Modeling and Managing Changes in Text Databases

PANAGIOTIS G. IPEIROTIS
New York University
and
ALEXANDROS NTOULAS
Microsoft Search Labs
and
JUNGHOO CHO
University of California, Los Angeles
and
LUIS GRAVANO
Columbia University

Large amounts of (often valuable) information are stored in web-accessible text databases. "Metasearchers" provide unified interfaces to query multiple such databases at once. For efficiency, metasearchers rely on succinct statistical summaries of the database contents to select the best databases for each query. So far, database selection research has largely assumed that databases are static, so the associated statistical summaries do not evolve over time. However, databases are rarely static and the statistical summaries that describe their contents need to be updated periodically to reflect content changes. In this article, we first report the results of a study showing how the content summaries of 152 real web databases evolved over a period of 52 weeks. Then, we show how to use "survival analysis" techniques in general, and Cox's proportional hazards regression in particular, to model database changes over time and predict when we should update each content summary. Finally, we exploit our change model to devise update schedules that keep the summaries up to date by contacting databases only when needed, and then we evaluate the quality of our schedules experimentally over real web databases.

Categories and Subject Descriptors: H.2.4 [**Database Management**]: Systems—*Textual Da-*

Authors' address: Panagiotis G. Ipeirotis, Department of Information, Operations, and Management Sciences, New York University, 44 West Fourth Street, Suite 8-84, New York, NY 10012-1126; email: panos@stern.nyu.edu; Alexandros Ntoulas, Microsoft Search Labs, 1065 La Avenida, Mountain View, CA 94043; email: antoulas@microsoft.com; Junghoo Cho, Department of Computer Science, University of California, Los Angeles, 3532E Boelter Hall, Los Angeles, CA 90095-1596; email: cho@cs.ucla.edu; Luis Gravano, Computer Science Department, Columbia University, 1214 Amsterdam Avenue, New York, NY 10027-7003; email: gravano@cs.columbia.edu. This material is based upon work supported by the National Science Foundation under Grants No. IIS-97-33880, IIS-98-17434, IIS-0643846, IIS-0534784, and IIS-0347993. The work of Panagiotis G. Ipeirotis is also supported by a Microsoft Live Labs Search Award and a Microsoft Virtual Earth Award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or of the Microsoft Corporation.

tabases, Distributed Databases; H.2.5 [**Database Management**]: Systems—*Heterogeneous Databases*; H.3.1 [**Content Analysis and Indexing**]: Abstracting Methods, Indexing Methods; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering, Information Filtering, Search Process, Selection Process*; H.3.4 [**Systems and Software**]: Information Networks, Performance Evaluation (efficiency and effectiveness); H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based Services*; H.3.6 [**Information Storage and Retrieval**]: Library Automation —*Large Text Archives*; H.3.7 [**Information Storage and Retrieval**]: Digital Libraries

General Terms: Algorithms, Experimentation, Measurement, Performance

Additional Key Words and Phrases: Metasearching, Text Database Selection, Distributed Information Retrieval

1. INTRODUCTION

A substantial amount of textual information on the web is stored in databases. While some databases are "crawlable" a significant fraction is not indexed by search engines. One way to provide one-stop access to the information in text databases (crawlable or not) is through *metasearchers*, which can be used to query multiple databases simultaneously. The *database selection* step of the metasearching process, in which the best databases to search for a given query are identified, is critical for efficiency, since a metasearcher typically provides access to a large number of databases. The state-of-the-art database selection algorithms rely on aggregate statistics that characterize the database contents. These statistics, which are known as *content summaries* [Gravano et al. 1999] (or, alternatively, as *resource descriptions* [Callan 2000]), usually include the *frequency* of the words that appear in a database, plus perhaps other simple statistics such as the number of documents in the database. How to update these summaries, which provide sufficient information to decide which databases are the most promising for evaluating a given query, is the focus of this article.

So far, database selection research has largely assumed that databases are static. However, real-life databases are not always static and, accordingly, the statistical summaries that describe their contents need to be updated periodically to reflect database content changes. Defining schedules for updating the database content summaries is a challenging task, because the rate of change of the database contents might vary drastically from database to database. Furthermore, finding appropriate such schedules is important to keep content summaries up to date without overloading databases unnecessarily to regenerate summaries that are already (at least close to) up to date.

In this article, we start by presenting an extensive study on how the content of 152 real web databases evolved over a period of 52 weeks. Given that small changes in the databases might not necessarily be reflected in the (relatively coarse) content summaries, we examined how these summaries change over time. Specifically, we analyzed the evolution of "complete" content summaries, which can be derived when we have full access to the database contents (e.g., via "crawlers" [Chakrabarti 2002]), as well as the evolution of "approximate" content summaries, which are used when database access is limited (e.g., as is the case for "hidden web" data-

bases [Bergman 2001]). Our study shows that summaries indeed change and that old summaries eventually become obsolete, which then calls for a content summary update strategy.

In our approach for modeling content changes, we resort to the field of statistics named "survival analysis." Using the Cox proportional hazards regression model [Cox 1972], we show that database characteristics can be used to predict the pattern of change of the summaries. We then exploit our change models to develop summary update strategies that work well even under a resource-constrained environment. Our strategies attempt to contact the databases only when needed, thus minimizing the communication with the databases. Our experimental evaluation, over 152 real web databases, shows the merits of our update strategies. Our experiments include a comparison with a technique from the literature developed for a different but related problem, namely how to decide when to recrawl (and update a search engine index of) crawlable web sites. We also develop and evaluate a machine learning approach for updating content summaries. Overall, our experiments show that our survival analysis approach significantly outperforms all the alternatives that we considered.

In brief, the contributions of this article are as follows:

—In Section 3, we report the results of our extensive experimental study on how the content summaries of 152 real web databases evolved over a period of 52 weeks.

—In Section 4, we use survival analysis techniques to discover database properties that help predict the rate of change of database content summaries. Our analysis examines the evolution of both complete and approximate content summaries. We show how to devise a change model and schedule content summary updates accordingly. The resulting update strategies attempt to contact the databases only when strictly needed, thus avoiding wasting resources unnecessarily.

—In Section 5, we outline alternative approaches for updating content summaries. In particular, we use machine learning and cast the update problem as a binary classification task, with classification features suitably derived from the databases by leveraging the survival analysis framework.

—In Section 6, we present an extensive experimental evaluation that compares the proposed survival analysis approach with the Section 5 alternatives, which include our highly-optimized machine learning technique. The experimental results establish the superiority of our survival analysis approach.

Finally, Section 7 discusses related work, while Section 8 provides further discussion and concludes the article.

2. BACKGROUND ON CONTENT SUMMARY CONSTRUCTION

This section introduces the notation and necessary background for this article. We first define the notion of a *content summary* for a text database and briefly summarize how database selection algorithms exploit these summaries. Then, we review how to approximate database content summaries via querying.

Definition 2.1. The *content summary* $C(D)$ of a database D consists of:

D_1, with $\lvert D_1\rvert$=51,500	
w	$f(w, D_1)$
algorithm	7,210
cassini	5
saturn	2

D_2, with $\lvert D_2\rvert$=5,730	
w	$f(w, D_2)$
algorithm	2
cassini	3,260
saturn	3,730

Table I. A fragment of the content summaries of two databases.

