropy as the average of the last $k = 7$ days before the prediction. For both the query and URL click time series, we consider the *topical distribution* of the URL or query. We used a standard topical classifier which classifies queries into topics based on categories from the Open Directory Project (ODP) [Bennett et al. 2010]. We classify queries into approximately 40 overlapping topics, such as travel, health, and so on.

5. OVERVIEW OF APPLICATIONS OF TIME-SERIES MODELING FOR SEARCH

Models that take advantage of historical user behavior data often aggregate the data uniformly, regardless of when the behavior is observed. These techniques fail to weight older data differently than newer data and may lead to a stale user experience. We now explore how the time-series modeling techniques presented in previous sections can be applied to resolve this problem. We first summarize how we predict future clicks and then describe two important search-related applications: ranking and query auto-suggestion. In Sections 6, 7, and 8 we provide the experimental results of those applications.

5.1. Experiments for Predicting Future Clicks

We start with an application of the methods we described for click prediction, and investigate three types of forecast: query click prediction, query-dependent URL click prediction, and query-independent URL click predictions. We extend the early results presented in Radinsky et al. [2012]. We provide a deep analyze the performance of different temporal models on the different types of predictions and their behavior for different prediction horizons (Section 6).

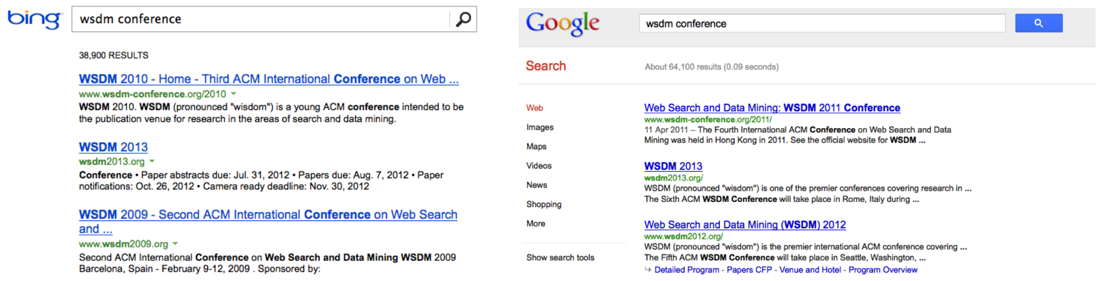
5.2. Experiments for Ranking Search Results

Web search engines often rely on usage data such as clicks and anchor text for ranking documents. Such techniques tend to favor older documents that have accumulated more behavioral data over time over fresher and potentially more relevant documents. As an example (shown in Figure 13), one of the highest ranked results for the query *WSDM conference* at the time of writing this paper is the WSDM 2010 (on Bing) or WSDM 2011 page (on Google). However, the current intention behind this query is more likely about finding information on the forthcoming WSDM 2013 conference. The older pages have more historical data than the more recent one, which only has sparse click data because it did not even exist prior to 2012. Accurate predictions of future query and click frequency can be used directly to re-ranking the search results. We refer to those predictions as *temporal features*. Alternatively, those features can be used along with other features to train a ranking function, e.g., BM25 features [Robertson et al. 2004]. We refer to those features as *base features*.

We explore both of these scenarios. In our first set of ranking experiments, we leverage temporal features as independent evidence for ranking. To understand how well our predictions can be used as independent evidence for ranking, we only use our prediction of future user behavior. For each query-URL pair we compute the predicted normalized number of clicks and use that prediction to rank the URLs.

In the second set of ranking experiments, we use temporal features as input to a learning-to-rank algorithm. We employ a supervised learning technique to learn the ranking function that best predicts the ground-truth ranking of URLs for a given query, using a variety of query, query-URL and URL features, combining both base features and temporal features. We show that rankers that use temporal modeling consistently outperform rankers only considering static user behavior.

We provide an analysis of the temporal models applied to ranking search results in Section 7.



38,900 RESULTS

WSDM 2010 - Home - Third ACM International **Conference** on Web ...
www.**wsdm-conference**.org/2010
WSDM 2010. **WSDM** (pronounced "wisdom") is a young ACM **conference** intended to be the publication venue for research in the areas of search and data mining.

WSDM 2013
wsdm2013.org
Conference • Paper abstracts due: Jul. 31, 2012 • Papers due: Aug. 7, 2012 • Paper notifications: Oct. 26, 2012 • Camera ready deadline: Nov. 30, 2012

WSDM 2009 - Second ACM International **Conference** on Web Search and ...
www.**wsdm**2009.org
Second ACM International **Conference** on **Web Search and Data Mining WSDM** 2009 Barcelona, Spain - February 9-12, 2009 . Sponsored by:

About 64,100 results (0.09 seconds)

Web Search and Data Mining: **WSDM** 2011 **Conference**
www.**wsdm-conference**.org/2011/
11 Apr 2011 – The Fourth International ACM **Conference** on Web Search and Data Mining was held in Hong Kong in 2011. See the official website for **WSDM** ...

WSDM 2013
wsdm2013.org/
WSDM (pronounced "wisdom") is one of the premier conferences covering research in ...
The Sixth ACM **WSDM Conference** will take place in Rome, Italy during ...

Web Search and Data Mining (**WSDM**) 2012
www.**wsdm**2012.org/
WSDM (pronounced "wisdom") is the premier international ACM conference covering ...
The Fifth ACM **WSDM Conference** will take place in Seattle, Washington, ...
↳ Detailed Program - Papers CFP - Venue and Hotel - Program Overview

Fig. 13. Search results for the query *WSDM conference* on August 5th 2012. The 2010 (left image) and 2011 (right image) conference websites are ranked higher than the 2013 conference website.

5.3. Experiments for Ranking Query Auto-Complete Candidates

Query auto-suggestion (QAS) is a feature incorporated in most search engines, where the goal is to save user time by predicting user's intent and suggesting other possible queries matching the first few keystrokes typed. In a typical QAS scenario, the user is presented with a list of query suggestions that match the prefix text entered in the search box so far (e.g., "di" in Figure 14). For each prefix $\mathcal{P}$, the list of candidates $\mathcal{C}(\mathcal{P})$ consists of all previous queries that start with $\mathcal{P}$[4]. The list of candidates is dynamically updated at run-time with each new character typed by the user.

The common practice for ranking QAS candidates is to use past query frequencies [Bar-Yossef and Kraus 2011; Chaudhuri and Kaushik 2009] aggregation as a proxy for the *expected* popularity in the future. Bar-Yossef and Kraus [2011] referred to this general form of QAS ranking as *MostPopularCompletion* (MPC). Those approaches assume that user intent is static and does not change over time. However, the query popularity is dynamics and affected by different temporal trends. Consider the example in Figure 14 where a user has typed *di* in Google query box on Sunday, November 6th, 2011. At first glance, knowing that *dictionary* is generally a more frequent query than *disney*, it might be difficult to notice how the ranking might be improved. However, looking at the daily trends for these queries in Figure 15 reveals that *disney* is more popular on weekends. Hence, given that the first snapshot was taken on a Sunday, swapping *disney* and *dictionary* could lead to a better ranking at the time of this query. In summary, the past is not always a good proxy for future particularly for trendy and seasonal queries. Today's frequency for query *dictionary* is not necessarily the best estimate for its frequency tomorrow.

We propose to use our temporal modeling techniques to enhance this ranking. In our time-sensitive QAS ranking model, the score of each candidate at time t is determined according to its predicted value calculated using time-series models that capture temporal trends and periodicity. Our time-sensitive QAS ranking model can be formalized as a variation of MPC model (Eq. 1),

$$\boldsymbol{TS}(\mathcal{P},t) = \arg\max_{q\in\mathcal{C}(\mathcal{P})} w(q|t), \quad w(q|t) = \frac{\hat{y}_t(q)}{\sum_{i\in\mathcal{Q}}\hat{y}_t(i)}, \tag{13}$$

where $\mathcal{P}$ is the entered prefix, and $\mathcal{C}(\mathcal{P})$ represents its list of QAS candidates, and $\hat{y}_t(q)$ denotes the estimated frequency of query q at time t.

[4]Without loss of generality, we ignore more advanced fuzzy matching techniques [Chaudhuri and Kaushik 2009; Ji et al. 2009] in our work.

