Predicting Content Change on the Web

Kira Radinsky*
Technion–Israel Institute of Technology
Haifa, Israel
kirar@cs.technion.ac.il

Paul N. Bennett
Microsoft Research
Redmond, WA, USA
paul.n.bennett@microsoft.com

ABSTRACT

Accurate prediction of changing web page content improves a variety of retrieval and web related components. For example, given such a prediction algorithm one can both design a better crawling strategy that only recrawls pages when necessary as well as a proactive mechanism for personalization that pushes content associated with user revisitation directly to the user. While many techniques for modeling change have focused simply on past change frequency, our work goes beyond that by additionally studying the usefulness in page change prediction of: the page's content; the degree and relationship among the prediction page's observed changes; the relatedness to other pages and the similarity in the types of changes they undergo. We present an expert prediction framework that incorporates the information from these other signals more effectively than standard ensemble or basic relational learning techniques. In an empirical analysis, we find that using page content as well as related pages significantly improves prediction accuracy and compare it to common approaches. We present numerous similarity metrics to identify related pages and focus specifically on measures of temporal content similarity. We observe that the different metrics yield related pages that are qualitatively different in nature and have different effects on the prediction performance.

Categories and Subject Descriptors

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

Keywords

Web Change Prediction, Web Dynamics, Web Crawling

1. INTRODUCTION

In many retrieval frameworks, web pages are often treated as static documents. However, in reality, pages are dynamic channels whose contents change with time. Some pages, like news entry points, change on a daily basis with regularity. While others, like faculty home pages, may be updated with far less frequency and regularity. These changes may involve significant changes in content, the addition or deletion of hyperlinks, or the addition of hyperlinks to previously unknown web pages. Furthermore, these changes in content drive changes in the query distribution that lead to a page, and impact the relevance of a page to the query.

The ability to predict key types of changes can be used in a variety of settings. For example, accurate prediction of a significant change in page content enables an improved incremental crawling strategy that only recrawls pages when necessary [18]. Similarly, accurate prediction of a new hyperlink to a previously unknown (i.e., non-indexed) web page, enables one to efficiently discover pages that should be indexed. In a personalized retrieval setting, accurate prediction of re-visitation to a page with new content enables one to push new content to a user.

When modeling any variable of interest as a function of time, it is natural to consider the past observed values as a basic means of prediction [8]. For example, in this setting, when predicting whether a page will change, we can simply predict based on the change frequency for the page observed over some past historical window. However, there are other signals that provide information beyond change frequency.

In particular, the content of a page enables better prediction of its change [4, 24, 23]. It provides a finer distinction of the degree of change that a page is undergoing, and the relationship among these changes. For example, in a series of ten days, a page may experience small changes on every day or large changes. These changes in content might be similar to each other or vastly dissimilar. For a page that experiences vastly dissimilar changes on a regular basis, the changes may be considered more significant (since the content change implies a need for reindexing). On the other hand, similar changes with the same regularity may be considered less significant. To distinguish these cases, the appropriate features need to be encoded.

Pages that are related to the prediction page may also change in a similar, although not necessarily, synchronous fashion [24, 23, 9, 12, 13]. These pages would then provide a useful signal for determining whether the prediction page will change. The *strength* and the *type* of the relationship between those pages is likely to be indicative of how well one can predict the change of the other. In the web setting, natural choices of relationship types include: distance in the web graph, similarity in content, and similarity in temporal change pattern. We explore all of these relationship types and evaluate their relative contributions over the page's content alone. In particular, our exploration of similarity in content *over time* through the use of cross-correlation and the dynamic time warping (DTW) algorithm is especially

*This research was performed while the author was at Microsoft Research.

WSDM'13, February 4–8, 2012, Rome, Italy.

novel. Both of these provide more robust computation of similar temporal change patterns by dealing with the case where content between two pages may be correlated in time but slightly out of phase.

Additionally, even after identifying other related information sources, there is a question as to how to best incorporate this information into a machine learning algorithm. We develop an expert combination approach where the experts are selected according to their degree of "relatedness" and demonstrate the superiority of this modeling choice to other combination and relational learning techniques.

This paper is organized as follows: In Section 2 we describe the related work. In Section 3, we present the formal problem description and outline the challenges and research questions. We then present a general characterization of the types of information sources available when predicting page change using past history (Section 4.1). Using this characterization, we develop specific feature sets that capture aspects of each of these information sources in a useful way for modeling change (Section 4.2). In particular, in addition to the past history of a page's change, we also incorporate features based on the page content of the page as well as the content of *related* pages. For each information source we provide several algorithms to combine the information in a learning setting. In particular, we present a novel algorithm to combine the information in an expert prediction framework (Section 5). We examine techniques for identifying related pages, including pages that are close in the web graph, pages with similar content, and pages with correlated content across time – where the last two prove to be especially effective (Section 6). Finally, in an empirical study, we not only demonstrate the effectiveness of our methods, but also explore alternative modeling choices, analyze the qualitative differences in the expert selection methods, and present the superiority of our prediction methods in a crawling application, which strives to maximize the *freshness* of its index (how many pages in the index remain up-to-date as compared with the online web copies).

2. RELATED WORK

Attempting to predict the probability of a web page changing is not a new problem and has been studied by a variety of researchers [9, 8, 4, 24, 23]. This has primarily been motivated by the incremental web crawling setting [18]. In this setting, recrawling a web page is linked to the probability of its change, and the goal is to maximize some tradeoff between freshness and coverage of a web index. A crawl policy that optimizes this tradeoff usually focuses only on the probability of change for pages that do not change too often or very rarely. The reason for this is that, under certain assumptions, recrawling pages that change very frequently dominates crawl resources and actually hurts the overall utility of a crawl by ensuring only these very frequently changing pages are kept fresh [10, 7, 22]. In line with these studies, we restrict our study to pages above a minimum threshold of change and below a maximum threshold.

However, while freshness may be one metric for measuring the utility of recrawling a web page, it may not reflect impact on the user of a search engine. As a result, Wolf *et al.* [25] look at minimizing embarrassment – the probability a search engine user clicks on a result URL whose live page doesn't match the query. This introduces a notion of the negative utility a user may experience but not what gains may be experienced from crawling. Therefore, Pandey & Olston [19] consider a user-centric utility. This

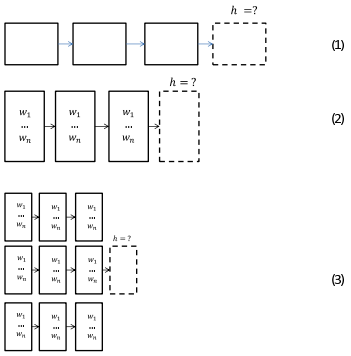

Figure 1: Fully observed setting. Each square represents a page, where w_i are the page contents. (1) 1D setting (2) 2D setting [3, 6] (3) 3D setting

utility weights each page by: (1) an observed query load on a search engine, and (2) the impact that downloading a new copy of the page has on the search ranking (as estimated via previous change). Similarly, Ali & Williams [3] measure the impact under a specific query load of several simple prediction schemes. They found that simply recrawling pages whose previous two observed snapshots changed by more than (approximately) 20 words significantly reduced crawling compared to recrawling all recently changed pages. This policy also maintained effective ranking as measured under the query load. In line with these, we restrict ourselves to predicting significant changes to a search page. In our case, we define this as cases where the page's change relative to the last observed snapshot is above a significance threshold (as defined in Equation 1).