—The actual number of documents in D, $|D|$, and

—For each word w, the number of D documents $f(w, D)$ that include w.

For efficiency, a metasearcher should evaluate a query only on a relatively small number of databases that are relevant to the query. The database selection component of a metasearcher typically makes the selection decisions using the information in the content summaries, as the following example illustrates:

EXAMPLE 2.2. *Consider the query* [cassini saturn] *and two databases D_1 and D_2. Based on the content summaries of these databases (Table I), a database selection algorithm may infer that D_2 is a promising database for the query, since each query word appears in many D_2 documents. In contrast, D_1 will probably be deemed not as relevant, since it contains only up to a handful of documents with each query word.*

Database selection algorithms work best when the content summaries are accurate and up to date. The most desirable scenario is when each database either (1) is crawlable, so that we can (periodically) download its contents and generate content summaries, or (2) exports these content summaries directly and reliably (e.g., using a protocol such as STARTS [Gravano et al. 1997]). Unfortunately, the so-called *hidden-web* databases [Gravano et al. 2003], which abound on the web, are not crawlable and only provide access to their documents via querying; furthermore, no protocol is widely adopted for web-accessible databases to export metadata about their contents. Hence, it is generally not possible to extract the complete content summary of a hidden-web database.

To characterize the contents of a hidden-web database, an interesting observation is that we can easily extract document samples from the database via querying. In turn, we can approximate the content summary of the database using the documents in a sample. In this article, we use the "hat" notation to refer to an *approximate, sample-based* content summary:

Definition 2.3. An *approximate, sample-based content summary* $\hat{C}(D)$ of a database D consists of:

—An estimate $|\hat{D}|$ of the number of documents in D, and

—For each word w, an estimate $\hat{f}(w, D)$ of $f(w, D)$.

The $\hat{C}(D)$ estimates are computed from a sample of the documents in D.

In this article, we study two state-of-the-art strategies for constructing approximate, sample-based content summaries:

—*Query-Based Sampling (QBS)*, as presented in [Callan and Connell 2001]: *QBS* starts by choosing words randomly from a dictionary and uses them to query a

given database until at least one document is retrieved. Then, *QBS* continues to query the database using words that are randomly chosen from the retrieved documents. Each query retrieves up to k previously unseen documents (we set $k = 4$ in our implementation following the suggestions by Callan and Connell [2001], who experimented with other values of k as well). Sampling stops after retrieving sufficiently many documents (we stop after retrieving 300 documents, again following [Callan and Connell 2001]). In our experiments, sampling also stops when 500 consecutive queries retrieve no new documents. (Getting no new results for 500 random queries is a signal that *QBS* might have retrieved the majority of the documents in the database.)

—*Focused Probing (FPS)*, as presented in [Ipeirotis and Gravano 2002]: Instead of sending (pseudo-) randomly picked words as queries, *FPS* derives queries from a classifier learned over a topic hierarchy. Thus, queries are associated with specific topics. For example, a query *[breast cancer]* is associated with the category "Health." We retrieve the top-k previously unseen documents for each query (we set $k = 4$ in our implementation, following the suggestions in [Ipeirotis and Gravano 2002]) and, at the same time, keep track of the number of matches generated by each query. When the queries related to a category (e.g., "Health") generate a large number of matches, probing continues for its subcategories (e.g., "Diseases" and "Fitness"). The output of the algorithm is both an approximate content summary and the classification of the database in a hierarchical classification scheme. In our experiments, the queries are derived from an SVM classifier following the techniques described in [Gravano et al. 2003], over the 72-node hierarchy also used in [Ipeirotis and Gravano 2002; Gravano et al. 2003].

In addition to the *QBS* and *FPS* approximate content summaries, we also study the evolution of the complete database content summaries (Definition 2.1), to which we will refer as *complete* (*CMPL*). To derive the complete content summary of a database, we retrieve all the documents from the database and compute the document frequency of each word. This technique requires that each database either allows direct access to its documents or supports the functionality of a protocol such as STARTS [Gravano et al. 1997].

Next, we present the results of our study, which examined how *CMPL*, *QBS*, and *FPS* content summaries of 152 text databases changed over a period of 52 weeks.

3. STUDYING CONTENT CHANGES OF REAL TEXT DATABASES

One of the goals of this article is to study how text database changes are reflected over time in the database content summaries. First, we discuss our data set in detail (Section 3.1). Then, we report our study of the effect of database changes on the content summaries (Section 3.2). The conclusions of this study will be critical for Section 4, when we discuss how to model content summary change patterns.

3.1 Data for our Study

Our study and experiments involved 152 searchable databases, whose contents were downloaded weekly from October 2002 through October 2003. These databases have previously been used in a study of the evolution of web pages [Ntoulas et al. 2004].

Domain	com	edu	gov	misc	org
%	47.3%	13.1%	17.1%	6.8%	15.7%

Table II. Domain distribution in our data set.

The databases in our study were —roughly— the five top-ranked web sites in a subset of the topical categories of the Google Directory, using the same topical categories as in [Gravano et al. 2003]. Google Directory, in turn, reuses the hierarchical classification of web sites from the Open Directory Project. (Please refer to [Ntoulas et al. 2004] for more details on the rationale behind the choice of these web sites.) From these web sites, we picked only those sites that provided a search interface over their contents, which are needed to generate sample-based content summaries. Also, since we wanted to study content changes, we only selected databases with crawlable content, so that every week we can retrieve the full database contents using a crawler. A complete list of the sites included in our experiments is available at `http://webarchive.cs.ucla.edu/`. Table II shows the breakdown of web sites in the set by high-level DNS domain, where the *misc* category represents a variety of relatively small domains (e.g., *mil*, *uk*, *dk*, and *jp*). Similarly, Table III shows the breakdown of web sites by topical category, as assigned by the Google Directory. In this case, the *misc* category represents various small topical categories (e.g., world, shopping, and games).

We downloaded the contents of the 152 web sites every week for a period of one year. For each web site, we started our crawl from the root web page and continued to download pages —in breadth-first order— until either we exhausted all pages within the site or we downloaded 200,000 pages from the site.[1] By following all the links recursively starting from the root page of each site we believe that we captured a relatively complete version of the contents of each site.[2] Each weekly snapshot consisted of three to five million pages, or around 65 GB before compression, for a total over one year of almost 3.3 TB of history data. We treated each web site as a database, and created —each week— the complete content summary $C(D)$ of each database D by downloading and processing all of its documents. This data allowed us to study how the *complete* content summaries of the databases evolved over time. In addition, we also studied the evolution over time of an *approximate* content summary $\hat{C}(D)$ of each database D, computed weekly[3] using either *QBS* or *FPS*. To reduce the effect of sampling randomness in our *QBS* experiments, we create five *QBS* content summaries of each database each week, in turn derived from five document samples, and report the various metrics in our study as averages over these five summaries.