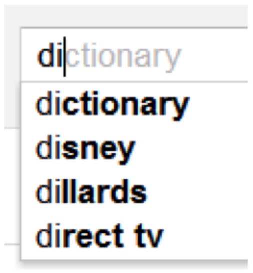


Fig. 14. Google auto-suggestion candidates after typing *di* on Sunday, February 13th, 2012. The user was typing from a US IP address, with personalization turned off.

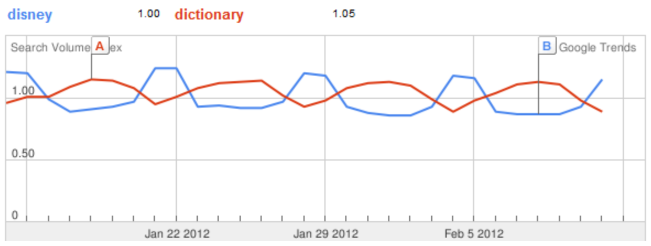


Fig. 15. Daily frequencies for queries *dictionary* (red) and *disney* (blue) during January 2012 according to Google Trends (the snapshot was taken on Monday, 13-Feb-2012). Among the two queries, *disney* is more popular on weekends, while *dictionary* is issued more commonly by users on weekdays.

We extend the early work by Shokouhi and Radinsky [2012] and provide a deeper analysis of numerous temporal models applied for ranking query auto-complete candidates in Section 8.

6. EXPERIMENTS FOR PREDICTING FUTURE CLICKS

We first describe the setup for the prediction experiments and perform several prediction experiments, namely predicting query, query-dependent URL click, and query-independent URL click frequencies. In every experiment, five SSM models (Section 3), two selection models (BIC (Section 4.1), DML (Section 4.2)), and four baseline models (Section 6.3.1) are evaluated. The prediction results are shown for a variant of the MSE to avoid numerical errors: $MSE(predicted) = E[|predicted - real|^{0.5}]$. The above error is averaged over 12 consecutive prediction days (April 14th to April 25th, 2011) to avoid over-fitting of a specific day. To compare the results of the different algorithms, we perform a t-test on the results.

6.1. Data

The dataset consists of query and URL activity obtained from Bing for the US market during the period December 15th, 2010 to April 25th, 2011. The data contains information about a query's daily click counts, a URL's daily click counts, and, for each query, all of the URLs presented along with their corresponding daily rank positions and daily click counts. We filter the data and consider only queries and URLs that have more than five clicks per day. For each query, we consider the top four URLs by click frequency. We normalize every activity time series by the total number of activities on that day, to mitigate known differences in daily query volume. For query and URL pairs, we also normalize by the number of clicks on the URL at the position at which it

Table I. Summary of query types.

Data	#Queries	#URLs	Description
General	10000	35862	General queries randomly sampled (unique on a given day)
Dynamic	504	1512	Queries labeled as requiring fresh results
Temporal Reformulations	330	1320	Queries reformulated with temporal word added
Alternating	1836	7344	Randomly sampled queries whose URLs change rankings

was displayed in the displayed ranked results for the query, producing a value which is not dependent on the position.

We describe the dataset in full detail in Section 6.2. Table I gives a summary of the data.

6.2. Queries

We investigate the effectiveness of different models on various types of queries. For this purpose, we use multiple sampling strategies to collect queries with different degrees of time sensitivity.

General Queries. We first present prediction over a general sample of queries issued to Bing. For this purpose, we use 10,000 queries randomly sampled without repetition on a given day, and 35,862 clicked URLs.

Time-series modeling is especially interesting in cases where the behavior of the population of users changes over time. To study such changes, we identified three types of queries that we believe would benefit from temporal modeling.

Dynamic Queries. Queries, such as *japan* described earlier, arise because of external events and require fresh content to satisfy users' needs. Trained judges labeled queries that required fresh results at specific points in time. We say that a query q is *Dynamic* if a human labeled it at time t as a query requiring fresh results. In the experiments, a total of 504 dynamic queries and 1512 labeled URLs were used.

Temporal-Reformulation Queries. Another way of identifying queries associated with time-sensitive informational goals is to examine queries that explicitly refer to a period of time. We focused on queries that were reformulated to include an explicit temporal referent (e.g., an initial query *world cup* might be later reformulated as *world cup 2011* or *world cup latest results*). We say that a query q was reformulated to a query $q' = q + w$ if a user issuing a query q at time t issued the query q with an additional word w at time $t + 1$ in the same session. We say that a query q was *temporally reformulated* if w is of type year, day of week, or if w is one of the following words: current, latest, today, this week. Reformulation data was obtained for queries issued in the years 2007-2010. A total of 330 queries and 1320 URLs were sampled.

Alternating Queries. Queries that exhibit interesting temporal behavior often show changes in the URLs that are presented and clicked on over time. We focus on a subset of queries, whose most frequently clicked URLs alternate over time. We say that a query q is alternating if $\exists t_1, t_2 \in T, i, j \in N : Click(u_i, t_1|q) > Click(u_j, t_1|q), Click(u_i, t_2|q) < Click(u_j, t_2|q)$, where $u_1, \ldots, u_n$ are the matching URLs to the query q, and $Click(u, t|q)$ is the number of clicks on u at time t for the query q. A total of 1836 queries and 7344 URLs were sampled.

6.3. Models

6.3.1. Baseline Methods. The most commonly used baseline for representing user search behavior is the averaging of activity over some period of time. Generally speaking, we consider baselines that perform some kind of uniform or non-uniform mapping of the data, and output the average of the mapping as the prediction. We call these different mappings *Temporal Weighting Functions*. The modeling in this case is of the form

$$y_t = \sum_{i=0}^{t-1} \frac{w(i, y_i) y_i}{\sum_{j=0}^{t-1} w(j, y_j)},$$

where $w(i, y_i)$ is a temporal weighting function. In this work, we consider the following baseline functions:

(1) **AVG**: Simple average of the time series: $w(i, y_i) = 1$.
(2) **LIN**: Linear weighting function: $w(i, y_i) = i$.
(3) **POW**: Power weighting function: $w(i, y_i) = i^p$ (in the experiments we set $p = 2$).
(4) **YES**: Yesterday weighting function that only considers the last value of the time series ("what happened yesterday is what will happen tomorrow"). The YES weighting function is as follows;

$$w(i, y_i) = \begin{cases} 1 & i = t-1 \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

6.3.2. Temporal Methods. The temporal models that we use in these experiments are the five SSM models: the Smoothing model (SMT), the Trend model (TRN), the Periodic model (PRD), the Trend and Periodic model (TRN+PRD), and the Surprise model (SRP). In addition to the temporal models, we also evaluate two temporal model selection methods: BIC and the DML method.

6.4. Prediction Task

For each of the baseline and temporal methods we learn models from the data from December 15, 2010 to April 13, 2011, and predict click behavior for April 14th to April 25th, 2011. We now present three prediction tasks: Query click prediction, URL click prediction task and Query-URL prediction task. The MSE error is averaged over 12 consecutive prediction days (April 14th to April 25th, 2011) to avoid overfitting of a specific day. To compare the results of the different algorithms, we perform a *t-test* on the results comparing against the best performing model. Statistically significant results ($p < 0.05$) of the best models in each row are show in bold.

6.5. Predicting Query Clicks

We now summarize the prediction results for query and URL click frequencies. The query click prediction results are shown in Table II. The table reports prediction errors, where smaller values indicate better prediction accuracy. The best performing model for each query type is shown in bold. For all of the query types, we observe that the DML method performs the best. DML always outperforms the well-known BIC method for model selection as well as all of the SSM models and baselines. This shows that *learning* which model to apply based on the different query features is useful for query-click prediction.

Comparing the temporal SSM models versus the baselines, we observe that, for the General class of queries, the model that smooths surprises performs the best. This result indicates that many queries are noisy and strongly influenced by external events that tend to interfere with model fitting. For the Dynamic class, temporal models that only take into account the trend or learn to decay historical data correctly perform the best. This result aligns with the intuition that queries that represent new events happening during the time of the prediction, thus requiring new results, need the most relevant new information. Thus, data that is too old interferes with prediction. Most of

Table II. The prediction error of different models for predicting the *total clicks* for a query (lower numbers indicate higher performance). The best performing statistical significant results are shown in bold.

