Modeling and Predicting Behavioral Dynamics on the Web

Kira Radinsky[‡*], Krysta Svore[†], Susan Dumais[†],
Jaime Teevan[†], Alex Bocharov[†], Eric Horvitz[†]

[‡] CS Department, Technion—Israel Institute of Technology, 32000 Haifa, Israel
[†] Microsoft Research, Redmond, WA 98052
[‡] kirar@cs.technion.ac.il [†] {ksvore|sdumais|teevan|alexeib|horvitz}@microsoft.com

ABSTRACT

User behavior on the Web changes over time. For example, the queries that people issue to search engines, and the underlying informational goals behind the queries vary over time. In this paper, we examine how to model and predict this temporal user behavior. We develop a temporal modeling framework adapted from physics and signal processing that can be used to predict time-varying user behavior using smoothing and trends. We also explore other dynamics of Web behaviors, such as the detection of periodicities and surprises. We develop a learning procedure that can be used to construct models of users' activities based on features of current and historical behaviors. The results of experiments indicate that by using our framework to predict user behavior, we can achieve significant improvements in prediction compared to baseline models that weight historical evidence the same for all queries. We also develop a novel learning algorithm that explicitly learns when to apply a given prediction model among a set of such models. Our improved temporal modeling of user behavior can be used to enhance query suggestions, crawling policies, and result ranking.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

Behavioral Analysis, Predictive Behavioral Models

1. INTRODUCTION

Numerous aspects of the Web change over time, from such salient changes as the addition and evolution of content, to more subtle but important dynamics. We explore the temporal dynamics of user behavior, investigating how we can model and predict changes in the queries that people issue, the informational goals behind the queries, and the search results they click on during Web search sessions. In information retrieval, models of user behavior have been employed in the past as static sources of evidence for enhancing access; user behavior has been aggregated over time and also used identically for all types of queries. We explore opportunities for learning about how the behavior of users changes over time and then using predictive models to enhance retrieval. As an example, for a population of users, the frequency that a query is issued and the number of times that a search result is clicked on for that query can change over time. We model such dynamics in the behavior as a time series, focusing on queries, URLs, and query-URL pairs as the behaviors.

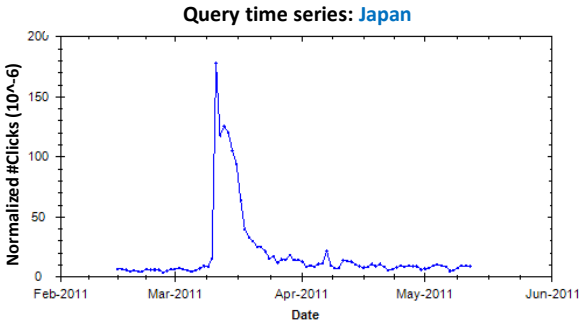

Figure 1: A time series (2/14–5/25, 2011) for the query *japan* (normalized by overall #clicks logged on each day).

In Figure 1, we show the query-click behavior of the query *japan* over time, i.e., number of clicks on any URL shown for the query. We can see a dramatic change in behaviors 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. Both the frequency with which a URL is returned for a query and the frequency that specific URLs are clicked on can change over time. We explore the changing behavior of a population of searchers via changes in the frequency of *query-URL pairs* over time. For the *japan* query example, the frequencies of access of some URLs change over time because of changes in query frequency, while others do not change (see Figure 2). We can also examine temporal patterns in how often a URL is presented and clicked, averaged over all queries where it is returned. In Figure 3, we provide an illustration of the number of clicks on the URL `http://en.wikipedia.org/wiki/Japan` over time.

These different kinds of dynamics, based in changing behaviors among a population of users, can be modeled systematically and predicted. Although the timing of the initial peak for the query *japan* may be hard to predict, there are many other cases in which search behaviors are easier to predict. Consider the queries presented in Figures 4 — 6. Figure 4 is a good example of a query showing a steady decrease in click frequency. Figure 5 shows a query undergoing a steady increase in click frequency. Figure 6 presents a query with periodic changes in frequency of clicks. Using

*Part of the research was performed while the author was at Microsoft Research.