[1]Only four web sites were affected by this efficiency-motivated page-download limitation: `hti.umich.edu`, `eonline.com`, `pbs.org`, and `intelihealth.com`.

[2]We are not aware of any site in our data set containing pages that are not reachable from the root page of the site.

[3]To compute the approximate content summaries, we indexed and queried the data using *ht://Dig* (`http://www.htdig.org/`), an off-the-shelf indexing package.

Category	%	Category	%
computers	22.5%	reference	7.3%
science	17.2%	sports	5.3%
health	9.9%	news	4.0%
arts	8.6%	business	4.0%
regional	7.9%	recreation	2.0%
society	7.3%	misc	4.0%

Table III. Category distribution in our data set.

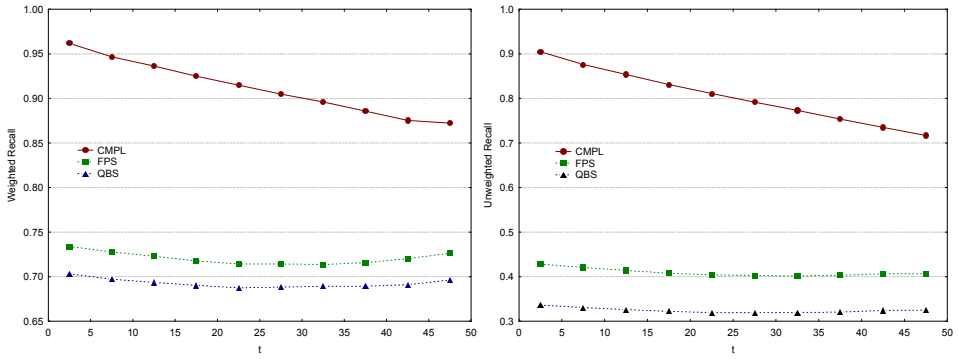
3.2 Measuring Content Summary Change

We now turn to measuring how the database content summaries —both the complete and approximate versions— evolve over time. For this, we resort to a number of metrics of content summary similarity and quality from the literature. We discuss these metrics and the results for the 152 web databases next.

For our discussion, we refer to the "current" and complete content summary of a database D as $C(D)$, while $O(D,t)$ is the complete summary of D as of t weeks into the past. The $O(D,t)$ summary can be considered as an (old) approximation of the (current) $C(D)$ summary, simulating the realistic scenario where we extract a summary for a database D and keep it unchanged for t weeks. In the following definitions, W_o is the set of words that appear in $O(D,t)$, while W_c is the set of words that appear in $C(D)$. Values $f_o(w,D)$ and $f_c(w,D)$ denote the document frequency of word w in $O(D,t)$ and $C(D)$, respectively.

3.2.1 *Recall.* An important property of the content summary of a database is its coverage of the current database vocabulary. An up-to-date and complete content summary always has perfect recall, but an old summary might not, since it might not include, for example, words that appear only in new database documents. The *unweighted recall (ur)* of $O(D,t)$ with respect to $C(D)$ is the fraction of words in the current summary that are also present in the old summary: $ur = \frac{|W_o \cap W_c|}{|W_c|}$. This metric gives equal weight to all words and takes values from 0 to 1, with a value of 1 meaning that the old content summary contains all the words that appear in the current content summary, and a value of 0 denoting no overlap between the summaries. An alternative recall metric, which gives higher weight to more frequent terms, is the *weighted recall (wr)* of $O(D,t)$ with respect to $C(D)$: $wr = \frac{\sum_{w \in W_o \cap W_c} f_c(w,D)}{\sum_{w \in W_c} f_c(w,D)}$. We will use analogous definitions of unweighted and weighted recall for a sample-based content summary $\hat{O}(D,t)$ of database D obtained t weeks into the past with respect to the current content summary $C(D)$ for the same database.

The *CMPL* lines in Figures 1(a) and 1(b) show the weighted and unweighted recall, respectively, for complete t-week-old summaries with respect to the "current" summary, as a function of t and averaged over every possible choice of "current" summary. Predictably, both the weighted and unweighted recall values decrease as t increases. For example, on average, 1-week-old summaries have unweighted recall of 91%, while older, 25-week-old summaries have unweighted recall of about 80%. The weighted recall figures are higher, as expected, but still significantly less than 1: this indicates that the newly introduced words have low frequencies, but



(a) Weighted Recall (b) Unweighted Recall

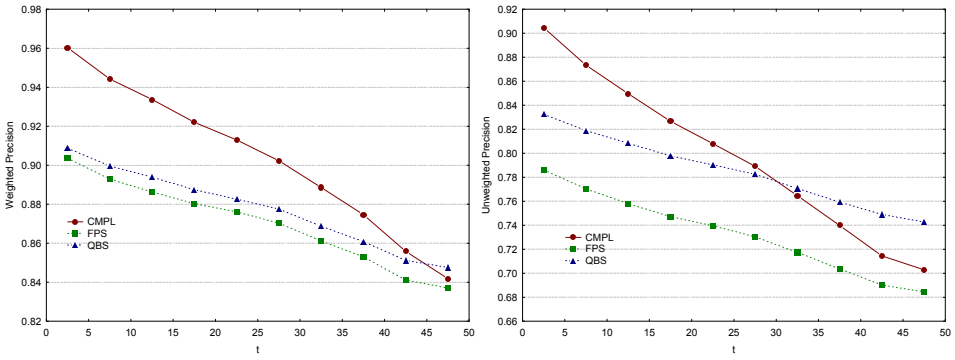
Fig. 1. The weighted and unweighted recall of content summary $O(D,t)$ (*CMPL*), and of the approximate content summaries $\tilde{O}(D,t)$ (*QBS* and *FPS*), with respect to the "current" content summary $C(D)$, as a function of time t and averaged over each database D in the data set.

constitute a substantial fraction of the database vocabulary as well.

The *QBS* and *FPS* lines in Figure 1 show the corresponding results for *QBS* and *FPS* content summaries. As expected, the values for all the approximate, sample-based summaries are substantially smaller than those for the complete summaries. Also, the recall values of the sample-based summaries do not change much over time, because the sample-based summaries are only marginally complete to start with and do not suffer a significant drop in recall over time. This shows that the inherent incompleteness of the sample-based summaries "prevails" over the incompleteness introduced by time.

Another interesting observation is that recall figures initially decrease (slightly) for approximately 20 weeks, then remain stable, and then, surprisingly, increase, so that a 50-week old content summary has higher recall than a 20-week old one, for example. This unexpected result is due to an interesting periodicity: some events (e.g., "Christmas," "Halloween") appear at the same time every year, allowing summaries that are close to being one year old to have higher recall than their younger counterparts. This effect is only visible in the sample-based summaries, which cover only a small fraction of the database vocabulary, and is not observed in the complete summaries, mainly because they are larger and are not substantially affected by a relatively small number of words.