In terms of methods of predicting change, several works [6, 11, 7] use past change-frequency and change-recency of a page, typically together with an assumption of a Poisson process, in order to predict whether the page will change. This is essentially equivalent to our baseline model described in Section 5.1. Both Barbosa *et al.* [4] and Tan & Mitra [23] demonstrate that using content-based features can improve prediction over just change-frequency alone [6]. The first of these works primarily uses static features (content features, such as file size available from a single snapshot) in addition to change history, while the second also considers dynamic features (features that change between two snapshots, such as the cosine similarity between the page contents). In our work, we focus on dynamic content features since we are concerned with predicting *significant* content change, and as argued by Ali & Williams [3], many of those static features are usually predictive of any type of change (e.g., new advertisements, images, etc.), that are usually uninteresting or do not impact indexing. Nevertheless, our methods are general and could easily incorporate such additional features.

Fetterly *et al.* [13] note that not only is the previous degree of content change a good predictor of future change, but that change is also correlated at the top-level domains of web pages. Cho & Ntoulas [9] exploited this type of correlation structure at the website level by using a sample of webpages from a website to estimate the change probability of any page from the website. Tan *et al.* [24, 23] extend this idea to the more general case by clustering pages based on static and dynamic content features into clusters that are more homogeneous in terms of change patterns before conducting the sampling based approach. Both of these can be viewed

as methods of identifying related pages. We also identify related pages, but use them to directly predict the probability of change without having sampled from the current time slice. Additionally, we investigate directly estimating relatedness via temporal content similarity – which could easily be used to cluster in a sampling based approach.

In contrast to previous work: (1) we focus on the task of predicting *significant* changes rather than any change to a web page; (2) we develop a wide array of dynamic content-based features that may be useful for the more general temporal mining case beyond crawling; (3) we explore a wide variety of methods to identify related pages including content, web graph distance, and temporal content similarity – where the last is both especially novel and effective; (4) we derive a novel expert prediction framework that effectively leverages information from related pages without the need for sampling from the current time slice.

3. PROBLEM FORMULATION

We consider the general setting where we want to estimate a function h for each page. This h can indicate several types of page change (Section 3.1). This function can be estimated using different information types (Section 3.2), which have different observability when estimating h (Section 3.3).

3.1 Types of Web Page Change

Let O be a finite set of pages, and let $h : O \times \mathfrak{T} \rightarrow \mathbb{R}$ be a temporal goal function, that provides information about the state of a page $o \in O$ at time $t \in \mathfrak{T}$, where $\mathfrak{T}$ is a discrete representation of time. Depending on the targeted application, a function h can be defined in numerous ways. For example, the change may be:

1. Whether the page $o \in O$ changed significantly. This can be used to identify when a page is stale enough to be recrawled.

2. Whether the change in page $o \in O$ corresponds to a change from non-relevant previous content to relevant current content for some specified query load. The function not only indicates whether the page is stale, but also that the change will result in the page being surfaced for a query.

3. Whether there is a new out link from page $o \in O$. This helps indicate when recrawling a page will lead to discover a new portion of the web graph, discover new anchor text, or impact link analysis metrics.

We focus on page changes of type 1, although our approach is readily applicable to the other change types. We consider a *significant page change* to be a change in content between the content vector at time t and the content vector at time $t + 1$ whose word distribution is different in a statistically significantly sense (as measured by a chi-squared test) from that of the previous day. That is, the p-value produced by performing the test satisfies:

$$p^{\chi^2}(\text{ContentDist}(t), \text{ContentDist}(t+1)) < 0.05 \qquad (1)$$

3.2 Information Sources

As we discussed before, when estimating h for each page, we can consider several types of information. In many previous studies [6, 11, 7, 3], only information about the changing object was used, e.g., the change rate of the specific page. We propose a general-unifying framework (formally discussed in Section 4.1) of the possible information types, ranging from only information about the changing object to information about all the other related objects. We divide the information types into the following information settings:

1. 1D setting: In this setting, only information about h of the page itself in previous time-stamps is used to estimate h. For example, only information involving the rate of the page change itself is used to estimate future page change. This setting was explored in the past [11, 7], and we consider it as one of the baselines.

2. 2D setting: In this setting, a wider range of information about the page h itself is used to estimate h. We consider the page as an entity with features which can be used to understand its future h, e.g., future change rate. This setting was explored by several works [3, 6], and we consider it as one of the baselines as well.

3. 3D setting: In this setting, not only is the information about the specific page used, but also the relations to the other pages are considered in determining the probability of a future h value of a page, e.g., the future change of a page.

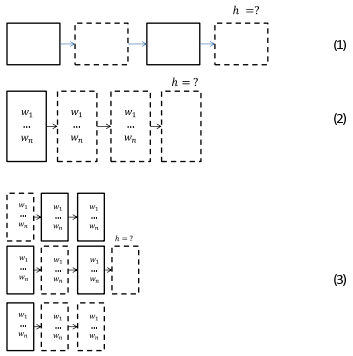

Figure 2: Partially observed setting.

3.3 Information Observability

In a *fully-observed scenario*, at each time stamp t, all the above information from the previous n time stamps is available. This is a scenario, where, given a set of k periodically observed pages over a period of n, the goal is to induce a function h that can be used to estimate page changes within this group in the following time-stamp. Figure 1 illustrates the three information settings in the fully-observed scenario.

The more general scenario, is a *partially-observed scenario*, where only partial information about the objects in previous times is given. That is, at each time stamp t, only the information about some subgroup is available to the learner. This is an online scenario, where, for example a crawling system doesn't observe the information about non-crawled pages. It only crawls pages it believes to have changed, and therefore only information about those crawled pages is given to the learner to induce an approximation of h. The three information settings in the partially-observed scenario are illustrated in Figure 2.