Copyright is held by the International World Wide Web Conference Committee (IW3C2). Distribution of these papers is limited to classroom use, and personal use by others.
WWW 2012, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1229-5/12/04.

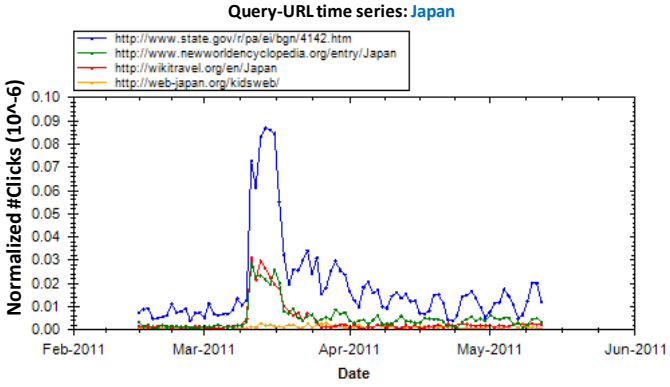

Figure 2: Time series (2/14–5/25, 2011) for sample clicked URLs for query *Japan* (normalized by total #clicks on each day).

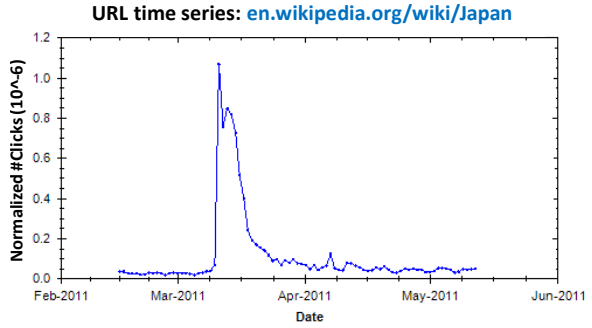

Figure 3: A Wikipedia article time series (2/14–5/25, 2011) for Japan (normalized by total #clicks on each day and position of the URL).

time series modeling, we can estimate these trends and periodicities and predict future values of the frequency of clicks. More accurate predictions of search behavior can be used to improve crawling policy and the ranking of search results.

The key contributions described in this paper are as follows: (1) We highlight the rich opportunities to study human behavior on the web and explore several models based on time series 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 properties seen in the dynamics of population behavior on the web, including *trend, periodicity, noise, surprise*, and *seasonality detection*. (2) We present a novel learning algorithm which we refer to as *dynamics model learner* (DML), that determines the right model for every behavior based on features extracted from logs of web behavior. We show that DML is superior to more traditional model selection techniques. (3) Finally, we perform empirical evaluation of our approaches over real-world user behavior, providing evidence for the value of our modeling techniques for capturing the dynamics of user behavior on the Web.

2. RELATED WORK

In this paper we examine how to model and predict peoples' Web search behavior. Of particular interest are changes in query frequency and clicked URLs over time. Researchers have examined previously how query volume changes over time. For example, some researchers have investigated aggregations of queries over time to understand changes in the popularity [21] and uniqueness of topics at different times of day [3]. Vlachos et al. [20] 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. Jones and Diaz [10] 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. Some studies [5, 16] used temporal patterns of queries to identify similar queries or words. Shokouhi [18] identified seasonal queries using time-series analysis. Kulkarni et al. [14] explored how queries, their associated documents, and the intentions behind the queries change over time. The authors identify several features by which changes to 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 intention behind queries. Kleinberg [11, 12] 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. [17] showed that queries reflect real-world news events, and Ginsberg et al. [8] used queries for predicting H1N1 influenza outbreak. Similarly, Adar et al. [2] 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 [7] and Dong et al. [1] developed algorithms for identifying queries that are related to breaking news and for blending relevant news results into core search results. Yang and Leskovec [22] studied temporal patterns associated with online content via a time series clustering approach that uses a similarity metric that is invariant to scaling and shifting. Information about time-varying user behavior was also explored by Koren et al. [13], who used matrix factorization to model user biases, item biases, and user preferences over time.