3.2.2 *Precision.* Another important property of the content summary of a database is the precision of the summary vocabulary. Up-to-date content summaries contain only words that appear in the database, while older summaries might include obsolete words that appeared only in deleted documents. The *unweighted precision (up)* of $O(D,t)$ with respect to $C(D)$ is the fraction of words in the old content summary that still appear in the current summary $C(D)$: $up = \frac{|W_o \cap W_c|}{|W_o|}$. This metric, like *unweighted recall*, gives equal weight to all words and takes values from 0 to 1, with a value of 1 meaning that the old content summary only contains words that are still in the current content summary, and a value of 0 denoting no overlap between the summaries. The alternative precision metric, which —just as in the *weighted recall* metric— gives higher weight to more frequent terms, is the



(a) Weighted Precision (b) Unweighted Precision

Fig. 2. The weighted and unweighted precision of content summary $O(D,t)$ (*CMPL*), and of the approximate content summaries $\hat{O}(D,t)$ (*QBS* and *FPS*), with respect to the "current" content summary $C(D)$, as a function of time t and averaged over each database D in the data set.

weighted precision (wp) of $O(D,t)$ with respect to $C(D)$: $wp = \frac{\sum_{w \in W_o \cap W_c} f_o(w,D)}{\sum_{w \in W_o} f_o(w,D)}$. We use analogous definitions of unweighted and weighted precision for a sample-based content summary $\hat{O}(D,t)$ of a database D with respect to the correct content summary $C(D)$.

The *CMPL* lines in Figures 2(a) and 2(b) show the weighted and unweighted precision, respectively, for complete t-week-old summaries with respect to the "current" summary, as a function of t and averaged over every possible choice of "current" summary. Predictably, both the weighted and unweighted precision values decrease as t increases. For example, on average, a 48-week-old summary has unweighted precision of 70%, showing that 30% of the words in the old content summary do not appear in the database anymore.

The *QBS* and *FPS* lines in Figure 2 show the corresponding results for *QBS* and *FPS* content summaries. As expected, precision decreases over time, and decreases much faster than recall. For example, almost 20% of the words in a 15-week-old *QBS* content summary are absent from the database. The periodicity that appeared in the recall figures is not visible for the precision results: the sample-based content summaries contain many more "obsolete" words that do not appear in the database anymore, so a small number of words that appear periodically cannot improve the results.

3.2.3 *Kullback-Leibler Divergence.* Precision and recall measure the accuracy and completeness of the content summaries, based *only* on the presence of words in the summaries. However, these metrics do not capture the accuracy of the frequency of each word as reported in the content summary. For this, the *Kullback-Leibler divergence* [Jelinek 1999] of $O(D,t)$ with respect to $C(D)$ (KL for short) calculates the "similarity" of the word frequencies in the old content summary $O(D,t)$ against the "current" word frequencies in $C(D)$: $KL = \sum_{w \in W_o \cap W_c} p_c(w|D) \cdot \log \frac{p_c(w|D)}{p_o(w|D)}$, where $p_c(w|D) = \frac{f_c(w,D)}{\sum_{w' \in W_o \cap W_c} f_c(w',D)}$ is the probability of observing w in $C(D)$, and $p_o(w|D) = \frac{f_o(w,D)}{\sum_{w' \in W_o \cap W_c} f_o(w',D)}$ is the probability of observing w in $O(D,t)$. The KL divergence metric takes values from 0 to infinity, with 0 indicating that the two

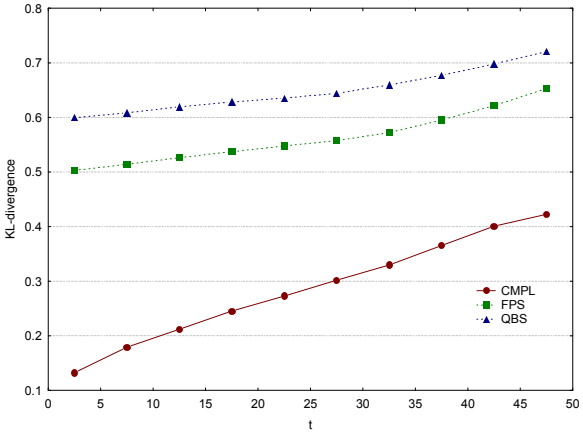


Fig. 3. The KL divergence of the content summary $O(D,t)$ (*CMPL*), and of the approximate content summaries $\hat{O}(D,t)$ (*QBS* and *FPS*), with respect to the "current" content summary $C(D)$, as a function of time t and averaged over each database D in the data set.

content summaries being compared are equal.

The *CMPL* line in Figure 3 shows that the KL divergence of old content summaries increases as t increases. This confirms the previously observed results and shows that the word frequency distribution changes substantially over time. Furthermore, the KL divergence of the old approximate summaries (lines *QBS* and *FPS*) also increases with time, indicating that approximate content summaries become obsolete just as their complete counterparts do.

3.2.4 *Conclusion.* We studied how content summaries of text databases evolve over time. We observed that the quality of content summaries (both complete and sample-based) deteriorates as they become increasingly older. Therefore, it is imperative to have a policy for periodically updating the summaries to reflect the current contents of the databases. We now turn to this important issue and show how we can use "survival analysis" for this purpose.

4. PREDICTING CONTENT SUMMARY CHANGE FREQUENCY

In the previous section, we established the need for updating database content summaries as the underlying text databases change. Unfortunately, updating a content summary involves a non-trivial overhead: as discussed, the content summaries of hidden-web text databases are constructed by querying the databases, while the summaries of crawlable databases are constructed by downloading and processing all the database documents. Therefore, in order to avoid overloading the databases unnecessarily, it is important to schedule updates carefully. In this section, we present our "survival analysis" modeling approach for deciding *when* to update content summaries. First, Sections 4.1 and 4.2 review the necessary background on survival analysis and the Cox regression model from the literature [Cox 1972]. Then, Section 4.3 shows how we can use this material for our own scenario, to model content summary changes, and Section 4.4 shows how to use the modeling

results for scheduling content summary updates.

4.1 Survival Analysis

Survival analysis is a collection of statistical techniques that help predict the time until an event occurs [Marques De Sá 2003]. These methods were initially used to predict the time of survival for patients under different treatments, hence the name "survival analysis." For the same reason, the "time until an event occurs" is also called *survival time*. For our purposes, the survival time of a database D is the minimum number of weeks t such that $O(D, t)$ becomes "sufficiently different" from the current content summary $C(D)$ for the database. (We formally define the survival time of a database in Section 4.3.)

Survival times can be modeled through a *survival function* $S(t)$ that captures the probability that the survival time of an object is greater than or equal to t. In the survival analysis literature, the distribution of $S(t)$ is also described in terms of a *hazard function* $h(t)$, which is the "rate of failure" at time t, conditional on survival until time t: $h(t) = -\frac{\frac{dS(t)}{dt}}{S(t)}$ and, equivalently, $S(t) = \exp\left(-\int_0^t h(z)dz\right)$. A common modeling choice for $S(t)$ is the *exponential distribution*, where $S(t) = e^{-\lambda t}$, and so the hazard function is constant over time ($h(t) = \lambda$). A generalization of the exponential distribution is the *Weibull distribution*, where $S(t) = e^{-\lambda t^\gamma}$, and so the hazard function varies over time ($h(t) = \lambda\gamma t^{\gamma-1}$). (The exponential distribution corresponds to the case where $\gamma = 1$.) Recent findings indicate that the exponential function is a good model to describe changes in web *documents* [Brewington and Cybenko 2000a; Cho and García-Molina 2003]. While these findings suggest using the exponential distribution to model the survival time of a database, we will see in Section 4.3 that the exponential distribution does not accurately describe changes for summaries of web *databases*. So, instead, we will use the Weibull distribution.