In this work we focus on the fully-observed setting and provide algorithms based on the 1D, 2D and 3D settings. We wish to derive insights and understanding of this setting before generalizing to the partially-observable scenario. Additionally, in Section 8.3 we empirically show a simple application of our algorithms for the partially-observed setting,

where the objects are iteratively sampled to gain more information. Section 4.1 formalizes the problem from a machine learning perspective, and Section 5 outlines our algorithmic solution to the problem.

4. SOLUTION FRAMEWORK

In this section, we formalize the learning problem and provide an algorithmic framework for predicting change. Unlike standard learning algorithms that assume a static feature set and a single object type, we present a formalism that handles temporal features, derived from both single and multiple objects over time. We then discuss the specific features for page change prediction that we build using this framework.

4.1 Formal Framework

Let O be a set of objects. Let $C \subseteq O$ be a goal concept. In our problem, O is the set of pages and C is the set of changing pages. In the traditional (binary) supervised learning setup, we define the goal function $h : O \rightarrow \{0, 1\}$, where $h(o) = 1 \leftrightarrow o \in C$. Let $E = \{\langle o_1, h(o_1)\rangle, ..., \langle o_K, h(o_K)\rangle\}$, where $o_i \in O$ and $h(o_i) \in \{0, 1\}$, be a set of training examples. To extend the basic setup to include classes and features that dynamically change over time, we assume $\mathfrak{T} = \{0, 1, ..., \infty\}$ to be a discrete representation of time. We define $h : O \times \mathfrak{T} \rightarrow \{0, 1\}$ to be a *temporal goal function*, and $E = \{\langle\langle o_1, t_1\rangle, h(o_1, t_1)\rangle, ..., \langle\langle o_K, t_n\rangle, h(o_K, t_n)\rangle\}$ to be a set of *temporal training examples*, where $o_i \in O$, $t_j \in \mathfrak{T}$ and $h(o_i, t_j) \in \{0, 1\}$.

The basic algorithm for predicting page change, as currently used in many applications [6, 11, 7], is prediction based on the $h(o_i, t_j)$ of past training examples of o_i. Using the previous notation, we call this information setting the *1D setting*, as it only utilizes the class (in our case, page change) behavior.

Let $F = \{f_1, ..., f_{|F|}\}$ be a set of feature functions, where $f_i : O \rightarrow Im(f_i)$. A traditional learning algorithm takes E and F as input and produces a hypothesis $\hat{h}$ which is a good approximation of h. We define $F = \{f_1, ..., f_{|F|}\}$ to be a set of *temporal feature functions*, where $f_i : O \times \mathfrak{T} \rightarrow Im(f_i)$. That is, $f_i(o, t)$ is the value of the feature i of object o at time t. A *temporal learning algorithm* receives as input E and F and produces a temporal hypothesis $\hat{h}$.

Traditionally in machine learning, the building blocks of a classifier $\hat{h}(o)$ are the features of the object $f_1(o), \ldots, f_{|F|}(o)$, e.g., changes in a page might depend only on its features, e.g., its terms. In our extended framework, $\hat{h}(o, t)$ can use, in addition to $f_i(o, t)$, also $f_i(o, t')$ for $t' < t$. For example, the page change might depend on the change direction values over the last several days. Using the previous notation, we call this information setting a *2D setting*, as it only utilizes the features of the page itself.

Alternatively, in our framework, $\hat{h}(o, t)$ can use features of other objects, $f_i(o', t')$ for $o' \neq o, t' < t$. For example, changes in other pages from previous observable times. We also assume a relation function $R_i : O \times O \leftarrow R$ between the objects. These functions can also be considered as a weighted graph structure over the objects. Thus, $\hat{h}(o, t)$ can use features of only related objects using metrics over the structure of relations R_i. In the most general setup, $\hat{h}(o, t)$ can combine the above two approaches and use $f_i(o', t')$ for $t' < t$ and $o' \neq o$. For example, to determine whether a certain page will change tomorrow, the classifier can use various parameters of other pages over the last week. A more extended view of the features f_i, are those that take into account several objects in an aggregated feature, i.e., $f_i : O^K \rightarrow Im(f_i)$. For example, a feature that determines the content similarity of the predicted page o with the other objects that are fed as input to produce $\hat{h}(o, t)$. In the previous notation, we call this setting a *3D setting*, as it utilizes the features of the page itself and other pages as well.

4.2 Framework Implementation

The previous section discussed the formal framework of the information sources. We now discuss the specific features that we implement for content change prediction.

4.2.1 2D Features

Given the time of classification t, the goal is to predict whether page A will change at time $t + \sigma$, where σ is the prediction interval. We denote the representation of a page to be classified at time t as $A(t)$. Let l be the regression parameter, i.e., the window size from which features are generated. For example, for $l = 2$ we create page features using its behavior over the last 2 time stamps (not to be confused with the number of examples n, which is the number of objects usually fed into a classifier). We define the following features for learning page change:

1. Frequency of change in page A during the window l. This feature is equivalent to the information captured by the baseline in Section 5.1.

2. Term unigram probability distribution of the words in page $A(t)$: $\{\frac{c_{i,A(t)}}{\sum_j c_{j,A(t)}}\}$ where $c_{i,A(t)}$ is the number of occurrences of the considered term (w_i) in page $A(t)$, and the denominator is the sum of number of occurrences of all terms in page $A(t)$, that is, the size of the document $A(t)$. These features are meant to capture the intuition that particular words on a page might be predictive of page change. For example, on a local organization's page the words "upcoming meeting" might be predictive of a change that notes of the meeting will soon be posted.

3. Features representing the magnitude of the change vector between the content at every time point in the window and the content of the page at time t. Formally, a set of l features representing how big of a change A experienced compared to each previous day in the window: $\{||A(t) - A(t - i)|||i \in [1 \ldots l]\}$. Intuitively, the magnitude of change in content between the last observed content (t) and the content on any given day in the window may be predictive of change. For example, a stable set without change may increase the chance of no change. While a series of small changes may presage an upcoming update. This will be dependent on the particular page and these features provide the expressiveness to learn such behaviors.

4. For each pair of prediction-interval separated timesteps in the window, we define features measuring the magnitude of the change vector in page content in a single timestep. Formally, a set of l features representing how big of a change A was experiencing compared to previous days in the window: $\{||A(t - i) - A(t - i - \sigma)|||i \in [1 \ldots l]\}$. Intuitively, this captures the magnitude of change typically seen given the prediction interval size.

5. Features representing whether the page changed on every time in the window. We define change of page at time t as:

$$\text{change}(A(t)) = \begin{cases} 1 & \text{if } A(t) \text{ changed} \\ 0 & \text{if } A(t) \text{ did not change} \end{cases}$$

We consider the following features: $\text{change}(A(t-1)), \ldots, \text{change}(A(t-l))$ representing whether the page changed significantly on the days before the prediction. Intuitively, these features are useful when an upcoming change might be predicted by any change (or lack thereof) a fixed number of days in the past or a regularity of change/stability might indicate a continuation of the pattern.