Although much has been done in the investigation of user behavior over time on the Web, few have pursued the construction of underlying models of this behavior, and then used these models for prediction of future behavior. We present methods for modeling behaviors over time that explain observed changes in the frequency of queries, clicked URLs, and clicked query-URL pairs.

3. TEMPORAL MODELING OF POPULATION BEHAVIOR ON THE WEB

In this section we present a modeling technique that can be used to capture the dynamic nature of web behavior. Of special importance in modeling web search behavior are global and local trends, periodicities, and surprises. We present the general theory behind this model (Section 3.1), the modeling techniques (Section 3.2), the learning procedure for these models from real-world data (Section 3.4), and how forecasting is performed using these models (Section 3.3).

3.1 State-Space Model

The state-space model (SSM) is a mathematical formulation used frequently in work on systems control [9] 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. Using 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:

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

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, it is usually assumed that $Y_{t+h} = Y_t$. In our work we assume (as commonly assumed in natural systems representations) that ϵ_t is a Gaussian process with variance σ^2. Equations (1) and (2) 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.

3.2 Population Behavior as a State Space Model

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 the search behavior of a population of people searching on the Web. We model the state X_t using a trend and seasonal component, as is often done for modeling time-series [15]. The trend represents the long-term direction of the time series. The seasonal (or periodicity) component is a pattern that repeats itself with a known periodicity, such as every week or every year. We now present models for user search behavior without trend and periodicity (using just historical smoothing), models with only trend or periodicity, and models that combine periodicity and trend. Each such combination will be represented by setting the scalars of the matrices $W(\theta), F(\theta), G(\theta)$.

3.2.1 Modeling with Smoothing

Figure 4 displays the click behavior of a population of users for the query *justin bieber*. The click popularity decreases with time, therefore models that simply average historical data, and then extrapolate a constant value as a prediction, can be poor models of the future. The Simple Holt-Winters model 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 equation

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

Intuitively, 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 3.4). 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

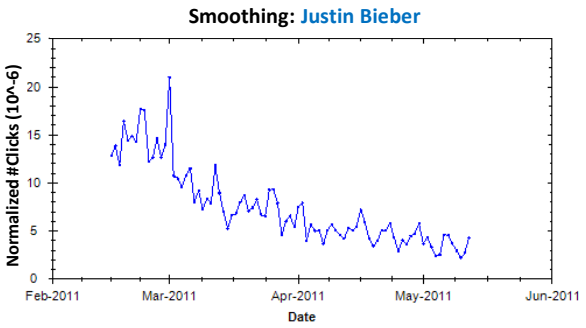

Figure 4: Query exhibiting behavior where historical data has little relevance for prediction (normalized by overall #clicks on each day).

defined as:

$$Y_t = y_t = l_{t-1}, \quad (3)$$
$$X_t = l_t = l_{t-1} + \alpha\epsilon_t.$$

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

3.2.2 Modeling with Local Trend

In many cases, using only one coefficient to decay previous data is not expressive enough to capture the dynamics of the system. One such case is a time series that exhibits a local trend. Figure 5 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 around this date shows a clear local growth trend in the time series. Simple smoothing of historical data would underestimate the dynamics of interest in this topic. A solution to this problem is the addition of a trend component to the simple Holt-Winters model:

$$y_t = l_{t-1} + d \cdot b_{t-1} + \epsilon_t, \quad (4)$$
$$l_t = l_{t-1} + b_{t-1} + \alpha\epsilon_t,$$
$$b_t = b_{t-1} + \beta^*(l_t - l_{t-1} - b_{t-1}),$$

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

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

Converting to SSM notation, let $X_t = (l_t, b_t)'$, then:

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

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

3.2.3 Modeling with Periodicity

Figure 6 shows query-click behavior for the query *consumer report* that exhibits weekly periodicity. Predictions based only on local trends or smoothing of the data will perform badly during the peaks. A solution is the addition of a periodic or seasonal component s_t to the Simple Holt-Winters model,