As described so far, the survival function $S(t)$ and the hazard function $h(t)$ are used to describe a single database, and are not "instantiated" since we do not know the values of the configuring parameters. Of course, it is important to estimate the parameters of the survival function $S(t)$ for each database, to have a concrete, database-specific change model. Even more imperative is to discover *predictor variables* that can influence the survival times. For example, when analyzing the survival times of patients with heart disease, the weight of a patient is a predictor variable and can influence the survival time of the patient. Analogously, we want to predict survival times individually for each database, according to its characteristics. Next, we describe the Cox proportional hazards regression model that we use for this purpose.

4.2 Cox Proportional Hazards Regression Model

The *Cox proportional hazards regression model* [Cox 1972] is widely used in statistics for discovering important variables that influence survival times. This model is non-parametric, because it makes no assumptions about the nature or shape of the hazard function. The only assumption is that the logarithm of the underlying

hazard rate is a linear[4] function of the predictor variables.

Let x be a predictor variable, and x_a and x_b be the values of that variable for two databases D_a and D_b, respectively. Under the Cox model, the hazard functions $h_a(t)$ and $h_b(t)$ can be expressed for databases D_a and D_b as:

$$h_a(t) = e^{\beta x_a} h_0(t) \Rightarrow \ln h_a(t) = \ln h_0(t) + \beta x_a \tag{1a}$$

$$h_b(t) = e^{\beta x_b} h_0(t) \Rightarrow \ln h_b(t) = \ln h_0(t) + \beta x_b \tag{1b}$$

where $h_0(t)$ is a *baseline hazard function*, common for all the members of the population, and β is the model coefficient. We can generalize the Cox model for n predictor variables: in this case $\ln h(t) = \ln h_0(t) + \sum_{i=1}^{n} \beta_i x_i$, where the x_i's are the predictor variables, and the β_i's are the model coefficients. Then, the survival function for a database has the form:

$$S(t) = \exp\left(-\exp\left(\sum_{i=1}^{n} \beta_i x_i\right) \int_0^t h_0(z)dz\right) \Rightarrow \ln S(t) = \exp\left(\sum_{i=1}^{n} \beta_i x_i\right) \ln S_0(t) \tag{2}$$

where $S_0(t) = \exp\left(-\int_0^t h_0(z)dz\right)$ is the *baseline survival function*, common for all the members of the population. The algorithm presented by Cox [1972] shows how to compute the β_i values.

The Cox model, as presented so far, seems to solve the same problem addressed by multiple regression. However, the dependent variable (survival time) in our case is not normally distributed, but usually follows the exponential or the Weibull distribution —a serious violation for ordinary multiple regression. Another important distinction is the fact that the Cox model effectively exploits incomplete or "censored" survival times, from cases that "survived" the whole study period. Excluding these cases from a study would introduce a strong bias in the resulting model. Those observations are called *censored* observations and contain only partial information, indicating that *there was no failure during the time of observation.* The Cox model effectively uses the information provided from censored cases. (For more information, see [Cox 1972].)

The Cox proportional hazards model is one of the most general models for working with survival times, since it does not assume any specific baseline hazard function. This model allows the extraction of a "normalized" hazard function $h_0(t)$ that is not influenced by predictor variables. This allows for easier generalization of the results, since $h_0(t)$ is not dependent on the distribution of the predictor variables in the data set used to extract $h_0(t)$. The only requirement for the applicability of Cox's model is that the predictor variables follow the "proportional hazard" assumption, which means that, for two individual databases D_a and D_b, the hazard ratio $\frac{h_a(t)}{h_b(t)}$ is constant over time.

[4]The "linearity" or "proportionality" requirement is essentially a "monotonicity" requirement (e.g., the higher the weight of a patient, the higher the risk of heart attack). If a variable monotonically affects the hazard rate, then an appropriate transformation (e.g., $\log(\cdot)$) can make its effect linear.

An interesting variation of the Cox model that overcomes the proportional hazard assumption is the *stratified Cox model* [Stablein et al. 1981], which is used to account for variables that do not satisfy the proportionality assumption. In this case, the variables that do not satisfy this assumption are used to split the data set into different "strata." The β_i Cox coefficients remain the same across the different strata, but each stratum now has a different baseline function $h_0(t)$.

Next, we describe how we use the Cox regression model to represent changes in text database content summaries.

4.3 Using Cox Regression to Model Content Summary Changes

Before using any survival analysis technique for our problem, we need to define "change." A straightforward definition is that two content summaries $C(D)$ and $O(D,t)$ are "different" when they are not identical. However, even a small change in a single document in a database will probably result in a change in the content summary of the database, but such change is unlikely to be of importance for database selection. Therefore, we relax this definition and say that two content summaries are different when $KL > \tau$ (see Section 3.2 for the definition of KL divergence), where τ is a "change sensitivity" threshold.[5] Higher values of τ result in longer survival times and the exact value of τ should be selected based on the characteristics of the database selection algorithm of choice. We will see how we can effectively use the Cox model to incorporate τ in our change model. Later, in Section 4.4, we show that we can define update schedules that adapt to the chosen value of τ.

Definition 4.1. Given a value of the change sensitivity threshold $\tau > 0$, the *survival time of a database D at a point in time* —with associated "current" content summary $C(D)$— is the smallest time t for which the KL divergence of $O(D,t)$ with respect to $C(D)$ is greater than τ.

Note that we define the survival time of a database with respect to its *complete* content summary. An alternative that we do not explore in this article is to define survival time over *approximate* content summaries whenever we use *QBS* and *FPS* to construct the summaries; note, however, that this alternate definition would be problematic for *QBS*, where randomization can cause successively computed (approximate) content summaries to differ even if the underlying database has remained unchanged.

Now that we have a definition of the survival time for a content summary, we describe our survival analysis approach in detail. Our approach consists of the following steps:

(1) Compute the survival times for all the databases and all points in time in our data set (Section 4.3.1).
(2) Select the *useful* database features (across a variety of *candidate* features) for predicting the survival time of the database content summaries (Section 4.3.2).

[5]We use KL divergence for our change definition (as opposed to precision or recall) because KL depends on the similarity of word-frequency distributions. As our later experiments show, an update policy derived from the KL-based change definition improves not only the KL divergence but also precision and recall.

(3) Use the survival times from Step 1 and the useful features from Step 2 to run a Cox regression analysis, with the survival times as dependent variables and the useful database features as independent variables (Section 4.3.3).

We now describe each of these steps in detail, and finally discuss our modeling conclusions (Section 4.3.4).