4.2.2 3D Features

Additionally, we can incorporate features of other pages B on the web. Their change and features might also reflect on the change in page A. The first five types of features we define are exactly analogous to those defined in Section 4.2.1, and they are meant to capture similar motivations but with the shift that if the feature types from the same page are useful, then the same ones from a related page may prove useful. We also add two new feature types targeted at representing the relationship between the two pages. We define the following set of features for each page B:

1. Frequency of change in page B during the window l.
2. Term unigram probability distribution of the words in page $B(t)$: $\{\frac{c_{i,B(t)}}{\sum_j c_{j,B(t)}}\}$.
3. A set of l features representing how big of a change B experienced compared to previous days in the window: $\{||B(t) - B(t-i)|||i \in [1 \ldots l]\}$.
4. A set of l features representing how big of a change B was experiencing compared to previous days in the window: $\{||B(t-i) - B(t-i-\sigma)|||i \in [1 \ldots l]\}$.
5. Feature $\text{change}_{t-\sigma-1}(B), \ldots, \text{change}_{t-l}(B)$ representing whether the page changed significantly on the days before the prediction.
6. A set of l features representing the similarity of page B to the prediction page A: $\cos(A(t), B(t))$. Intuitively, the degree of content similarity of the pages at time $t \in [t \ldots t-l]$ may indicate the strength of predictivity between the pages.
7. A set of l features representing the similarity in change of page B to the content of page A. Formally, let $\delta(B)_i = B(t) - B(t-i)$, and $i \in [1 \ldots l]$, thus this feature is defined as: $\cos(\delta(B), A)$. Intuitively, not only may the degree of content similarity be indicative of the relationship strength, but also the similarity between the content of B that changed and A's full content. We also consider a set of l features representing the similarity of the changes of page B to the changes of the prediction page A: $\cos(\delta(B(t-i)), \delta(A(t-i))$ for $i \in [1 \ldots l]$.

5. LEARNING ALGORITHMS

In this section we outline the specific algorithms based on the solution framework we described in the previous section.

5.1 Baseline Algorithm

The basic algorithm for predicting page change, as currently used in many applications [6, 11, 7], is prediction based on the probability of the page to change significantly based on the past training examples. That is, $h(o_i, t_j) = 1$ with probability:

$$p(h(o_i, t_j) = 1 \mid h(o_i, t_k) \in E \text{ where } t_k < t_j \text{ and } (t_j - t_k) \leq l).$$

The parameter l is typically referred to as a window size and is a user-specified parameter.

5.2 Single Expert Algorithm

The single expert is an algorithm that represents the pages with a set of features. Given M time stamps, we create training examples with the features described in Section 4.2.1. For example, if we want to predict for 5 days in the future ($\sigma = 5$) and we have a total of 300 days for training, we can prepare training objects for a learner. Each object is made of features of l days, where l is the regression parameter. For example, for $l = 3$ a total of $300 - \sigma - l + 1 = 293$ examples are created to train a learner with. Given prediction interval σ, each example is labeled with the h value of the page in σ time stamps, i.e., $h(o, t + \sigma)$. Those E examples are used to train a classifier C. In our experiments we use SVM classifiers for this purpose.

During prediction, the set of the above features is extracted for the test instance and given as input to C. In our example, we are given another set of consecutive 21 days (separate from training period), from which $21-\sigma-l+1 = 14$ testing objects are created, on which the learner will be tested. In this work we consider the case where $\sigma = 1$ and $l = 1$, i.e., we observe the page as it is today and predict how it will be tomorrow. We then output the prediction of the classifier.

5.3 Algorithms Using Related Objects

In this section we discuss algorithms that consider both the page's features and features of other pages. We first present a novel algorithm that models the related pages as experts, and follows a voting scheme (Section 5.3.1) In order to investigate whether this framework is in fact advantageous we also investigate several alternative mechanisms for incorporating information from related pages: an algorithm that collects the information of the other sources into a single expert (Section 5.3.2), and algorithms that use multiple models to boost performance (Section 5.3.3).

5.3.1 Multiple-Experts Algorithm

In this section we propose a novel algorithm, that performs a temporal relational choice of experts. By temporal experts we mean related pages that predict change. The novel algorithm we present here takes into consideration the structure of the object-relations network, and builds relational experts. Each page will be considered an expert, and will include the features described in Section 4.2.1 and Section 4.2.2 (features are calculated in respect to page $A(t)$ – the object of classification). Each expert is trained with the target function, h, of the *relevant page*. Intuitively, each expert can predict based on the changes it experienced, its content and similarity to page A at time t and previous times, whether page A will experience a change in the future. For example, if we wish to predict a change in an Information Retrieval professor's page, we might train a classifier based on the features of the WSDM page, where the label would be whether the professor's page changed or not.

While the combination of the pages decision can be done using any ensemble combination method we use weighted majority voting (see weighting techniques in Section 6). Formally, given prediction interval σ, for each object $o \in O$ train a classifier using examples (with the features described in Section 4.2.2) and labeled with the h value of the page A in σ time stamps, i.e., $h(o, t + \sigma)$. Those examples are used to train a classifier C (in our experiments we use SVM classifiers for this purpose). Eventually O classifiers are trained.

During prediction, for each classifier C_i (corresponding to an object o_i), the set of the above features is extracted for the test instance and given as input to C_i. We then output the combined prediction of the classifier by selecting the majority weighted vote. See full algorithm in Figure 3.

Procedure MULTIPLE EXPERTS($WeakLearner, O, o_{target}$)
Train:
Foreach $o_j \in O$
$E(o_j) = \{ \langle f_1^t(o_j, t_i), \ldots, f_n^t(o_j, t_i)\rangle, h(o_{target}, t_i + \sigma)|\}$
Call *WeakLearner* with $E(o_j)$,
and receive the hypothesis classifier h_j
Add h_j to the ensemble, ξ
Test: Given an unlabeled instance o_{target} at time t
Evaluate the ensemble $\xi = \{h_1, \ldots, h_{|O|}\}$ on $o_{target}(t)$
Let $V_{t,j} = \begin{cases} R_i(o_{target}, o_j) & \text{if } h_t \text{ picks class } c_j \\ 0 & \text{otherwise} \end{cases}$
be the vote given to class c_j by classifier h_t
Obtain total vote received by each class
$V_j = \sum_{t=1}^{T} v_{t,j}$ for $j \in \{0, 1\}$
Return the class that receives the highest total vote.