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

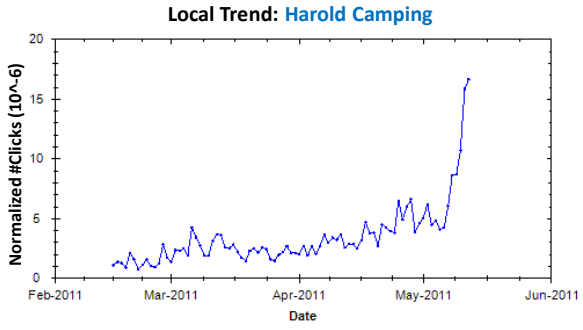

Figure 5: Query exhibiting behavior with local trend.

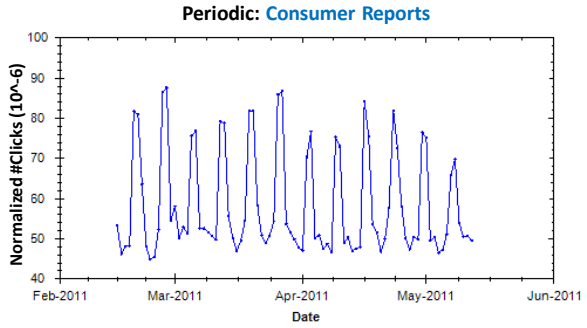

Figure 6: Query exhibiting a periodic behavior (normalized by overall #clicks on each day).

where m is the periodicity parameter that is estimated based on the data along with other parameters.

In SSM notation, the equation system can be written as:

$$\begin{aligned} y_t &= l_{t-1} + s_{t-m} + \epsilon_t, \\ l_t &= l_{t-1} + \alpha\epsilon_t, \\ s_{t-i} &= s_{t-i+1}, \\ &\dots, \\ s_t &= s_{t-m} + \gamma\epsilon_t, \end{aligned} \tag{5}$$

and for $X_t = (l_t, s_1, \dots, s_m)$, we can represent the parameters F, G, W in a form of matrices similar to the Trend Holt-Winters model formulation.

3.2.4 Modeling with Local Trend and Periodicity

In the previous models, the trend and periodicity were considered separately. However, for many queries, like the query *vampire diaries* shown in Figure 7, the trend and periodicity components are mixed together. The addition of trend and periodicity parameters produces the following model:

$$\begin{aligned} y_t &= l_{t-1} + d \cdot b_{t-1} + s_{t-m} + m_t + \epsilon_t, \\ l_t &= l_{t-1} + b_{t-1} + \alpha\epsilon_t, \\ b_t &= b_{t-1} + \beta\epsilon_t, \\ s_t &= s_{t-m} + \gamma\epsilon_t, \\ &\dots, \\ s_{t-i} &= s_{t-i+1}. \end{aligned} \tag{6}$$

3.2.5 Modeling Surprises

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 non-modeled processes in the open world. For example, in Figure 8, 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 have a strong effect on the forecast and parameter estimation of a model. We want 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.

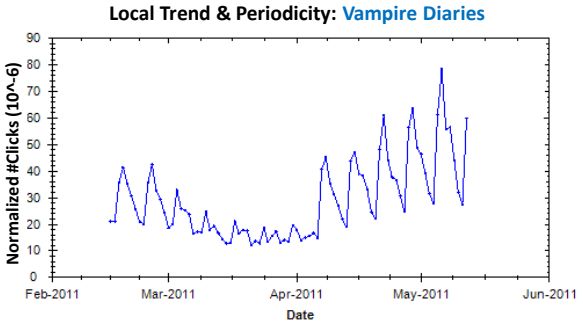

Figure 7: Query exhibiting periodic behavior with local trend (normalized by overall #clicks on each day).

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:

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

We discuss in Section 4.4 methods for identifying the surprises k_t in a time series.

3.3 Forecasting

Once this 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 (2), and based on these, a distribution of the future values Y_{n+1} is generated using the measurement equation (1). The final prediction can be generated by simply using the expected value of this distribution.