	Baselines (Section 6.3.1)				Temporal SSM Models (Section 3.1)					Model Selection (Section 4)	
Query Type	AVG	LIN	POW	YES	SMT	TRN	PRD	TRN+ PRD	SPR	DML	BIC
General	0.17	0.40	0.40	0.18	0.19	0.18	0.17	0.16	0.15	**0.14**	0.19
Dynamic	0.54	0.68	0.68	0.56	0.49	0.49	0.58	0.59	0.60	**0.44**	0.48
Temp Reform	0.56	0.67	0.65	0.78	0.76	0.77	0.59	0.62	0.63	**0.52**	0.73
Alternating	0.30	0.34	0.33	0.33	0.30	0.30	0.44	0.45	0.45	**0.29**	0.33

those queries exhibited a trend in the end of the period. Few disturbances (surprises) were detected, therefore the Surprise model was not useful for this set. For Temporal-reformulations, the best performing temporal models are those that take into account the periodicity of the query. For all query classes, the baseline that performs simple averaging of historical data yields the best results.

Overall, predictions are the most accurate for the General class of queries, indicating that many queries are predictable. The Temporal-reformulation class of queries provides the most difficult challenge for predictive models, showing prediction errors of 0.52 (best prediction model error). Most errors result from queries that are seasonal with a periodicity of more than a year, e.g., holiday related queries (*sears black Friday*) or other annual informational goals as exemplified by the query *2007 irs tax*. As we only had data for a period of five months, such longer-term periodicities were not detected.

We investigated how the models behaved over longer prediction periods h (the prediction horizon). For each such horizon we trained a new model. We present the error of the different models on the General set of queries as a function of the prediction horizon in Figure 16. We observe an interesting phenomenon: almost all methods have a sharp decrease in performance after a prediction horizon of approximately $h = 7$. This demonstrates the complexity of the problem for predicting queries further than a week, perhaps due to the importance of weekly periodicities in queries. However, we did not observe the same phenomenon in the DML method, providing evidence that learning the correct model to apply remains stable throughout all prediction horizons. We observe that the relative performance of the methods remains the same, with one exception – the DML method did not experience any change in bigger prediction horizons.

6.6. Predicting Query-Independent URL Clicks

The predictions for URL clicks aggregated over all queries are given in Table III. We see again that the DML procedure has the best prediction accuracy for Dynamic and Temporal-Reformulation queries. For these classes of queries, models that learn to weight historical behavior and trend models achieve the best performance. For Alternating queries, we observe that predicting clicked URLs benefits from seasonality modeling. For Temporal Reformulation queries, the baseline and DML models show the best prediction performance. Interestingly, the Periodic, Trend+Periodic and Surprise models perform poorly. Similar to what we observed for query prediction, the majority of errors stem from URLs that have periodic content with an annual periodicity lag, which is bigger than the 5 months of data obtained for training the models. The models incorrectly estimate the lag, which is outside of the scope of the data, and therefore the prediction accuracy is poor.

For URL prediction, the best accuracy is achieved for the General query set. The most difficult prediction challenge is found on the Dynamic set. The main reason for

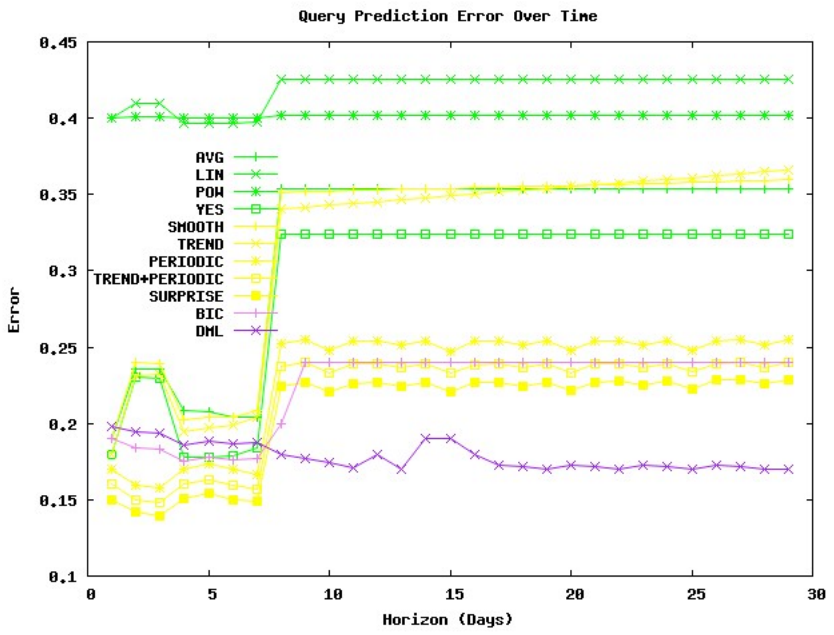


Fig. 16. Query total click prediction error over time on the General queries set (lower values indicate higher performance). Static models are shown in green, Temporal models are shown in yellow, and Temporal model selection methods are shown in purple.

Table III. The prediction error of different models for predicting the query-independent number of clicks for a URL (lower numbers indicate higher performance). Best performing statistical significant results are shown in bold.

	Baselines (Section 6.3.1)				Temporal SSM Models (Section 3.1)					Model Selection (Section 4)	
Query Type	AVG	LIN	POW	YES	SMT	TRN	PRD	TRN+ PRD	SPR	DML	BIC
General	0.02	0.02	0.02	0.02	0.01	0.02	0.01	0.01	0.01	0.02	0.02
Dynamic	0.77	0.75	**0.71**	**0.72**	**0.73**	**0.72**	0.74	0.76	0.76	**0.72**	**0.73**
Temp Reform	0.31	0.30	**0.27**	**0.27**	0.32	0.29	0.78	0.72	0.72	**0.27**	0.28
Alternating	0.49	0.49	0.48	0.47	0.51	0.51	**0.41**	**0.42**	**0.42**	0.48	0.51

the lower prediction performance is that those queries often require new URLs, and are thus harder to predict when no long-term patterns appear.

We present the error of the different models as a function of the prediction horizon in Figure 17. Similar to the results for query prediction, we see that the performance of most models decreases after a prediction horizon of approximately 7 days. Notable exceptions are the DML methods and the LIN and POW methods that remain stable over all prediction horizons.

6.7. Predicting Query-Dependent URL clicks

In the previous section, we compared the forecast models for predicting the overall clicks on a URL aggregated across all queries. We now repeat the analysis but focus on query-dependent clicks instead. Query-URL pair click prediction results are shown in Table IV. We first observe that DML has the best performance for the General, Temporal Reformulation and Alternating queries. The temporal model which smooths across surprises (SPR) is the most accurate for the Dynamic query type.

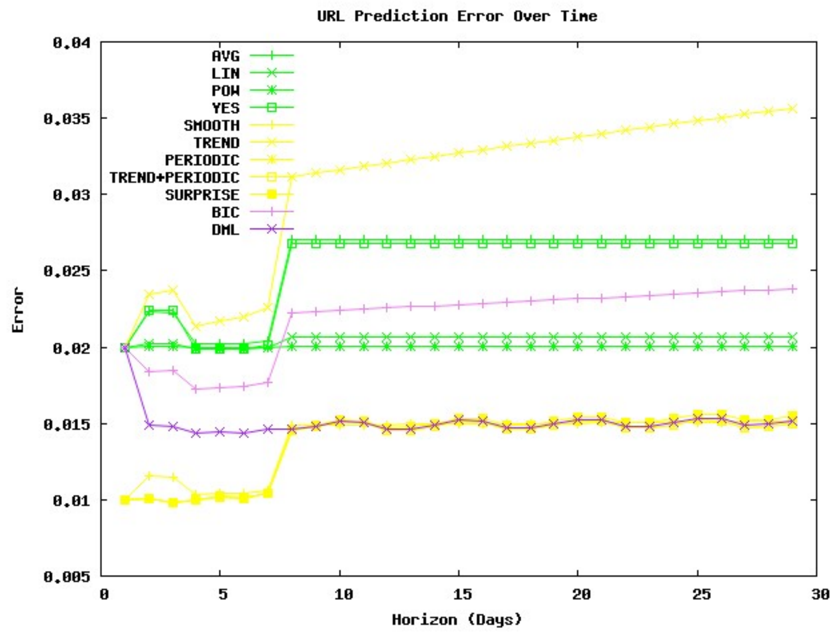


Fig. 17. URL query-independent click prediction error over time on the General queries set (lower numbers indicate higher performance). Static models are shown in green, Temporal models are shown in yellow, and Temporal model selection methods are shown in purple.