Figure 3: Temporal experts algorithm. The inputs are a weak learner, the pages O and the page to be classified o_{target}. R_i is a weighting of the relation between two objects. The prediction interval is σ.

5.3.2 Relational-Union Algorithm

In many relational learning scenarios relational features are created by collecting features over related objects, which are then fed into a classifier. This type of prediction has been used widely on the web [15]. One such example is the ad-click prediction problem [21]. Here the nodes are the ads, bids and queries, and the edges are the relations between the three. The goal is to predict a certain click on an ad. Similarly here, the nodes are the pages and the edges are the hyperlinks. The goal is to predict a certain page change.

Given M time stamps, we create training examples with the features described in Section 4.2.2. Given prediction interval σ, each example is labeled with the h value of the page of prediction in σ time stamps, i.e., $h(o, t+\sigma)$. Those E examples are used to train a classifier C. In our experiments we use SVM classifiers for this purpose. During prediction, the set of the above features is extracted for the test instance and given as input to C, which provides the prediction.

5.3.3 Relational-Stacking Algorithms

Stacking methods use multiple models to improve prediction over just a single model. Stacking algorithms, similar to the previous models, combine many features from related objects, and then train several "base" models either by bagging or reweighting the example distribution (boosting) before combining these models in the final model output.

Bagging [5] is one such method, where randomly sampled examples from the training set are used to train different base models. The base models predictions are then combined, where every model has an equal weight. Let k be the number of base models and let E be the set of examples as in Section 5.3.2. Given a training set of size $|E|$, this method generates k new training sets, each of size $|E|$, by sampling examples from the original training set uniformly and with replacement. A model is trained for each of the k training sets, and combined by majority voting. This method was shown to be useful in improving many linear models [5].

An extension of this idea, is **Boosting**. Intuitively, instead of randomly choosing instances, the ensemble model is built incrementally, where every new base model is trained with the training instances that previous models did not perform well on. Let E be the examples (as mentioned in Section 5.3.2), k number of base models, and let C be a classifier. AdaBoost [14] defines a vector of weights W for each example in E. It is first initialized as: $W_0(j) = \frac{1}{|E|}$. The algorithm performs k iterations, at each iteration reweighing the examples and retraining a classifier C_i with the weighted examples. The examples are reweighed in the following way:

$$W_{i+1}(j) = \frac{W_i(j) \exp(-\alpha_t y_j C_i(x_j))}{Z_i}$$

where Z_i is a normalization factor, $C_i(x)$ is the prediction of the classifier at iteration i for example x, and $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$ is the weighted error rate at iteration i, where $\epsilon_i = \sum_{j=1}^{|E|} W_i(j)[y_j \neq h_t(x_j)]$ is the error rate at iteration i. The output of the ensemble model is a combination of the k base models created at the end of each iteration weighted by the error α_i they experienced.

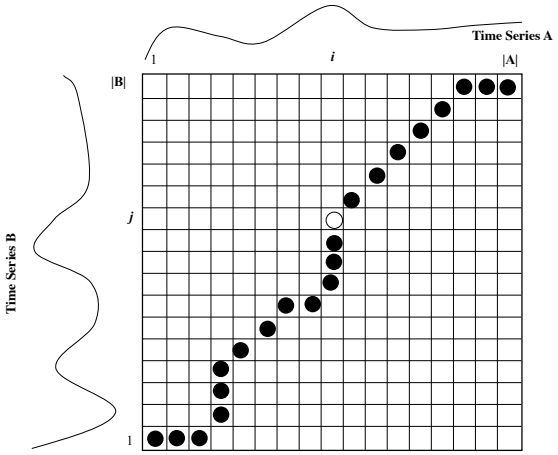

Figure 4: DTW finds the best alignment between time series A and B by maintaining the illustrated cost matrix $C(i, j)$.

6. EXPERT SELECTION

The number of pages on the web, and therefore the number of temporal experts (or number of 3D features as used by the relational algorithm), is extremely large. Due to this computational considerations, only a subset of the pages in O can be considered and monitored for every page. Based on our formalism from Section 4.1, we assumed a relation function $R_i : O \times O$ between the objects. We explore several such functions that choose the experts to monitor for every page – what we refer to as the *Expert Selection* problem.

Let p be the page whose h value is being predicted, and let e be a candidate expert. Given a predefined constant k of pages the system is capable of monitoring, the goal of an expert selection algorithm is to find the k experts such that the prediction accuracy of page h values is the highest. Notice that simple cross validation is a hard task due to its computation complexity (selecting all possible set of experts). We next present several heuristic algorithms for estimation of these experts.

6.1 Graph Distance KNN

We hypothesize that pages that are linked to one another in the web graph have higher probability of experiencing

similar changes. In this case, we identify the experts for a page by using in-links and out-links and finding the closest neighbor using a single step using an undirected representation of the links. In other words, the experts are selected using a K-nearest-neighbors procedure in the web graph, and we weight each page equally ($R_i(o, neighbor(o) = 1$). We used a web graph based on a sample of 22 million web pages (reached from good reputation sites seed) and 345 million edges, extracted from a web crawl performed in 2006 (for more information see [17]), at the same time the content crawling was performed.

6.2 Content Similarity KNN

We hypothesize that content similarity is a potential catalyst of mutual page changes. As the expert and the page discuss similar topics, a change in one might lead to a change in the other. Each page can be represented in some vector space with a predefined distance metric, and then a k nearest algorithm can be applied to find the most similar experts in content. In this work, we represent each page as a bag of words and the distance metric chosen is the cosine similarity (i.e., $R_i(o, neighbor(o) = cos(text(o), text(neighbor(o)))$).

6.3 Temporal Content Similarity KNN

The above mentioned approaches can be considered as finding k nearest neighbors in some static snapshot (either the web graph or web content). In this section, we also leverage the content change dynamics of the pages. The change patterns of the expert and the page over time are essential information to understand their mutual temporal behavior. Intuitively, a good predictive expert of change will have some correlation in content not just at a static time, but also will tend to trend in change together with the predicted page across a range of time – although one may lag/lead the other. Furthermore, the change correlation will be in similar topics. That is, the temporal correlation of the content-similarity of the change behavior of the two pages is high. We explore the monitoring of content similarity using cross-correlation and dynamic time warping (DTW). Similar temporal semantic modeling techniques have been used successfully in the past to improve semantic relatedness tasks [20]. Cross-correlation is a measure of similarity of two time series as a function of some time-lag applied to them, and defined as

$$\frac{\sum_i[(ts_1[i] - E(ts_1)) \cdot (ts_2[i-d] - E(ts_2))]}{N \cdot \sigma_{ts_1}\sigma_{ts_2}},$$

where $d = 1 \ldots N$ is a possible delay, N is the minimal time series length, σ is the time series variance, and E is the means of the corresponding series. The delay with the best correlation is chosen as the similarity. DTW (see Figure 5), on the other hand is a measure of similarity of two time series as a function of both time-lag and speed. For example, DTW will give high score to the similarity of page A and page B, even if A changes more than B, but they still have a similar change pattern, i.e., B is influenced by some of the changes in A. DTW is a dynamic programming algorithm, that tries to find an optimal match of the time series by stretching and compressing section of those time series. It holds a cost matrix C, where $C(i, j)$ is the minimum distance between two points of the time series i and j (see illustration in Figure 4). Cross-correlation can be thought of a simple case of DTW, where only diagonal steps are allowed in the matrix.