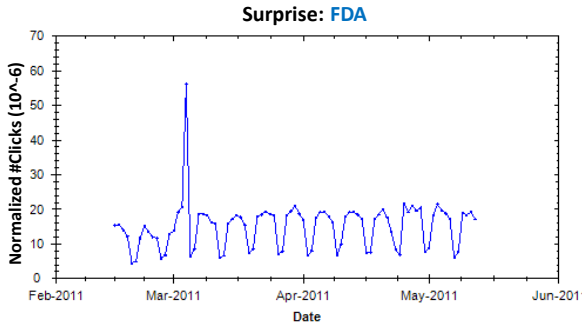

Figure 8: Query exhibiting behavior with surprises.

3.4 Parameter Estimation

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 [19] to optimize the model parameters based on training data. We assume here the following loss function, which is a common criteria for measuring forecast error:

$$F = \sum_{t=1}^{T} \epsilon_t^2. \tag{8}$$

We define this to be the likelihood that the residual $\epsilon = (\epsilon_0, \ldots, \epsilon_T)$ is zero. The initial values of X_0 are set heuristically, and refined along with the other parameters. The seasonal component m is estimated from the data by *auto-correlation* analysis (see Section 4.3).

4. LEARNING TO PREDICT THE BEHAVIOR OF A POPULATION OF SEARCHERS

As described in Section 3, different temporal models can be employed to represent user behaviors over time. We now present a known method for model selection based on the information criteria 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 the Web behaviors (Section 4.2). We conclude with models to detect periodicity (Section 4.3) and surprise (Section 4.4).

4.1 Information Criteria

We employ training data to select the best predictive model. A common practice in model selection is to optimize the Bayesian information criterion (BIC),

$$\text{BIC} = -2 \cdot \log(L) + q \cdot \log(n), \tag{9}$$

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

$$\text{BIC} = n \cdot \log(\sigma_e^2) + q \cdot \log(n), \tag{10}$$

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 Learning Temporal Behaviors

The criteria mentioned in Section 4.1 takes into account only the model behavior on the time-series values. 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. In this section, we 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 Learning Problem Definition

We start by motivating the algorithm formally and defining the learning problem. Let T be the discrete representation of time and O be the set of objects and let $f_i : O \times T \rightarrow Image(f_i)$ be a set of features. For example, O can be the 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, the learner is given examples of a *single object* o over some period $t_0, \ldots, t_n$, and produces a classifier $C : T \rightarrow \mathbb{R}$ based on those examples. In its use for prediction, the classifier is provided with an example of the object seen at training time t_i, and produces the prediction of its class at time t_{i+1} (for example, how many times the URL o is clicked tomorrow).

In regression learning, the learner is given examples of *multiple objects* $O' \subset O$ and produces a classifier $C : O \rightarrow \mathbb{R}$ based on those examples. During prediction, the classifier is given an example of an object o_i, that has not been seen before and produces the prediction of its class.

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 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 classifier 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 class at time t_{n+1}.

4.2.2 Learning Algorithm

In this section we present a new learning algorithm, *dynamics model learner (DML)*, 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 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 Y_i on the learning period. We then check for mean square 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 $m(i,o)$ be the model index with the lowest MSE on the test period for the object o. We construct a set of examples $E = \{\langle f_1(o), \ldots, f_n(o)\rangle, m(i,o) | o \in O'\}$ — a vector representing an object and labeled with 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 $m(i, o_{target}) = C(o_{target})$ represents the most appropriate model for the object o_{target}. We train model i using the behavior of o_{target} during $t_1, \ldots, t_{n+\sigma}$. A detailed algorithm is illustrated in Figure 9. An example of a learned decision tree is shown in Figure 10. In this figure, we see that the periodic model should be applied only on queries considered periodic (as defined in Section 4.3), along with other characteristics. If the query is not periodic, either the trend or smoothing model should be applied, depending on the query shape (quefrencies values) (see Section 4.2.3). Thus by using the DML model we learn to apply the correct model for every query, URL or query-URL pair.