4.3.1 *Computing Survival Times.* Using the study of Section 3 as well as Definition 4.1, we computed the survival time of each content summary for different values of the change sensitivity threshold τ. For some databases, we did not detect a change within the period of the study. As explained in Section 4.2, these "*censored*" cases are still useful since they provide evidence that the content summary of a database with the given characteristics *did not change* within the allotted time period and for the change sensitivity threshold τ of choice. The result of our study is a set of survival times, some marked as censored, that we use as input to the Cox regression model.

4.3.2 *Feature Selection.* After extracting the survival times, we select the database features that we pass as parameters to the Cox model. We use two sets of features: a set of "*current*" features and a set of "*evolution*" features. The *current* features are characteristics of the database at a given point in time. For example, the topic of the database (e.g., "*health*") and its DNS domain (e.g., "*.gov*") are *current* features of a database. On the other hand, we extract the *evolution* features by observing how the database changes over a (training) time period.

In our study, the initial set of *current* features that we used was:

—The change sensitivity threshold τ.

—The topic of each database, defined as the top level category under which the database is classified in the Open Directory. This is a categorical variable with 16 distinct values (e.g., "Arts," "Sports," and so on). We encoded this variable as a set of 16 dummy binary variables: each variable has the value 1 if the database is classified under the corresponding category, and 0 otherwise.

—The domain of the database, which is a categorical variable with five distinct values (com, org, edu, gov, misc). We encoded this variable as a set of 5 binary variables.

—The logarithm of the size of the database. For hidden-web databases that offer only query-based access to their contents, we estimate the size of the database using the "sample-resample" method from [Si and Callan 2003].

—The number of words in $C(D)$, for crawlable databases, or in $\hat{C}(D)$, for hidden-web databases, computed over the "current" document sample.

To extract the set of *evolution* features, we retrieved content summaries from each database every week over a period of 10 weeks. Then, for each database we compared every pair of summaries that were extracted exactly k weeks apart (i.e., on weeks t and $t + k$) using the precision, recall, and KL divergence metrics. Specifically, the features that we computed were:

—The average KL divergence $\kappa_1, \ldots, \kappa_9$ between summaries extracted with time difference of $1, \ldots, 9$ weeks.

—The average weighted and unweighted precision of summaries extracted with time difference of $1, \ldots, 9$ weeks.

—The average weighted and unweighted recall of summaries extracted with time difference of $1, \ldots, 9$ weeks.

After selecting the initial set of features, we trained the Cox model using the variables indicated above. We validated the results using leave-one-out cross validation.[6] The results of the initial run indicated that, from the *current* features, the number of words and the topic of a database are not good predictor variables. From the *evolution* features, the KL features are uniformly good predictors and are strongly and positively correlated with each other. The predictive value of precision and recall depends on the summary construction technique: precision and recall are not good predictor variables for *QBS* but they are for *FPS* and *CMPL*. This result is not surprising: *FPS* usually[7] issues the same queries each time that it samples a particular database. If the database has not changed, then the returned documents are the same and precision and recall are high. If the database has changed, then the returned set of documents is different, resulting in low precision and recall. Similarly, *CMPL* is sensitive to any changes in the underlying database. This property does not hold for *QBS*: *QBS* issues a potentially different set of queries each time that it samples a particular database, so the documents in a newly extracted sample may be completely different from those in earlier samples, even if the database has not changed. Therefore, precision and recall are not good predictor variables under *QBS*.

Given these results, we decided to drop the number of words and the topic variables from the *current* set, keeping only the change sensitivity threshold τ, the database size, and the domain. From the *evolution* set, we dropped recall and precision. Despite the fact that recall and precision are good predictor variables for *FPS* and *CMPL*, their importance in the presence of the KL features is negligible. From the KL features, we kept only κ_1: given its presence, features κ_2 through κ_9 were largely redundant. Since we only needed the κ_1 feature, we reduced the training time from 10 to two weeks. To examine whether any of the selected features —other than threshold τ, which we always keep, and domain, which we treat differently, as explained below— is redundant, we trained Cox using (a) size and τ; (b) κ_1 and τ; and (c) κ_1, size, and τ. We describe our findings next.

4.3.3 *Training the Cox Model.* After the initial feature selection, we trained the Cox model again. The results confirmed that all the features that we had selected are good predictor variables[8] and strongly influence the survival time of the extracted summaries. However, the domain variable did not satisfy the proportionality assumption, which is required by the Cox model (see Section 4.2): the hazard ratio between two domains was not constant over time. Hence, we resorted to the

[6]Since each database generates multiple survival times, we leave out one *database* at a time for the cross-validation.

[7]The queries for a database remain the same if the database classification does not change.

[8]For all models, the statistical significance is at the 0.001% level according to the Wald statistic [Marques De Sá 2003].

Method	Features	β_s	β_κ	β_τ
QBS	κ_1, size, τ	0.094	6.762	-1.305
	size, τ	0.179	-	-1.313
	κ_1, τ	-	8.3	-1.308
FPS	κ_1, size, τ	0.135	10.143	-1.329
	size, τ	0.179	-	-1.313
	κ_1, τ	-	14.765	-1.24
CMPL	κ_1, size, τ	0.155	2.849	-1.3712
	size, τ	0.178	-	-1.302
	κ_1, τ	-	3.198	-1.225

Table IV. The coefficients of the Cox model, when trained for various sets of features and for different content summary construction methods.

stratified Cox model, stratifying on domain.[9]

The result of the training was a set of coefficients β_s, β_κ, and β_τ for features size, κ_1, and τ, respectively. We show the Cox coefficients that we obtained in Table IV. Note that the *FPS* and *QBS* results for features size and τ are equivalent, as both estimate the database size using the sample-resample method [Si and Callan 2003]. In contrast, *CMPL* knows the actual database size. The results for *FPS* and *QBS* are slightly different from those for *CMPL* in this case, but the difference is not statistically significant. Overall, the positive values of β_s and β_κ indicate that larger databases are more likely to change than smaller ones and that databases that changed during training are more likely to change in the future than those that did not change. In contrast, the negative value for β_τ shows that —not surprisingly— higher values of τ result in longer survival times for content summaries.

Given the results of the analysis, for two databases D_a and D_b from the same domain, from Equation 2 we have:

$$\begin{aligned} \ln S_a(t) &= \exp(\beta_s \ln(|D_a|) + \beta_\kappa \kappa_{1a} + \beta_\tau \tau_a) \cdot \ln S_0(t) \\ \ln S_b(t) &= \exp(\beta_s \ln(|D_b|) + \beta_\kappa \kappa_{1b} + \beta_\tau \tau_b) \cdot \ln S_0(t) \end{aligned}$$

where $S_0(t)$ is the baseline survival function for the respective domain. The baseline survival function, by definition, corresponds to a "baseline" database D with size $|D| = 1$ (i.e., $\ln(|D|) = 0$), $\kappa_1 = 0$, and $\tau = 0$.