We define *temporal content similarity* over time as the weighted combination of the correlation over the words in the two documents. Let a document d be represented by a sequence of words $w_1, \ldots, w_m$. Let $t_1, \ldots, t_T$ be a sequence of consecutive discrete time points (e.g., days). We define the *dynamics* of a word w in a document d to be the time series of its frequency of appearance in d:

$$\text{Dyn}(w, d) = \langle \text{tfidf}(w, d(t_0)), \ldots, \text{tfidf}(w, d(t_T)) \rangle$$

where $d(t_i)$ us the snapshot of document d at time t_i.

Let Q be a correlation function between time series (such as DTW or cross-correlation). We define a distance metric between p and e as follows:

$$\text{Distance}(p, e) \quad = \quad \sum_{i=1}^{k} \omega(w_i)[Q(\text{Dyn}(w_i, p), \text{Dyn}(w_i, e))]$$

where $\omega(w_i)$ is a weighting function for every word, which can represent how dominant the word was in past changes, how many times it changed, etc. In our work we consider a simple version where the function represents the significance of the word for the document in the last snapshot of page: $\omega(w_i) = \text{tfidf}(w_i, p(T))$. As before, the closest k experts can now be selected using the distance metric Distance, and the weights are the values of their correlation.

Procedure DTW(ts_1, ts_2, C)
  $n \leftarrow Min(|ts_1|, |ts_2|)$
  $dtw(ts_1, ts_2) \leftarrow$ new $[|ts_1| \times |ts_2|]$
  For $i = \{1 \ldots n\}$
    $dtw(i, 1) \leftarrow dtw(i-1, 1) + C(i, 1)$
    $dtw(1, i) \leftarrow dtw(1, i-1) + C(1, i)$
  For $i = \{1 \ldots n\}$
    For $j = \{1 \ldots n\}$
      $dtw(i, j) = |ts_1(i) - ts_2(j)| +$
        $Min(dtw(i-1, j), dtw(i, j-1), dtw(i-1, j-1))$
  Return $dtw(n, n)$

Figure 5: Dynamic time warping algorithm (DTW)

7. EXPERIMENTAL SETUP

We implemented the 1D, 2D and 3D algorithms and conducted a variety of experiments to test their performance, compared to the state of the art in the prediction literature.

7.1 Dataset

We evaluate our method on a real-world data collection. The data consists of 54,816 pages crawled on an hourly basis for a 6 months period (2007-06-30 till 2007-12-31). We collected URLs from the logs of logged-in users of the Microsoft Live Search toolbar. The pages to crawl were not random, but rather driven by user visitation patterns. We are interested in pages whose change is going to affect the satisfaction of many users, and therefore we sampled pages from those visited by 612,000 people for a five week period. For more information see the work by Adar et al. [2]. In order to prevent trivial predictions, we filtered out pages that exhibit a constant behavior, e.g., pages that either change all the time or don't at all. Additionally, for pages below a change frequency threshold (whose change patterns appear to be random), one can recrawl them on an infrequent schedule paying an amortized cost for having a stale copy. Let $\text{change}(o, t) = 1$ if the page o changes at time t, and $\text{change}(o, t) = 0$ otherwise. We require $|\{t|\, \text{change}(o, t) \neq \text{change}(o, t+1)\}| \geq \alpha$, where in our experimentation $\alpha = 0.3$, i.e., that page changes behavior at least in 30% of the data.

1D features [11, 7] (Baseline)	2D features [3, 6] (Single Expert)	3D features (Multiple Experts)
54.84%	62.93%	**72.72%**

Table 1: Prediction accuracy average on a sliding window. Results statistically significant using a t-test ($p < 0.05$) when compared to Multiple Experts.

Relational Union	Relational Stacking		Multiple Experts
	Boosting	Bagging	
62.91%	63.44%	63.58%	**72.72%**

Table 2: Prediction accuracy using different relational algorithms. Results statistically significant using a t-test ($p < 0.05$) when compared to Multiple Experts.

7.2 Empirical Methodology

In order to perform correct statistical evaluations of the algorithms over sufficient data, the empirical evaluation over each dataset is performed as follows:

- In each prediction, for each algorithm, we use the previous n days as training examples (we experimented on several n values). We construct each example where the task is to predict what happens on the following day ($\sigma = 1$). In order to construct each example the two days previous are used for the dynamic change features ($l = 1$).

- We repeated the predictions $|O|$ (number of pages) times. Each time, a different page was used as the target of prediction.

For each training set of size n and the consecutive test day, all algorithms train on all but last time point in the training set, and classify o_i with the resulting classifier. The classifier always takes as input the last ($l = 1$) time points. We use the same SVM polynomial classifier (degree=3) to implement all of the learning approaches to control for variance. All parameters of the learning algorithm were calibrated on the training data. The accuracy of a prediction algorithm for a page is averaged over all sliding windows (i.e., over $t_n - (t_1 + \sigma)$ predictions).

8. RESULTS

In this section we discuss the different experiments we have conducted. We first start with the main result showing that the use of information found in other pages provides higher precision for page change prediction. We then proceed with different analysis experiments of our algorithm.

8.1 Main Result

The accuracy results of all algorithms are presented in Table 1, with $k = 20$ experts selected using the cross-correlation method, trained with training size of $n = 50$. These particular values, as well as the number of experts and selection of cross-correlation as the primary temporal selection method, were chosen over a small sample of data. We investigate sensitivity of performance of these methods to varying choices of the amount of history used (n) as well as the number of experts selected (k) in Sections 8.2. Each value in the table represents an average performance over all objects. The results provide evidence that the temporal experts outperform the single experts and baseline, and that the single experts outperform the baseline (both results are statistically significant, as measured by a paired t-test).