Table IV. The prediction error of different models for predicting the query-dependent number of clicks for a URL (lower numbers indicate higher performance). Best performing statistical significant results are shown in bold.

	Baselines (Section 6.3.1)				Temporal SSM Models (Section 3.1)					Model Selection (Section 4)	
Query Type	AVG	LIN	POW	YES	SMT	TRN	PRD	TRN+ PRD	SPR	DML	BIC
General	0.16	0.20	0.20	0.16	0.12	0.14	0.42	0.23	0.23	**0.10**	0.12
Dynamic	0.28	0.40	0.39	0.41	0.29	0.28	0.20	0.20	**0.19**	0.25	0.28
Temp Reform	0.53	0.68	0.67	0.69	0.66	0.69	0.58	0.58	0.59	**0.48**	0.63
Alternating	**0.08**	0.12	0.11	0.13	0.13	0.13	0.17	0.16	0.16	**0.09**	0.12

Across Tables II–IV, there appear to be two groups of SSM models: (1) Smooth and Trend models, and (2) Periodic, Trend+Periodic and Surprise models. Smooth and Trend models show very similar performance to each other. Similarly the Periodic, Trend+Periodic and Surprise models behave in the same way. Sometimes one group performs better than the other, but the groupings are consistent across the three tables and four query types. The main reason for this is that the second group considers periodicities and therefore has a larger set of variables to evaluate. These models have higher expression complexity, but are harder to learn. However, when sufficient data exists and the data is less noisy we see those models perform better.

We present the error of the different models as a function of the prediction horizon in Figure 18. We observe very poor performance of temporal models that incorporate periodicity of any sort (Periodic, Trend+Periodic, Surprise), even for larger horizons. The problem becomes even more evident at a prediction horizon of $h = 7$ and longer. This provides evidence that many of the query-URL predictions are not easily fit by periodic models. However, the different selection models had very high and stable performance.

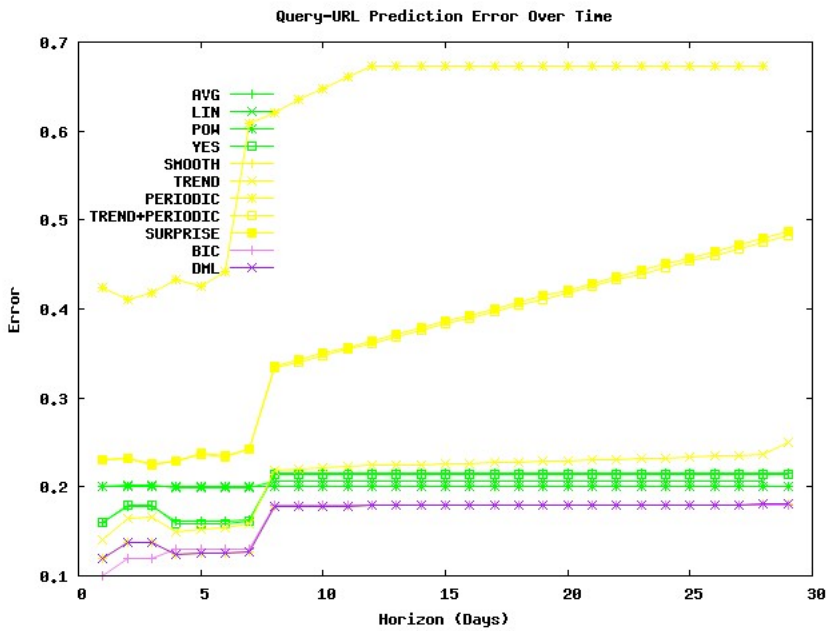


Fig. 18. The query dependent URL click prediction error over time on the General queries set (lower numbers indicate higher performance). Static models are shown in green, Temporal models are shown in yellow, and Temporal model selection methods are shown in purple.

7. EXPERIMENTS FOR RANKING SEARCH RESULTS

We now apply the methods described in Section 3 to ranking of search results. The dataset used in this experiment is described in Section 6.1.We evaluate our techniques in two types of ranking experiments. In the first set of experiments, we use our predictions of future click behavior (derived from the models in 6.4) as the only source of evidence for ranking. In the second set of experiments, we combine our temporal predictions with other query-dependent features (e.g., number of words in query) and query-independent features (e.g., the URL depth) in a learning-to-rank framework.

7.1. Ground-truth Ranking

To evaluate the accuracy of a ranking system, it is common practice to use explicit human relevance judgments (labels) on query-URL pairs. Since user intent can change over time, relevance judgments may also change over time. Obtaining daily judgments on a large set of queries over a long period of time is a difficult challenge. Furthermore, it may be difficult for judges who are unfamiliar with a query to understand the time-varying intentions for the query. To overcome these problems, we use *implicit judgments* drawn from query logs as our gold standard. The gold standard ranking for each day is determined by ordering by the click frequencies of theURL for a query on that day. Similar methods have been used to study personalized and contextual search [Teevan et al. 2005; White et al. 2010]. One challenge with using log data is that users are more likely to click on higher-ranked results than lower-ranked results [Yue et al. 2010]. In order to adjust for this position bias, we normalize by the probability of a click on the given URL at the displayed position for the given query. This method provides an unbiased estimate of the number of times a user would click on a given URL for a given query at a certain time.

Table V. Quality of URL ranking models produced by different forecast models using only predicted clicks, as measured by the Pearson correlation with the ground truth URL ranking. The best performing statistical significant results are shown in bold.

	Baselines (Section 6.3.1)				Temporal SSM Models (Section 3.1)					Model Selection (Section 4)	
Query Type	AVG	LIN	POW	YES	SMT	TRN	PRD	TRN+ PRD	SPR	DML	BIC
General	0.91	0.92	0.93	**0.92**	0.98	**0.98**	0.08	0.98	**0.99**	**0.99**	0.98
Dynamic	0.28	0.35	0.38	**0.36**	0.41	0.41	0.06	0.06	**0.46**	0.42	0.41
Alternating	0.80	0.82	0.84	0.82	**0.89**	**0.89**	0.06	0.06	0.06	**0.89**	0.87
Temp Reform	0.95	0.95	0.95	**0.95**	0.97	0.96	0.24	0.25	0.65	**0.99**	0.95

7.2. Evaluation Metrics

We seek to compare the quality of the rankings generated by our models with those from the gold standard. A common way to compare two ranked lists is to consider the *correlation* between the two rankings. We computed both the Kendall's τ rank correlation and the Pearson product-moment correlation. Rank correlations assess the extent to which one variable increases as the other increases. Pearson correlation assesses the linear relationship between two orderings, and is especially effective for comparing two *regression* variables. The score not only measures the correctness of the ranking, but also takes into account the magnitude of the difference. Results obtained from the Kendall's τ rank correlation and the Pearson product-moment correlation are similar, so we present only the Pearson correlations in the paper.

7.3. Baselines

In experiments where we only use temporal features as evidence for ranking, we compare the ranking performance of the temporal-modeling methods (Section 3) versus the performance of the static-modeling baselines (Section 6.3.1).

In the experiments where temporal features are added to the feature set of a *base* ranker, we compare the performance of the ranking algorithm using three different sets of features: only base features; the base features plus static modeling features; and the base features plus temporal modeling features. The feature set of the *base* ranker consists of approximately 200 typical information retrieval features such as several variants of BM25 [Robertson et al. 2004] to measure the similarity of a query to document body, title, and anchor texts. Additionally, we used other query dependent features, such as the matched document frequency of a term, bigram frequency, number of words in query, etc., and a set of query-independent features, such as the URL depth and the number of words in the document's body, title.

7.4. Ranking Task

For each query-URL pair we compute the predicted behavior and use that prediction to rank the URLs. We first divide the data into training and test sets. Behaviors from time $t_i, \ldots, t_{i+n}$ are used to estimate the model parameters and the resulting models are used to predict the user behavior on the following day t_{i+n+1}. For each query, we compare the predicted ranking to the gold standard ranking on day t_{i+n+1}. We investigate different prediction windows (of 1–10 days), and perform prediction on different days (training was performed on the dates 12-15-2010 until 4-15-2011 and the testing was done on on the subsequent 10 days). For each query type (described in Section 6.2), we calculate the mean of the Pearson correlation (ρ)over the queries in our test set.