Under the Cox model, the returned baseline survival functions are defined only by a set of values $S_0(1), \ldots, S_0(n)$, which correspond to the probability of survival of the baseline database for the weeks $1, \ldots, n$. In our experiments, we had five baseline survival functions, one for each domain (i.e., com, edu, org, gov, misc). To fit the baseline survival functions, we assumed[10] that they follow the Weibull distribution (see Section 4.1), which has the general form $S(t) = e^{-\lambda t^\gamma}$. We applied curve fitting using a least-squares method, namely the Levenberg-Marquardt method [Moré 1977], to estimate the parameters of the Weibull distribution for each domain. For all estimates, the statistical significance was at the 0.001% level. Table V summarizes the results.

[9]This meant that we had to compute a separate baseline hazard function for each domain.

[10]Typically, the first choice for modeling survival times is the exponential distribution. If the survival times do not follow the exponential distribution, then the Weibull distribution, itself a generalization of the exponential distribution, is the next natural choice.

An interesting conclusion is that the survival functions do not follow the exponential distribution ($\gamma = 1$). Previous studies [Cho and García-Molina 2003] indicated that individual web *documents* have lifetimes that follow the exponential distribution. Our results, though, indicate that content summaries, with aggregate statistics about *sets of documents*, change more slowly. Another interesting result is that the λ_{dom} values are significantly lower for *FPS* than for *QBS* and *CMPL*. This result is due to the significantly higher weight assigned to the κ_1 feature when *FPS* is used. As discussed above, *FPS* retrieves the same documents each time it samples a database, as long as the database does not change. Hence, any changes in the retrieved documents are a strong signal that the database has changed, while this is not the case for *QBS*. Furthermore, a change of a single document in the *FPS* sample typically signals that many other documents in the database have changed as well. This is in contrast to *CMPL*, in which a change in a single document does not imply other changes in the database. For this reason, κ_1 has a higher weight for *FPS*, resulting in baseline functions with significantly lower λ_{dom} values than their *QBS* and *CMPL* counterparts.

4.3.4 *Modeling Conclusions.* We have presented a statistical analysis of the survival times of database content summaries. We used Cox regression analysis to examine the effect of different variables in the survival time of database content summaries and showed that the survival times of content summaries follow the Weibull distribution, in most cases with $\gamma < 1$ (i.e., summaries tend to remain unchanged for longer time periods as their age increases). We summarize our results in the following definition:

Definition 4.2. The function $S_i(t)$ that gives the survival function for a database D_i is:

$$S_i(t) = \exp\left(-\lambda_i t^{\gamma_{dom}}\right), \quad \text{with} \tag{3a}$$

$$\lambda_i = \lambda_{dom}\left(|D_i|^{\beta_s} \cdot \exp\left(\beta_\kappa \kappa_{1i}\right) \cdot \exp\left(\beta_\tau \tau_i\right)\right) \tag{3b}$$

where $|D_i|$ is the size of the database; κ_{1i} is the KL divergence of the content summaries obtained for D_i during the training period; τ_i is the value of the change sensitivity threshold for D_i (Definition 4.1); β_s, β_κ, and β_τ are the Cox coefficients from Table IV; and λ_{dom} and γ_{dom} are the domain-specific constants from Table V.

Definition 4.2 provides a concrete change model for a database D that is specific to the database characteristics and to the change sensitivity, as controlled by the threshold τ. An interesting result is that summaries of large databases change more often than those of small databases, as indicated by the positive value of β_s, which corresponds to the database size. Figure 4 shows the shape of $S(t)$ for different domains, for a hypothetical database D with $|D| = 1000$, $\kappa_1 = 0.1$ (computed using *QBS*), and for $\tau = 0.5$. This figure shows that content summaries tend to vary substantially across domains (e.g., compare the "misc" curve against the "gov" curve).

4.4 Deriving an Update Policy

So far, we have described how to compute the survival function $S(t)$ for a text database. Now, we describe how we can exploit $S(t)$ to schedule database content

Method	Features	Domain	λ_{dom}	γ_{dom}
QBS	κ_1, size, τ	com	0.0180	0.901
		edu	0.0205	0.585
		gov	0.0393	0.780
		misc	0.0236	1.050
		org	0.0274	0.724
	size, τ	com	0.0211	0.844
		edu	0.0392	0.578
		gov	0.0193	0.701
		misc	0.0163	1.072
		org	0.0239	0.723
	κ_1, τ	com	0.0320	0.886
		edu	0.0774	0.576
		gov	0.0245	0.795
		misc	0.0500	1.014
		org	0.0542	0.715
FPS	κ_1, size, τ	com	2.65×10^{-4}	0.787
		edu	3.40×10^{-4}	0.670
		gov	1.85×10^{-4}	0.710
		misc	1.90×10^{-4}	1.020
		org	3.74×10^{-4}	0.764
	size, τ	com	0.0211	0.844
		edu	0.0392	0.578
		gov	0.0193	0.701
		misc	0.0163	1.072
		org	0.0239	0.723
	κ_1, τ	com	7.59×10^{-5}	0.743
		edu	1.20×10^{-4}	0.641
		gov	5.92×10^{-5}	0.722
		misc	6.69×10^{-5}	0.920
		org	7.46×10^{-5}	0.728
CMPL	κ_1, size, τ	com	0.0315	0.800
		edu	0.0267	0.784
		gov	0.0181	0.767
		misc	0.00589	1.41
		org	0.02587	0.811
	size, τ	com	0.0241	0.753
		edu	0.0364	0.683
		gov	0.0209	0.685
		misc	0.0227	1.020
		org	0.0260	0.773
	κ_1, τ	com	0.1058	0.7093
		edu	0.1044	0.7371
		gov	0.0729	0.7478
		misc	0.0294	1.24
		org	0.0942	0.7685

Table V. The parameters for the baseline survival functions for the five domains. The baseline survival functions describe the survival time of a database D in each domain with size $|D| = 1$ ($\ln(|D|) = 0$), with average distance between the summaries $\kappa_1 = 0$, and for change sensitivity threshold $\tau = 0$.

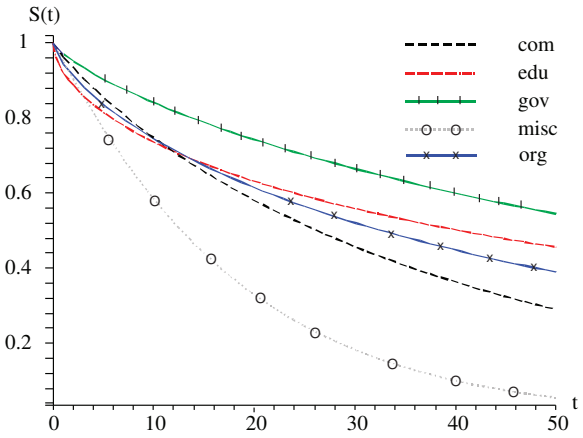


Fig. 4. The survival function $S(t)$ for different domains ($|D| = 1000$, $\tau = 0.5$, $\kappa_1 = 0.1$, *QBS* summaries).

summary updates and contact each database only when necessary.

A metasearcher may provide access to hundreds or thousands of databases and operate under limited network and computational resources. To optimize the overall quality of the content summaries, the metasearcher has to carefully decide when to update each of the summaries, so that they are acceptably up to date during query processing.