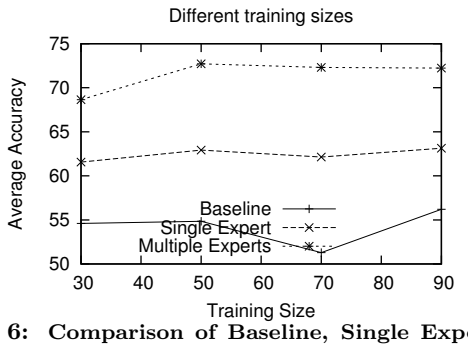

Figure 6: Comparison of Baseline, Single Expert and Multiple Experts over different training sizes. Each point in the graph represents the prediction accuracy (y label) of every of the method using the specified size of training examples (x label).

We further experiment with different training sizes (Figure 6). There was a statistical significant difference in all periods of times in favor of the multiple expert approach, and the single expert approach outperformed the baseline over all training sizes. The behavior of the algorithm exhibits almost constant behavior over the different training sizes, indicating that a crawler only monitoring the last 30 days of every page can perform with high accuracy. All learning algorithms performed poorly with less than 30 training examples, and therefore are not shown in the graph.

8.2 Parameter Selection Analysis

In this section we analyze the different multiple experts algorithms, the different expert selection method and how the number of experts affects performance.

8.2.1 Different Expert Algorithms

The accuracy results of all algorithms are presented in Table 2, with 20 experts selected using the cross-correlation method, trained with a training size of 50. We observe that the multiple experts algorithm we have suggested in this paper reaches the highest performance (significant as measured by a paired t-test), providing evidence that this type of mechanism is appropriate for combining information from multiple sources. We conclude that, although all algorithms were trained using the same set of features, just combining the features without the taking into account the structure, whether one trains a simple classifier, bags, or boosts, is not enough.

8.2.2 Expert Selection Methods

In this section we present the results for the different expert selection methods (see Table 3). We performed a paired t-test on the results, and found statistical significance difference between the cross-correlation approach and the non-temporal approaches with p-value $p < 0.05$. However, we found no statistical significance between the different temporal content approaches. All expert selection algorithms show statistically significantly higher performance than the performance of methods using 2D and 1D features.

The results provide an interesting viewpoint on the change behavior and its indicators. Although graph distance may seem an intuitive measure for the web, it provides little impact for change prediction. Correlation over time in the same topics is the more beneficial approach in those cases.

Graph Distance	Content Similarity	Temporal Content Similarity	
		Cross-Correlation	DTW
64.40%	67.27%	**72.72%**	70.33%

Table 3: Prediction accuracy average of multiple experts using the different selection algorithms. Results statistically significant using a t-test ($p < 0.05$) when compared to DTW.

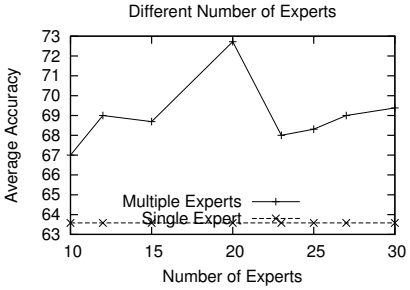

Figure 7: Number of experts as a function of accuracy. Each point in the graph represents the prediction accuracy (y label) of the multiple experts algorithm using the specified size of experts (x label).

8.2.3 *Number of Selected Experts*

In this section we investigate the performance of the multiple expert algorithm as a function of the number of selected experts. We show the results in Figure 7. While there is some sensitivity in performance to the number of experts selected, the gains over the single expert using the 2D features is robust across a range of values for number of experts. Up to 20 experts it seems the system gains more precision by the addition of more experts. However, after reaching this performance peak the addition of other features seems to be adding unnecessary noise to the system. Additionally, we see that even a small number of experts has enough predictive power for predicting page content change.

8.3 Application to Crawling: Maximizing Freshness

We show an application of our work to a crawling task in which the goal is to maximize the freshness of the indexed pages. It is standard practice to assume that the set of crawled pages is fixed and to assume a maximum crawl rate r [18]. We set a crawling limit of 10% of the total number of pages to simulate a setting where the number of pages to be crawled far exceeds the crawling resources, and adopt a scheduling policy that targets to re-visit pages as closely as possible to their change frequency by flipping a biased coin based on the estimated change probability. This probability is estimated by fitting logistic regression models (Platt recalibration [1]) to the outputs of the SVM classifier for the multiple and single experts, and for the baseline it is estimated based on the page frequency. We follow a batch crawling scenario, therefore the classifiers are not re-trained during the crawling period. Once a page is selected to be crawled it is used to update the training of all the methods. We compare the overall freshness of the index using different methods of estimating this probability as a function of the number of days the crawl was performed. Figure 8 shows the performance according to average freshness as a function of the number of observed crawling days (*i.e.*, the amount of data available to train the classifier). Both the 2D and 3D methods improve over the baseline with significant

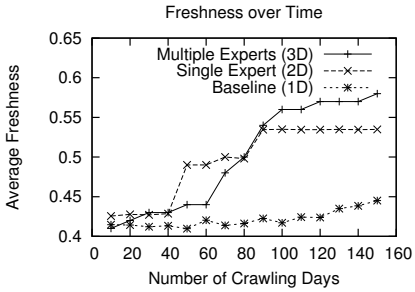

Figure 8: Freshness as a function of crawling days

gains in freshness occurring after observing 50 days of crawl data. The 3D method shows performance on par with the 2D method after 80 days with superior and significant performance after 100 days of crawl data. Overall, the results show the superiority of the 3D algorithm over the baseline after a small number of days were sampled and over the 2D approach after a moderate number of days, showing that the approach is also applicable for when some of the features are missing (as presented in the partially-observed setting).

9. DISCUSSION

In this section we give some qualitative examples to provide insight into the multiple experts and expert selection algorithms. We then also discuss how our methods can be efficiently calculated and scaled.

9.1 Qualitative Examples

In this section we provide some qualitative examples of the prediction results.

An example where related page expert selection outperforms the 1D and 2D approaches is for the URL `http://www.thekirkreport.com/archives.html`. This is a URL irregularly giving updates on stock trading. The content similarity KNN method (Section 6.2) chose the page of the American first credit union `http://www.amerfirst.org/`, that provides updates on rates, credits and loans. And indeed, in many of the times that the kirkreport had changes, it followed a significant update of all the rates on the amerfirst site. However, using simple content similarity pages as experts might also yield unsatisfactory result for change prediction. For example, the experts selected for the Southern Methodist university (`http://smu.edu/`, were other university sites such as the university of Windsor `http://www.uwindsor.ca`, and the university of Vermont `http://www.uvm.edu/`. Although all the above university pages change frequently, their change is usually due to updating university news, such as awards and local news. Therefore using those experts as indicators for the change in smu was unhelpful. As we use the page itself as an expert of its own change, in this case the ensemble algorithm weighted the smu page very high, yielding similar results to the 2D approach.