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

Figure 9: 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 Temporal 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 (partial description of the features is available on [1]), and group them into three groups: aggregate features of the time series o, shape feature of the time series o, and other domain-specific features such as the query class.

Aggregate Features. Features such as 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* (see Section 4.3) and *number of surprises* (see Section 4.4). We also consider the *size of the spikes* during the surprises (the magnitude of the disturbance).

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 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

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

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. Application of a log function on the resulting frequency vector and an additional FFT operator transforms it to what is called the *cepstral domain* [6], and the values of x in this product are called *quefrencies*. We consider these features to represent the shape of the series.

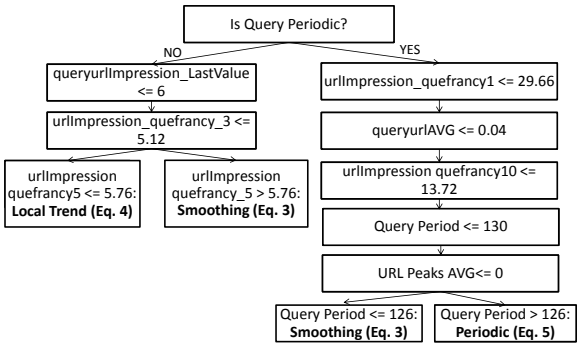

Figure 10: A part of the learned dynamic model.

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:

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

where $u_1, \ldots, u_n$ are the clicked URLs at time t, and $click(u_i, q)$ is the number of clicks on u_i for a query q at time t. 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) [4]. We chose to classify into approximately 40 overlapping topics, such as travel, health, and so on.

4.3 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. This method can be used to detect repeating patterns with noise, and is defined as

$$Autocorrelation(f,h) = \int_{-\infty}^{\infty} f(t+h) f(t)\, \mathrm{d}t, \qquad (11)$$

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:

$$Autocorrelation(f,h) > \omega, \qquad (12)$$

[1] http://www.technion.ac.il/ kirar/Datasets.html

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 (see Section 6.2).

4.4 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 neighboring 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 surprises that have long impact on the model should be considered as a surprise. 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 criteria). When the model stops improving, we output the surprises. The full surprise detection algorithm is given in Figure 11.

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 $Events$

Figure 11: Detecting surprises in time series.

5. EXPERIMENTAL SETUP

We now outline the experimental methodology we employed to test the temporal models that we have presented. We first describe the setup for the prediction experiments, and then for the periodicity and surprise detection experiments.

5.1 Prediction Experiments

5.1.1 Empirical Methodology

We perform several prediction experiments, namely predicting query, URL, and query-URL click frequencies. We also modeled and predicted query, URL and query-URL impressions. Because of space limitations, we present results only for click behavior. In every experiment, a time series is created for the behavior for 5 SSM models (Section 3), 2 selection models (BIC (Section 4.1), DML (Section 4.2)), and 4 baseline models (Section 5.1.3). 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 (to avoid overfitting of a specific day).

5.1.2 Data

For the experiments, we use a dataset containing query and URL user visit activity obtained from Bing 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 5 clicks per day. For each query, we consider the first 4 top URLs by click frequency. We normalize every activity time series by the total number of activities on that day, to known seasonality of high searches over weekends. For query and URL pairs, we also normalize by the number of clicks on the URL at the position at which it was displayed in the displayed ranked results for the query, producing a value which is not dependent on the position.

We focus on several types of queries:

1. **General Queries** We first present prediction over a general sample of queries issued to Bing. For this purpose we use 10000 queries randomly sampled without repetition on a given day, and 35,862 clicked URLs.

2. **Temporal Queries** 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 will benefit from temporal modeling.

   (a) Dynamic Queries Queries like the *japan* query 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 URLs were used.

   (b) 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. In the experiments, a total of 330 and 1320 URLs were used.

(c) 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. In the experiments, a total of 1836 and 7344 URLs were used.