7.5. Temporal Features as Only Evidence for Ranking

We begin by looking at how well the predictions for URL click behavior perform compared with several baseline predictions. The results can be found in Table V, that presents the Pearson scores for each of our four query types.

For every type of query, the proposed temporal modeling approach outperforms all of the baselines, showing that learning the appropriate weighting for historical data is preferable to selecting a single weighting for all query and/or URLs. For most classes of queries, the Pearson scores are quite high, suggesting that the predicted ranking is very close to the ideal ranking. We observe that ranking based only on predicted user clicks is consistently better than the baselines for all types of queries. The correlations are noticeably lower for Dynamic queries, although temporal models still significantly outperform baseline models. Dynamic queries are ones that have been labeled as requiring fresh information. For these queries, there is limited behavioral data available related to this fresh information need. Although we observe that models that weight more recent data more heavily (like the trend and power weight models) tend to perform well on this query class, the overall performance is still significantly lower than the accuracy in other query classes. A closer look at the data reveals that most of those queries are celebrity and sports-related queries that have sudden spikes. These sudden spikes are hard to predict, as they are event driven. However, after such a spike starts, the prediction of its peak and its decline are more easily predicted, as those spikes usually persist for 1-2 days. Most errors of those models occur in queries where the test date is drawn from the beginning of a spike.

For every query class, two of the temporal approaches that we explored, Smoothing (SMT) and Trend (TRN), show significant improvements over all of the baselines. Recall that ranking with Smoothing creates a short-range prediction assuming a stable mean, and Trend adds to this model the notion of a trend. The more complex temporal models, namely Periodic (PRD), and Trend + Periodic (TRN+PRD), does not generally perform as well. The Periodic model exhibits poor performance for most query sets. The Periodic model requires the estimation of many parameters – the size of the parameters is linear by the size of the periodicity. Estimating such a large number of parameters likely requires more data than we had available; we believe that this is the main reason for its lower performance. Furthermore, some of the periodicities were yearly, and therefore not identified well. The poor performance of the model on non-periodic queries stems from the miscalculation of the periodicity of those queries, which usually results in a noisy prediction.

We reported results for the top four URLs. We also examined performance of the models over different numbers of top ranked URLs. We have experimented on rankings of up to 12 URLs. We did not observe any statistically significant change in the results, and hence do not present them here.

7.6. Temporal Features as Input to a Ranking Algorithm

In this section, we investigate the effectiveness of predicted temporal features when added to a baseline learning-to-rank retrieval model. We employ a supervised learning technique to learn the ranking function that best predicts the ground-truth ranking of URLs for a given query, using a variety of query, and URL click features. In these experiments, we use boosted regression trees as our ranking function [Burges 2010].

We divide the data into training queries Q^{train} (80% of the total number of queries Q) and test queries Q^{test} (20% of Q). The features used in Q^{train} are obtained using only data from the times $t_1, \ldots, t_n$. For every query in Q^{test}, we use the data from the times $t_1, \ldots, t_n$ to predict the user behavior for the following day, and use this prediction as a feature fed into the learner for testing. Using separate train and test queries allows

Table VI. Quality of URL ranking models produced by different feature sets for learning-to-rank as measured by the Pearson correlation with the ground truth URL ranking. The best performing statistical significant results are shown in bold.

	Baselines (Section 6.3.1)				Temporal SSM Models (Section 3.1)					Model Selection (Section 4)	
Query Type	AVG	LIN	POW	YES	SMT	TRN	PRD	TRN+ PRD	SPR	DML	BIC
General	0.47	0.97	0.98	0.98	0.91	**0.99**	**0.99**	0.60	0.97	**0.99**	0.72
Dynamic	-0.08	0.30	0.30	0.39	0.59	0.46	0.01	0.30	**0.62**	0.61	0.35
Alternating	0.23	0.64	0.90	0.74	0.78	0.87	0.90	0.23	0.65	**0.98**	0.84
Temp Reform	0.19	0.73	0.97	0.96	**0.99**	**0.99**	0.98	0.98	**0.99**	**0.99**	0.96

us to generalize to queries that have not previously been seen. Using the results of the ranker, we again calculate the average Pearson correlation for every query category.

We performed experiments comparing performance of a ranker using a variety of query-dependent and query-independent features in addition to temporal features based on historical patterns of user behavior.

The Pearson correlation results of these experiments can be found in Table VI. In nearly all cases, the inclusion of features based on user behavior in the ranker improved performance, regardless of how the behavior was modeled. The level of improvement observed was often quite significant, confirming that user behavior is a very important aspect of Web search result ranking [Agichtein et al. 2006]. The rankers that use temporal modeling consistently performed the best across all query classes.

7.7. Query Effects

We have seen that the different models perform differently on different classes of queries. In this section, we analyze characteristics of the queries that benefit from temporal behavioral signals when used for ranking. We perform an analysis of the results using the same 973 characteristics discussed in Section 4.2.3, and identify the characteristics that differentiate between the ranking results with and without the use of temporal models. For these experiments, we assess statistical significance using paired t-tests comparing the best performing temporal models to all other models.

Click Entropy. We begin by looking at whether ambiguous queries are more or less likely to benefit from temporal modeling of user behavior. We use *query click entropy* as a measure of the variation in what users click following a query, as defined in Eq. 12. We calculate the click entropy over the learning period for each of our queries.

We observe that the ranking of queries with higher click entropy (over 0.37) improves when using a model with temporal signals compared to using models with only baseline user behavior features. Queries with many different distinct search results clicked at different times benefit from temporal modeling. Because the click entropy was calculated over the entire study period, and not just a single slice of time, it is likely that high click entropy often indicates the search engine users' need for different information over time. For example, the query *lottery* has high click entropy, as the different lotteries results are reported on different days, depending on the location. Therefore, every day, users tend to click on the lottery site of the appropriate location. As the result are reported periodically in each location, the temporal models manage to predict the correct ranking of the results.

Click Predictability. We measure the predictability of a query by calculating the average error in predicting its search volume (query clicks) using either a baseline model (e.g., Averaging) or a temporal model (e.g., Smoothing) over the seven days before the prediction. An analysis of the data reveals that temporal ranking performs poorly compared with the baseline models when the query clicks predictability is low. Regardless of the model used for predicting query clicks, the error is high (0.5-0.58) when the performance of the temporal model is worse than the baseline model. The error is low (0.27-0.36) when the performance of the temporal model is better than the baseline model. This indicates that if the query search behavior on its own is predictable, predicting the right ranking for it over time might be a simpler task.

Similarly, we found a high correlation between the predictability of the query-URL search click volume and the performance of the ranker with time-series modeled features. We measure this predictability by measuring the average error of prediction of every URL for the query. The error is measured by averaging over the temporal model prediction error for 7 days before the prediction over the URLs. Queries for which the time-sensitive ranker performed well indicate that query-URL features have high predictability. The error rate is lower (0.20) for instances where temporal modeling performs better than the baseline, and is higher (0.27) when the temporal modeling had worse performance. Wound those differences to be statistically significant. This query-URL number of clicks feature is important for ranking, as it is eventually the goal function being optimized. Queries and query-URLs whose search volume changes in an unpredictable manner pose harder challenges for temporal ranking. We believe that difficulties arise during learning because extracting meaningful statistics is more challenging in those scenarios. In these cases, simple average modeling of the query-URL volume yields better results.

Spiking Trends. It may seem intuitive that the behavior of queries associated with numerous spikes over time is hard to predict. We analyze the correlation between ranking performance and the number of spikes in the query, query-URL or URL time series. We found no significant correlation. We believe that this is because some spikes can be predictable with high precision, such as in the case of the periodic behavior.

Query Shapes

In this section we summarize how the different query shapes indicate which temporal model to apply. As discussed in Section 3, query shape often indicates whether a temporal model or a baseline should be applied in ranking. We clustered the different click behaviors in our dataset based on their shapes, using Expectation–maximization clustering algorithm. Figure 19 shows the four query and query-URL shapes where the temporal model yielded better rankings than baselines rankers. . These four shapes have upward or downward trends with different slopes. Those can be modeled well by the Smoothing and Trend techniques with a correctly learned slope. This clustering result is consistent with the results in Sections 7.5 and 7.6, where these two modeling techniques tended to yield the highest performance. We did not find distinctive query clusters for the cases when the baseline ranker outperformed the ranker using temporal modeling.