Those type of errors were easily fixed by the temporal content similarity method using cross-correlation (Section 6.3), that not only takes content into account, but also the actual correlation across time. For example, the best correlating pages for `http://www.boxofficeindia.com` were `http://www.musicnmovies.com` and `http://www.bollywoodworld.com`, as the pages experience change in the same topics when a new movie comes out to the theater.

The graph distance similarity (Section 6.1) usually se-

lects pages with no clear connection to the temporal dynamics; therefore the experts extracted using this method are not satisfactory. For example, for the URL `http://www.mtv.com/music/video` the experts selected were relevant pages such as a radio site `http://www.ksidradio.com`, but also irrelevant pages, such as a wrestling site `http://prwrestling.com`. For comparison, the cross-correlation experts for this same URL were `http://www.mtv.com/ontv/dyn`, `http://www.imdb.com/Sections/Quotes` and `http://www.newsoftheworld.co.uk/story_pages`. The first URL is an aggregate of new content in MTV, the second indicates a new popular movie that indicates new YouTube films about it, the last one is a correlation indicating that after specific news events there will be a new YouTube movie.

9.2 Efficient Calculation

In this work we showed that information from related pages strongly contributes to better page change detection. We investigated numerous ways of identifying such related pages – either by graph distance, content similarity or temporal content similarity. We have showed empirically that the temporal content similarity methods outperform the first two. However, identifying those related pages can be computationally expensive. Fortunately, several methods have been proposed in the literature, that can be used to speed up identifying such pages in an efficient way. For example, Keogh and Pazzani [16] discuss several ways of efficiently identifying similar time series in a large database. The method reduces the size of every time series by providing a method to divide it into frames, on which a time-series similarity algorithm can later be performed. The algorithm was applied using DTW algorithm, but the extension to the application of cross-correlation is similar.

Additional concern in such large scale prediction tasks is that storing snapshots from multiple iterations is prohibitively expensive. However, the cross-correlation method for finding temporal experts, that had the best performance in our empirical study, can be computed online without the need to actually store the previous snapshots. As more information arrives the correlation can be updated with only the correlation of the new snapshot. We saw in the empirical study that gains in performance asymptote after 30 days of historical data. Some classifiers can be trained online as well, preventing the need to keep the last snapshots. Additionally, in our experiments the object used for prediction is composed of only the past two snapshots ($l = 1$), and we did not see a significant higher gain when using bigger l values. This result is akin to the result by Ali & Williams [3], that showed that simply using only the previous two snapshots of a page to determine when to recrawl significantly reduces crawling compared to recrawling all recently changed pages.

10. CONCLUSIONS

In this paper we tackled the problem of predicting significant content change on the web. Our results shed light on how and why content changes on the web, and how it can be predicted. We explored several information sources that can be used to predict such change – the commonly used frequency of change, the page content and related pages content. We have seen that the addition of the page content improves prediction when compared to simple frequency-based prediction. Additionally, the addition of information of related pages content improves over the usage of page's content alone. We experimented with several methods for selecting related pages, and have shown that related pages that correlate over time in their content change contain valuable information for page content change prediction. We have investigated several ways of combining information from related pages, and proposed a novel algorithm that outperformed the common learning mechanisms.

11. REFERENCES

[1] B. S. A. J. Smola, P. Bartlett and D. Schuurmans. *Advances in Large Margin Classifiers*. MIT Press, 1999.

[2] E. Adar, J. Teevan, S. Dumais, and J. Elsas. The web changes everything: Understanding the dynamics of web content. In *Proc. of WSDM*, 2009.

[3] H. Ali and H. E. Williams. What's changed? measuring document change in web crawling for search engines. In *SPIRE*, pages 28–42, 2003.

[4] L. Barbosa, A. Salgado, F. de Carvalho, J. Robin, and J. Freire. Looking at both the present and the past to efficiently update replicas of web content. In *Proc. of WIDM Workshop*, 2005.

[5] L. Breiman. Bagging predictors. *ML*, 24(2):123–140, 1996.

[6] J. Cho and H. García-Molina. The evolution of the web and implications for an incremental crawler. In *Proc. of VLDB*, 2000.

[7] J. Cho and H. García-Molina. Effective page refresh policies for web crawlers. *TODS*, 28(4):390–426, 2003.

[8] J. Cho and H. García-Molina. Estimating frequency of change. *TOIT*, 3(3):256–290, 2003.

[9] J. Cho and A. Ntoulas. Effective change detection using sampling. In *Proc. of VLDB*, 2002.

[10] E. Coffman, Z. Liu, and R. Weber. Optimal robot scheduling for web search engines. *Journal of Scheduling*, 1(1), 1998.

[11] J. Edwards, K. McCurley, and J. Tomlin. An adaptive model for optimizing performance of an incremental web crawler. In *Proc. of WWW*, 2001.

[12] D. Fetterly, M. Manasse, M. Najork, and J. L. Wiener. A large-scale study of the evolution of web pages. In *Proc. of WWW*, 2003.

[13] D. Fetterly, M. Manasse, M. Najork, and J. L. Wiener. A large-scale study of the evolution of web pages. *Software Practice and Experience*, 1(1), 2003.

[14] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *JMLR*, 4:933–969, 2003.

[15] L. Getoor and L. Mihalkova. Exploiting statistical and relational information on the web and in social media. In *Proc. of WSDM*, 2011.

[16] E. Keogh and M. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proc. of KDD*, 2000.

[17] J. Leskovec, S. Dumais, and E. Horvitz. Web projections: learning from contextual subgraphs of the web. In *Proc. WWW*, 2007.

[18] C. Olston and M. Najork. Web crawling. *Foundations and Trends in Information Retrieval*, 4(3):175–246, 2010.

[19] S. Pandey and C. Olston. User-centric web crawling. In *Proc. of WWW*, pages 401–411, 2005.

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

[21] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *Proc. of WWW*, 2007.

[22] D. Shahaf, Y. Azar, E. Lubetzky, and E. Horvitz. Optimal web crawling. Technical Report, April 2011.

[23] Q. Tan and P. Mitra. Clustering-based incremental web crawling. *TOIS*, 28(4), 2010. Article 17.

[24] Q. Tan, Z. Zhuang, P. Mitra, and C. Giles. A clustering-based sampling approach for refreshing search engine's database. In *Proc. of WebDB Workshop*, 2007.

[25] J. Wolf, M. Squillante, P. Yu, J.Sethuraman, and L.Ozsen. Optimal crawling strategies for web search engines. In *Proc. of WWW*, 2002.