5.1.3 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**: A simple average of the time series: $w(i, y_i) = 1$.
2. **LIN**: The linear weighting function: $w(i, y_i) = i$.
3. **POW**: The power weighting function (in the experiments we set $p = 2$): $w(i, y_i) = i^p$.
4. **YES**: The 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: $w(i, y_i) = \begin{cases} 1 & i = t - 1 \\ 0 & \text{otherwise.} \end{cases}$

5.2 Periodicity Detection Experiment

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 [18] that contained 259 queries, each annotated as seasonal or not by several judges. We compare our method for periodicity detection against the method described in Shokouhi [18], 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.

5.3 Surprise Detection Experiment

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 queries behavior for the years 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.

6. EXPERIMENTAL RESULTS

We now summarize the results for prediction for different temporal models, and also provide results of experiments on detecting periodicity and surprise.

6.1 Prediction Experiments

We first summarize the different prediction results for query, URL, and query-URL click behavior.

6.1.1 Query Prediction

Query click prediction results are shown in Table 1. The table reports prediction errors, so the smaller the number the better the 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 extremely 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 most 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 which require new results need the most relevant new information. Thus, old data interferes with prediction. Most of 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. A similar pattern is shown for the Alternating queries. For the Temporal-reformulations query class, the baseline that performs simple averaging of historical data yields the best results. In this category, the best performing temporal models are those that take into account the periodicity of the query.

Overall, predictions were 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 5 months, such longer-term periodicities were not detected.

6.1.2 URL Prediction

URL click prediction results are given in Table 2. 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 the clicked URL behavior 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, here again the

	Baselines				Temporal SSM Models					Model Selection	
Query Type	AVG	LIN	POW	YES	SMOOTH	TREND	PERIODIC	TREND+ PERIODIC	SURPRISE	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

Table 1: Query number of click prediction Error (lower numbers indicate higher performance). Statistically significant results are shown in bold.

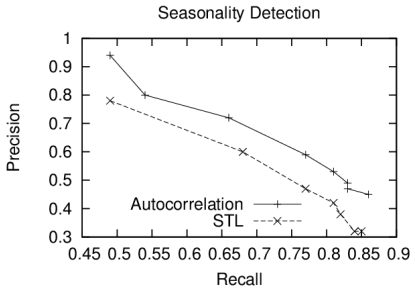

Figure 12: Comparison of STL method with Autocorrelation method for seasonality detection by precision (y axis) and recall (x axis).

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 came from the Dynamic set.

6.1.3 Query-URL Prediction

Query-URL pair click prediction results are shown in Table 3. We first observe the DML has the best performance for the General and Temporal Reformulation queries. The temporal model which smooths across surprises (Surprise) is especially useful for the Dynamic query type.

Across Tables 1–3, there appear to be two groups of SSM models: (1) Smooth and Trend models (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 model behave similarly. Sometimes one group performs better than the other, but the groupings are consistent across the three tables and four query types.

6.2 Periodicity Detection Experiment

To evaluate our periodicity model, we evaluate performance for different autocorrelation thresholds, ω. Figure 12 shows the precision-recall results for our autocorrelation model (Autocorrelation) compared to the baseline model (STL). The Autocorrelation method proposed in this work reaches the same maximum recall as the state-of-the-art STL 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.

6.3 Surprise Detection Experiment

Table 4 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 by the surprise detector.

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

Table 4: Recall and precision of the different surprises detectors. Statistically significant results are shown in bold.

7. 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 different temporal representations and learning procedures and showed how we can construct models that predict future behaviors from historical data.

Temporal models performed better than the baseline models that use the same weighting of historical data for all queries in almost all cases. We also found that different temporal models are appropriate for different types of queries. Query click behavior tends to be rife with disturbances (surprises). Thus, models that identify and smooth out such disturbances tend to predict well. For specific types of queries, such as Dynamic and Alternating queries, trend models are more appropriate. URL click behavior tends to exhibit 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. We found the dynamics seen in query-URL click behavior to be interesting sources of insight into changes in users' intentions over time. We observed that, in general, the application of correct smoothing or incorporating trend is the best methodology.

	Baselines				Temporal SSM Models					Model Selection	
Query Type	AVG	LIN	POW	YES	SMOOTH	TREND	PERIODIC	TREND+ PERIODIC	SURPRISE	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

Table 2: URL number of click prediction Error (lower numbers indicate higher performance). Statistically significant results are shown in bold.