8. EXPERIMENTS FOR RANKING QUERY AUTO-COMPLETE CANDIDATES

In this section, we apply the techniques presented in Section 3 to predict query clicks for QAS ranking. The ground-truth QAS ranking for a prefix $\mathcal{P}$ is the list of all queries that start with $\mathcal{P}$ (or are somehow filtered for the prefix) ordered according to their *true* popularity. At any given time t, the true popularity values are set according to the observed query frequency values at that time. Obviously, this information is not available to QAS ranking models at runtime, and we have to rely on historical data at time $t-1$ and before to *predict* this unobserved ground-truth ($\mathcal{G}$) and rank candidates accordingly.

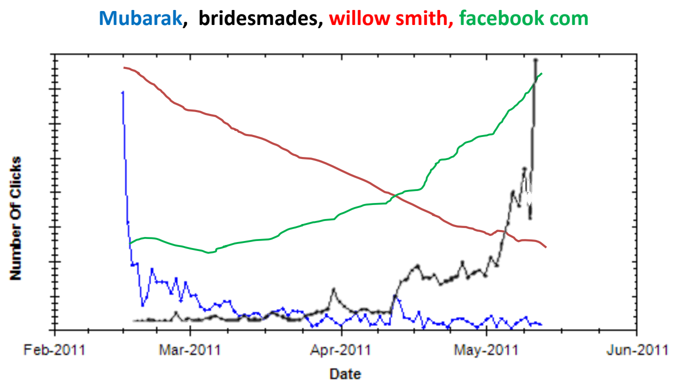


Fig. 19. Dominant query shapes for queries where the proposed temporal model yields better rankings than the baseline rankers.

8.1. Data & Ground-truth Ranking

We use the dataset described in Section 6.1 and generate an auto-completion trie based on query frequency predictions for each of the days in the testing period. For each forecast method at time t, we rank the top-5 ground-truth candidates based on the data available at time $t-1$ and before. In an oracle list of QAS suggestions, candidates are sorted in descending order of their *ground-truth* popularity. We consider two types of ground-truth: (1) QAS Clicks Ground-truth; and (2) Total Clicks Ground-truth.

QAS Clicks Ground-truth. We obtained click data for each QAS suggestion for the period of May 6th until May 12th, 2011 (a total of 4,687,814 prefixes and queries). The raw data consists of a date, a prefix typed, and the suggestion clicked for this prefix on that date. Therefore, our gold standard for a certain time and prefix is the ranking induced by the number of times the query suggestion was clicked for this prefix at this time.

Total Clicks Ground-truth. The coverage of QAS clicks is limited to those queries that were available in the auto-completion trie and were selected from the auto-completion list by the users. However, a significant fraction of queries submitted by users are not available in the auto-completion trie. In addition, users sometimes issue the query directly from the search box even when it exists in the auto-completion list. Therefore, we generate another *ground-truth* dataset in which we consider the total clicks on queries regardless of whether they were chosen from auto-completion lists or not. We obtained click data for each query for the period of May 6th until May 12th, 2011 (a total of 3,384,954 queries). We computed the probability of a click on a query by the volume of clicks on this query on that date. The raw data consists of a date, a query issued and how many times it was clicked at that date. Therefore, our gold standard for a certain time and prefix is the ranking induced by the number of times queries with this prefix were issued and had a click on that date. That is, for each prefix $\mathcal{P}$ at time t, we match all the queries that start with $\mathcal{P}$ and rank them according to their true total frequency at time t to generate the ground-truth ranking.

8.2. Evaluation Metrics

We compare the quality of the QAS generated by our models with those from the gold standard. As in our previous experiments, we use Pearson product-moment correlation to compare the orderings of the QAS.

Table VII. Quality of auto-completion ranking models produced by different learning-to-rank feature sets, as measured by the Pearson correlation with QAS (top) and total clicks (bottom) Ground-truth ranking. The best performing statistical significant results are shown in bold.

	Baselines (Section 6.3.1)				Temporal SSM Models (Section 3.1)					Model Selection (Section 4)	
Ground-Truth	AVG	LIN	POW	YES	SMT	TRN	PRD	TRN+ PRD	SPR	DML	BIC
QAS	0.88	0.88	0.88	0.88	0.87	0.88	0.83	0.83	0.89	**0.91**	0.88
Total Click	0.93	0.94	0.94	0.93	0.93	0.93	0.85	0.85	0.95	**0.97**	0.94

8.3. QAS Ranking Results

In Table VII we present the results for ranking query suggestions based on both ground-truths. We did not observe any significant difference in the results using temporal or static user behavior models. However, we see that the DML model, which is trained to choose among the more specific models, achieves the highest performance over all queries, particularly when total clicks are considered as ground-truth. The results were found to be statistically significant. Periodic models have shown low performance in both sets, since many of the queries are not periodic and therefore the periodic modeling hurts the performance. The results indicate once again that correct modeling of historical user behavior signals is beneficial.

8.4. Query Effects

We examined the data in more detail to investigate which queries benefit from temporal modeling. We categorized the queries into two groups: news and periodic queries.

News or event-related queries. Many queries that have sudden spikes benefit from Trend or Smooth modeling. For example, the query *Navy Seals* started trending on May 2nd, 2011 during the Osama Bin-Laden mission. In the period following May 2nd, we observed that static modeling of this query for the prefix *navy* had lower performance than the Trend and Smooth modeling. This is because this event had not been observed in the past, so past data was not indicative for the current ranking. An interesting subclass of queries are celebrity queries. Queries about celebrities (such as *Chris Hemsworth* and *Marie Osmond*) are usually event related, and benefit from the temporal modeling. For example, at the time of the prediction, a new film starring Chris Hemsworth was released and Marie Osmond remarried her first husband. Both events caused spikes in the query volume which static models have difficulty modeling. Although the YES model performed well on this Hemsworth query predictions, it still had lower performance than that of the Trend model that correctly modeled the interest over time.

Periodic queries. Periodic queries also benefited from the temporal periodic modeling. For example, the query *first friday* (for the prefix *first*) was very poorly modeled by the static behavior models, but the Periodic models modeled it with very high performance. The periodic models had a Pearson correlation of 0.99 vs. -0.34 for the static models over the periodic queries.

9. CONCLUSIONS

We developed methods for modeling the dynamics of the query and click behaviors seen in a large population of Web searchers. We modeled temporal characteristics that are often observed in query and URL click behavior, including trend, periodicity, and surprise disruptions. We presented several different temporal representations and learning procedures and showed how we can construct models that predict future behaviors from historical data.

In almost all cases, we found that temporal models performed better than the baseline models that use the same weighting of historical data for all queries for predicting clicks, ranking search results and ranking query suggestions. We also found that different temporal models are appropriate for different types of queries. Query click behavior tends to be rife with unmodeled disturbances (surprises). Thus, extending models with the ability to identify and smooth out such disturbances can enhance predictive power. For specific types of queries, such as Dynamic and Alternating queries, trend models are more appropriate. URL click behavior tends to exhibit somewhat different temporal behavior showing fewer disturbances (as we did not see much gain with using the surprise temporal model). Thus, models that understand trend and smooth historical data using learned models tend to perform better for these predictions. For Alternating queries, periodic modeling tends to work better. The dynamics seen in query-URL click behavior over time provides interesting insight into changes in users' intentions over time. For General and Alternating queries smoothing and trend models are the best temporal models, whereas for Dynamic and Temporal Reformulations modeling periodicities improves performance. We observed that, in general, the application of smoothing or incorporating trend is the best methodology.

We introduced the Dynamics Model Learner (DML) algorithm that learns the correct model to apply for queries, URL, or query-URL pairs without a priori categorization of queries into groups. We compared the predictive performance of this method with static baselines, temporal models, and a standard model selection algorithm (BIC), and found that it has superior performance for all groups of queries. We believe this result paves the way for incorporating multiple types of models for better time-aware search procedures.

The kinds of time-aware modeling of user behavior that we introduced in this paper can be incorporated in many search-related applications. In this paper, we examined basic click predictions, as well as two end-to-end applications – ranking of results and query suggestions. Query click prediction can be used to improve query auto-completion to present the most appropriate suggestions at the time the query is issued. We studied this application and provided evidence that, in most cases, the temporal models are more successful in query suggestions. Query-URL prediction can induce better ranking that is more aware of the user query-URL temporal intent. We evaluated this claim in a variety of experiments providing support that temporal modeling improves ranking for many classes of queries.

There are several directions for future work. We believe that further experiments examining how long it takes for temporal aspects of a new query or URL to sufficiently permeate the search stream so that it can be aided by temporal prediction are needed to enhance many applications. A more in-depth study with human subjects can also reveal whether users perceive a qualitative difference in the user experience when temporal techniques to improve ranking, QAS, etc. Additionally, studies of other model selection techniques, such as AIC or multimodel approaches, e.g., using Bayesian model averaging, might also shed light on the pragmatic use of these techniques.

Additional applications of these techniques are also possible and include the use of URL click prediction to improve re-crawling strategies, by focusing crawling efforts on URLs that are likely to be clicked. In general, we believe similar techniques might be used to optimize indexing, storage, and retrieval strategies to improve response time. Furthermore, the models presented can also be used at different time granularities and applied on different types of objects (Query, URL and Query-URL), either aggregated for all users or applied on specific user behavior, thus creating time-aware personalized retrieval, where the temporal intent is modified to accommodate individual searchers.

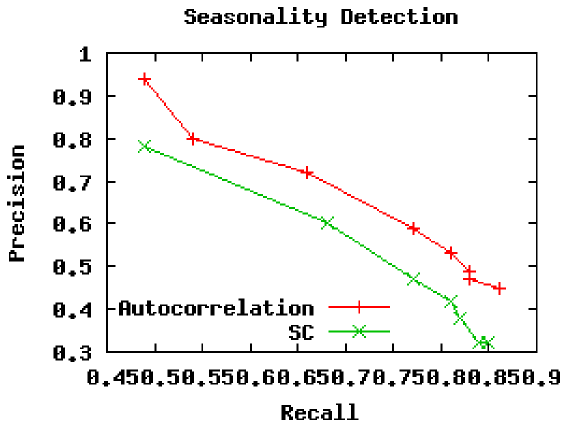


Fig. 20. Comparison of SC method with Autocorrelation method for seasonality detection by precision (y axis) and recall (x axis).

ACKNOWLEDGMENTS

We thank Dan Liebling for help with obtaining the user behavior data, and Chris Meek and Kuansan Wang for fruitful discussions.

A. APPENDIX

This section summarizes the details of training surprise and periodicity detection models.

A.1. Learning To Detect Periodicity

Detecting the periodicity size of a time series is crucial for periodic models. For this process we use a signal processing method called *autocorrelation* that provides a signal's correlation value with itself [Dunn 2005]. This method can be used to detect repeating patterns with noise, and is defined as

$$Autocorrelation(f,h) = \int_{-\infty}^{\infty} f(t+h)f(t)\,\mathrm{dt}, \tag{15}$$

where h is the shift of the series, i.e., the periodicity. Several h values are experimented, and the highest value of the autocorrelation for those values is used. A query is classified as *periodic* based on a certain threshold ω:

$$Autocorrelation(f,h) > \omega, \tag{16}$$

Specifically in the Web domain, we found it beneficial to limit the possible h values to common Web-periodic intervals, such as weekly, monthly, and yearly.

We performed experiments to evaluate the periodicity detection algorithms. This component is crucial for initializing the SSM seasonal models. To capture long as well as short periodicity, search logs for query-click data for the period 2006–2011 were obtained, and a daily time series was generated for every query. We used the seasonality dataset [Shokouhi 2011] that contained 259 queries, each annotated manually as seasonal or not. We compare our method for periodicity detection against the method described in Shokouhi [2011] (we refer to it as SC), which was applied to perform seasonal-trend decomposition on the time series, comparing the seasonal component to that of the time series before the decomposition.

To evaluate our periodicity model, we evaluate performance for different autocorrelation thresholds, ω. Figure 20 shows the precision-recall results for our autocorrelation model (Autocorrelation) compared to the baseline model (SC). The Autocorrelation method proposed in this work reaches the same maximum recall as the state-of-the-art SC autocorrelation method (around 0.85), and outperforms it in precision for every recall level by up to 15 percent. We also performed experiments for applying autocorrelation without any lag limiting. The result yields a recall of about 0.25–0.27 with precision of 0.5–0.6. We conclude that the lag limitation for regular Web periodicity ($h = 7, 28, \ldots, 31, 360 \ldots 365$) is important.

A.2. Learning To Detect Surprises

Queries can be modeled using one of the SSM models introduced previously but, as we discussed earlier, sometimes areas of the time series exhibit *surprising* behavior that is not well-fit by the model. We conjecture that, when a temporal model encounters a surprise in the data, the model's residual error stops behaving linearly. Intuitively, in the beginning of a spike the model *"under-predicts"* the data since the previous data did not include any indication of a spike. After the spike, the model tends to *over-predict* the data, as it still considers the spike's data. We introduce *surprises* as significant changes in the residual during the period of an event. Let $r = r_1, \ldots, r_n$ be the residuals of the temporal model for the times $t_1, \ldots, t_n$, where $r_t = o(t) - \widehat{o}(t)$, and $\widehat{o}(t)$ is the prediction of the model for time t. Let $t'_1, \ldots, t'_m$ be surprise candidates time points, such that $r_{t'_1-1} \cdot r_{t'_1} < 0$ for each $t' \in \{t'_1, \ldots, t'_m\}$, i.e., locations in the time series where the residual changes signs. Let r_{t_1} and r_{t_2} be two neighbouring sign-change points, such that $r_{t_1-1} \cdot r_{t_1} < 0, r_{t_2+1} \cdot r_{t_2} < 0, r_t \cdot r_{t_1} > 0, t_1 \leq t \leq t_2$. We define an impact of an event as

$$Impact(t_1, t_2) = MSE(o, t_1, t_2, m) = \frac{\sum_{t=t_1}^{t_2} r_t^2}{t_2 - t_1}.$$

Intuitively, only unexpected dynamics that have long impact on the model should be considered as surprises. We propose a greedy procedure that adds the surprise locations starting from highest to lowest impact to the model and measures the improvement of the model (using BIC criterion). When the model stops improving, we output the surprises. The surprise detection algorithm is given in Figure 21.

We performed experiments to evaluate our surprise detection algorithm. We obtained a set of 24 queries judged by humans as news-related queries. Judges were also asked to provide a label (event or not) for every week in the series (a total of 313 data points for query trends between 2006-2011). A total of 61 events were detected. For every query we trained the surprise detection model and a baseline method, and compare their results. The baseline model is a method that detects peaks as events. This is a reasonable baseline, as many of the events can be seen as peaks in the query stream.

Table VIII shows results of our surprise detection algorithm compared to the baseline peak detection algorithm. We see the surprise detector has high precision and low recall. On the other hand, the peak detector identifies all surprises but with very low precision. As most peaks in queries are usually noise, and not real events, we prefer the Surprise Detector over the peak detection method. For example, the query *credit rating* has some seasonal peaks after S&P declarations, which should not be considered a surprise. However, the peak detector identifies them as such, while the surprise detector does not recognize it as an event. On Aug 8th, when U.S. credit rating was downgraded, the query volume exhibited an unusual peak, which was identified as a surprise by the surprise detector.

```
Procedure SURPRISE DETECTOR($o(t_1), \ldots, o(t_n)$,m)
  $BIC_{i-1} \leftarrow \infty$
  $BIC_i \leftarrow \infty$
  $EventCandidates = \{t'_1, \ldots, t'_m | r_{t'_{i-1}} \cdot r_{t'_i} < 0\}$
  $Events = \{\}$
  Do
    $curEvent \leftarrow \arg\max_{t'_i} Impact(t'_i)$
    $EventCandidates \leftarrow EventCandidates / \{curEvent\}$
    $m_i \leftarrow$ Build model $m$ with events $Events$
      using training data $o(t_1), \ldots, o(t_n)$
    $BIC_i \leftarrow BIC(m_i)$
    If $BIC_i > BIC_{i-1}$
      $Events \leftarrow Events \cup \{curEvent\}$
      $BIC_{i-1} \leftarrow BIC_i$
    Else Return $Events$
  While $EventCandidates \neq \{\}$
  Return $Events$
```

Fig. 21. Detecting surprises in time series.

Table VIII. Surprises Detector Results. Recall and precision of the different surprises detectors. Statistically significant results (as measured by a t-test) are shown in bold.

	Surprises Detector	Peak Detector
Precision	**96.42%**	46.66%
Recall	59.92%	**100%**

REFERENCES

ADAR, E., WELD, D. S., BERSHAD, B. N., AND GRIBBLE, S. D. 2007. Why we search: visualizing and predicting user behavior. In *Proceedings of International World Wide Web Conference (WWW)*.