	Baselines				Temporal SSM Models					Model Selection	
Query Type	AVG	LIN	POW	YES	SMOOTH	TREND	PERIODIC	TREND+ PERIODIC	SURPRISE	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

Table 3: Query-URL number of click prediction Error (lower numbers indicate higher performance). Statistically significant results are shown in bold.

We introduced a novel Dynamics Model Learner (DML) algorithm, that learns the correct model to apply for every query, URL or query-URL pair that it is presented with, 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 in all groups. 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. Query click prediction can be used to improve query suggestions to present the most appropriate suggestions at the time the query is issued. URL click prediction can be used to improve re-crawling strategies, by focusing crawling efforts on URLs that are likely to be clicked. And, Query-URL prediction can induce better ranking that is more aware of the user query-URL temporal intent. Additionally, the models we have presented in this work 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.

8. ACKNOWLEDGMENTS

We thank Dan Liebling for help with obtaining the user behavior data, and Chris Meek and Kuansan Wang for fruitful discussions. We also thank Milad Shokouhi for providing the labeled seasonal queries.

9. REFERENCES

[1] Z. Zheng G. Mishne J. Bai R. Zhang K. Bichner C. Liao A. Dong, Y. Chang and F. Diaz. Towards recency ranking in web search. In *WSDM*, 2010.

[2] E. Adar, D. S. Weld, B. N. Bershad, and S. D. Gribble. Why we search: visualizing and predicting user behavior. In *WWW*, 2007.

[3] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. Hourly analysis of a very large topically categorized web query log. In *SIGIR*, 2004.

[4] P. N. Bennett, K. Svore, and S. T. Dumais. Classification-enhanced ranking. In *WWW*, 2010.

[5] S. Chien and N. Immorlica. Semantic similarity between search engine queries using temporal correlation. In *WWW*, 2005.

[6] D. P. Skinner D. G. Childers and R. C. Kemerait. The cepstrum: A guide to processing. *IEEE*, 65:1428–1443, 1977.

[7] F. Diaz. Integration of news content into web results. In *WSDM*, 2009.

[8] J. Ginsberg, M. Mohebbi, R. Patel, L. Brammer, M. Smolinski, and L. Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–4, 2009.

[9] J.Durbin and S.Koopman. *Time Series Analysis by State Space Methods*. Oxford University Press, 2008.

[10] R. Jones and F. Diaz. Temporal profiles of queries. *ACM Trans. Inf. Syst*, 2004.

[11] J. Kleinberg. Bursty and hierarchical structure in streams. In *KDD*, 2002.

[12] J. Kleinberg. Temporal dynamics of on-line information systems. *Data Stream Management: Processing High-Speed Data Streams. Springer*, 2006.

[13] Y. Koren. Collaborative filtering with temporal dynamics. In *KDD*, 2009.

[14] A. Kulkarni, J. Teevan, K. M. Svore, and S. T. Dumais. Understanding temporal query dynamics.

[15] J.Ord R. Hyndman, A.Koehler and R.Snyder. *Forecasting with Exponential Smoothing (The State Space Approach)*. Springer, 2008.

[16] K. Radinsky, E. Agichtein, E. Gabrilovich, and S. Markovitch. A word at a time: Computing word relatedness using temporal semantic analysis. In *WWW*, 2011.

[17] K. Radinsky, S. Davidovich, and S. Markovitch. Predicting the news of tomorrow using patterns in web search queries. In *WI*, 2008.

[18] M. Shokouhi. Detecting seasonal queries by time-series analysis. In *SIGIR*, 2011.

[19] Jan A. Snyman. *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*. Springer, 2005.

[20] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *SIGMOD*, 2004.

[21] P. Wang, M. W. Berry, and Y. Yang. Mining longitudinal web queries: trends and patterns. *JASIST*, 54:743–758, 2003.

[22] J. Yang and J. Leskovec. Patterns of temporal variation in online media. In *WSDM*, 2011.