AGICHTEIN, E., BRILL, E., AND DUMAIS, S. T. 2006. Improving web search ranking by incorporating user behavior information. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*.

BAR-YOSSEF, Z. AND KRAUS, N. 2011. Context-sensitive query auto-completion. In *Proceedings of International World Wide Web Conference (WWW)*. Hyderabad, India, 107–116.

BEITZEL, S. M., JENSEN, E. C., CHOWDHURY, A., GROSSMAN, D., AND FRIEDER, O. 2004. Hourly analysis of a very large topically categorized web query log. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*.

BENNETT, P. N., SVORE, K., AND DUMAIS, S. T. 2010. Classification-enhanced ranking. In *Proceedings of International World Wide Web Conference (WWW)*.

BICKEL, S., HAIDER, P., AND SCHEFFER, T. 2005. Learning to complete sentences. In *Proceedings of the European Conference on Machine Learning (ECML)*. 497–504.

BLEI, D. M., NG, A. Y., AND JORDAN, M. I. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research (JMLR) 3*, 993–1022.

BOGERT, B., HEALY, M., AND TUKEY, J. 1967. Cepstrum pitch determination. *Journal of the Acoustical Society of America 41*, 293–309.

BURGES, C. J. C. 2010. From RankNet to LambdaRank to LambdaMART: An Overview. Tech. rep.

CHAUDHURI, S. AND KAUSHIK, R. 2009. Extending autocompletion to tolerate errors. In *Proceedings of the International Conference on Management of Data (SIGMOD)*. 707–718.

CHIEN, S. AND IMMORLICA, N. 2005. Semantic similarity between search engine queries using temporal correlation. In *Proceedings of International World Wide Web Conference (WWW)*.

CHILDERS, D., SKINNER, D., AND KEMERAIT, R. 1977. The cepstrum: A guide to processing. *IEEE 65*, 1428–1443.

DAI, N., SHOKOUHI, M., AND DAVISON, B. D. 2011. Learning to rank for freshness and relevance. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*.

DAKKA, W., GRAVANO, L., AND IPEIROTIS, P. G. 2008. Answering general time sensitive queries. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*.

DARRAGH, J. J., WITTEN, I. H., AND JAMES, M. L. 1990. The reactive keyboard: A predictive typing aid. *Computer 23*, 41–49.

DIAZ, F. 2009. Integration of news content into web results. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*.

DONG, A., CHANG, Y., ZHENG, Z., MISHNE, G., BAI, J., ZHANG, R., BUCHNER, K., LIAO, C., AND DIAZ, F. 2010a. Towards recency ranking in web search. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*.

DONG, A., ZHANG, R., KOLARI, P., BAI, J., DIAZ, F., CHANG, Y., ZHENG, Z., AND ZHA, H. 2010b. Time is of the essence: improving recency ranking using twitter data. In *Proceedings of International World Wide Web Conference (WWW)*.

DUNN, P. F. 2005. *Measurement and Data Analysis for Engineering and Science*. McGraw-Hill.

DURBIN, J. AND KOOPMAN, S. 2008. *Time Series Analysis by State Space Methods*. Oxford University Press.

EFRON, M. 2010. Linear time series models for term weighting in information retrieval. *Journal of the American Society for Information Science and Technology (JASIST) 6,* 7.

EFRON, M. AND GOLOVCHINKSY, G. 2011. Estimation methods for ranking recent information. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*.

ELSAS, J. L. AND DUMAIS, S. T. 2010. Leveraging temporal dynamics of document content in relevance ranking. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*.

FAN, J., WU, H., LI, G., AND ZHOU, L. 2010. Suggesting topic-based query terms as you type. In *Proceedings of the International Asia-Pacific Web Conference (APWeb*. Washington, DC, 61–67.

GINSBERG, J., MOHEBBI, M., PATEL, R., BRAMMER, L., SMOLINSKI, M., AND BRILLIANT, L. 2009. Detecting influenza epidemics using search engine query data. *Nature 457,* 7232, 1012–4.

GRABSKI, K. AND SCHEFFER, T. 2004. Sentence completion. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*. Sheffield, United Kingdom, 433–439.

HOLT, C. C. 2004. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting 20,* 1, 5–10.

HYNDMAN, R., A.KOEHLER, J.ORD, AND R.SNYDER. 2008. *Forecasting with Exponential Smoothing (The State Space Approach)*. Springer.

JI, S., LI, G., LI, C., AND FENG, J. 2009. Efficient interactive fuzzy keyword search. In *Proceedings of International World Wide Web Conference (WWW)*. Madrid, Spain, 371–380.

JONES, R. AND DIAZ, F. 2007. Temporal profiles of queries. *ACM Transactions on Information Systems 25*.

KLEINBERG, J. 2002. Bursty and hierarchical structure in streams. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*.

KLEINBERG, J. 2006. Temporal dynamics of on-line information systems. *Data Stream Management: Processing High-Speed Data Streams*.

KÖNIG, A. C., GAMON, M., AND WU, Q. 2009. Click-through prediction for news queries. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*.

KOREN, Y. 2009. Collaborative filtering with temporal dynamics. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*.

KULKARNI, A., TEEVAN, J., SVORE, K. M., AND DUMAIS, S. T. 2011. Understanding temporal query dynamics.

LAU, T. AND HORVITZ, E. 1998. Patterns of search: Analyzing and modeling web query refinement. In *Proceedings of the Seventh International Conference on User Modeling*. Springer Wien.

LI, X. AND CROFT, W. B. 2003. Time-based language models. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*.

METZLER, D., JONES, R., PENG, F., AND ZHANG, R. 2009. Improving search relevance for implicitly temporal queries. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*.

NANDI, A. AND JAGADISH, H. V. 2007. Effective phrase prediction. In *Proceedings of the Conference on Very Large Databases (VLDB)*. Vienna, Austria, 219–230.

RADINSKY, K., AGICHTEIN, E., GABRILOVICH, E., AND MARKOVITCH, S. 2011. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of International World Wide Web Conference (WWW)*.

RADINSKY, K., DAVIDOVICH, S., AND MARKOVITCH, S. 2008. Predicting the news of tomorrow using patterns in web search queries. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI)*.

RADINSKY, K., SVORE, K., DUMAIS, S., TEEVAN, J., BOCHAROV, A., AND HORVITZ, E. 2012. Modeling and predicting behavioral dynamics on the web. In *Proceedings of International World Wide Web Conference (WWW)*.

ROBERTSON, S., ZARAGOZA, H., AND TAYLOR, M. 2004. Simple bm25 extension to multiple weighted fields. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*.

SCHWARZ, G. E. 1978. Estimating the dimension of a model. *Annals of Statistics 2,* 6, 461–464.

SHIMSHONI, Y., EFRON, N., AND MATIAS, Y. 2009. On the predictability of search trends. In *Technical Report*.

SHOKOUHI, M. 2011. Detecting seasonal queries by time-series analysis. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*.

SHOKOUHI, M. AND RADINSKY, K. 2012. Time-sensitive query auto-completion. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*.

SNYMAN, J. A. 2005. *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*. Springer.

TEEVAN, J., DUMAIS, S. T., AND HORVITZ, E. 2005. Personalizing search via automated analysis of interests and activities. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*.

VLACHOS, M., MEEK, C., VAGENA, Z., AND GUNOPULOS, D. 2004. Identifying similarities, periodicities and bursts for online search queries.

WANG, P., BERRY, M. W., AND YANG, Y. 2003. Mining longitudinal web queries: trends and patterns. *Journal of the American Society for Information Science and Technology (JASIST) 54*, 743–758.

WANG, X., ZHAI, C., HU, X., AND SPROAT, R. 2007. Mining correlated bursty topic patterns from coordinated text streams. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*.

WHITE, R. W., BENNETT, P. N., AND DUMAIS, S. T. 2010. Predicting short-term interests using activity-based search context. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*.

WHITE, R. W. AND MARCHIONINI, G. 2007. Examining the effectiveness of real-time query expansion. *Information Processing and Management 43*, 685–704.

YUE, Y., PATEL, R., AND ROEHRIG, H. 2010. Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data. In *Proceedings of International World Wide Web Conference (Proceedings of International World Wide Web Conference (WWW)*.