To model the constraint on the workload that a metasearcher might handle, we define F as the average number of content summary updates that the metasearcher can perform in a week. Then, under a *Naive* strategy that allocates updates to databases uniformly, $T = \frac{n}{F}$ represents the average number of weeks between two updates of a database, where n is the total number of databases. For example, $T = 2$ weeks means that the metasearcher can update the content summary of each database every two weeks, on average.

As we have seen in Section 4.3, the rate of change of the database contents may vary drastically from database to database, so the *Naive* strategy above is bound to allocate updates to databases suboptimally. Thus, the goal of our update scheduling is to determine the update frequency f_i for each database D_i individually, in such a way that the function $\sum_{i=1}^{n} S_i(t)$ is maximized, while at the same time not exceeding the number of updates allowed. In this case, we maximize the average probability that the content summaries are up to date. One complication is that the survival function $S_i(t)$ changes its value over time, so different update scheduling policies may be considered "optimal" depending on when $S_i(t)$ is measured. To address this issue, we assume that the metasearcher wants to maximize the *time-averaged* value of the survival function, given as:

$$\bar{\mathcal{S}} = \lim_{t \to \infty} \frac{1}{t} \int_0^t \sum_{i=1}^{n} S_i(t) dt.$$

This formulation of the scheduling problem is similar to that in [Cho et al. 2000]

D_i	λ_i	$T = 40$	$T = 10$
`tomshardware.com`	0.088	46 weeks	5 weeks
`usps.com`	0.023	34 weeks	12 weeks

Table VI. Optimal content-summary update frequencies for two databases.

for the problem of keeping the index of a search engine up to date. In short, we formulate our goal as the following optimization problem.

PROBLEM 4.3. *Find the optimal update frequency f_i for each database D_i, $i = 1, \ldots, n$, such that the time-averaged survival function $\bar{\mathcal{S}}$ is maximized under the constraint $\sum_{i=1}^{n} f_i = \frac{n}{T}$, where T is the average number of weeks between two updates of a database.*

Given the analytical forms of the $S_i(t)$ functions in the previous sections, we can solve this optimization problem using the *Lagrange-multiplier method* (as shown, for example, in [Cho et al. 2000; Olston and Widom 2002]). From this analysis, we conclude that the optimal update frequencies f_i's are the solutions to the following equations:

$$\frac{1}{\gamma_i \lambda_i^{\frac{1}{\gamma_i}}} \left[\Gamma\left(\frac{1}{\gamma_i}\right) - \Gamma\left(\frac{1}{\gamma_i}, \frac{\lambda_i}{f_i^{\gamma_i}}\right) \right] - \frac{1}{f_i} \exp\left(-\frac{\lambda_i}{f_i^{\gamma_i}}\right) = \mu \quad \text{for } 1 \leq i \leq n \tag{4}$$

where λ_i is the rate of change of database D_i, $\gamma_i = \gamma_{dom}$ of D_i, $\Gamma(x)$ and $\Gamma(x, y)$ are the complete and the incomplete gamma functions,[11] respectively, and μ is the Lagrange multiplier decided from the constraint $\sum_{i=1}^{n} f_i = \frac{n}{T}$. Note that the optimal frequencies f_i cannot be expressed in simple analytical form, so they need to be computed numerically by solving Equation 4.

Cho et al. [2000] investigated a special case of this optimization problem when $\gamma_i = 1$ (i.e., when the rate of change is constant over time) and observed the following:

(1) When λ_i is small relative to the resource constraint F (i.e., when the database changes infrequently compared to our update resource constraint), the optimal revisit frequency f_i becomes larger as λ_i grows larger.
(2) When λ_i is large relative to the resource constraint F, the optimal revisit frequency f_i becomes smaller as λ_i grows larger.

In our solution to the above generalized optimization problem, we also observed similar trends even when $\gamma_i \neq 1$ (i.e., when the rate of change varies over time). As an example, in Table VI we show the optimal update frequencies for the content summaries of two databases, `tomshardware.com` and `usps.com`. We can see that, when T is small and we can update summaries often (i.e., for $T = 10$, meaning that we update the summaries every 10 weeks on average), we update `tomshardware.com` more often than `usps.com`, since λ_i is larger for `tomshardware.com`. However, when T is large and we can only rarely update summaries (i.e., for $T = 40$, meaning that we update the summaries every 40 weeks on average), the optimal update

[11] By definition, $\Gamma(x) = \int_0^\infty t^{x-1} \exp(-t) dt$ and $\Gamma(x, y) = \int_y^\infty t^{x-1} \exp(-t) dt$.

frequencies are reversed. The scheduling algorithm decides that `tomshardware.com` changes "too frequently" and is not beneficial to allocate more resources to try to keep it up to date. Therefore, the algorithm decides to update the content summary from `tomshardware.com` less frequently, and instead focus on databases like `usps.com` that can be kept up to date. This trend holds across domains and across values of γ_i.

5. EXPERIMENTAL SETUP

We now describe the techniques that we compare for our experimental evaluation of Section 6. In Section 5.1 we describe the variations of our survival analysis approach that we used. Next, in Section 5.2, we describe a machine learning technique for predicting when content summaries should be updated. Finally, in Section 5.3 we describe a state-of-the-art method for scheduling when to recrawl (and update a web search engine index of) crawlable web sites; this method was originally presented by Cho and Ntoulas [2002] and we evaluate it as an alternative scheduling approach for updating content summaries of crawlable databases.[12]

5.1 Survival Analysis Techniques for Updating Content Summaries

In Section 4.3, we showed how to compute the form and parameters of the survival function $S_i(t)$, which measures the probability that the summary of a database D_i is up to date t weeks after it was computed. Based on Cox's model, we derived a variety of models that compute $S_i(t)$ based on three different sets of features (see Tables IV and V). Now, we use these models to devise three update policies, using the approach from Section 4.4 and the following feature sets:

—κ_1, size, and τ: We use all the available features.

—size and τ: We do not use the history of the database, i.e., we ignore the evolution feature κ_1 and we use only the database size and the change sensitivity threshold τ.

—κ_1 and τ: We use only the history of the database and the change sensitivity threshold τ. We consider this policy to examine whether we can work with databases without estimating their size.[13]

We also consider the *Naive* policy, discussed above, where we uniformly update all summaries every T weeks.

5.2 Machine Learning for Predicting Content Summary Changes

So far, we described a regression-based approach for scheduling updates for content summaries. The Cox proportional hazards regression returns (probabilistic) esti-

[12]As we discuss in Section 7, a number of update scheduling policies have been developed specifically for search engines, such as [Pandey and Olston 2005] and [Wolf et al. 2002]. While these policies show further improvement compared to [Cho and Ntoulas 2002], they exploit specific properties of the ranking functions used by the search engines. As a result, these optimizations are not directly applicable for our content summary update problem.

[13]The size estimation method that we use [Si and Callan 2003] relies on the database returning the number of matches for each query. This method becomes problematic for databases that do not report such numbers with the query results.

U	$week_1$	$week_2$	age	τ	$\ln(\|D\|)$	I_{com}	...	κ_1	...
0	1	2	1	0.5	12.3	1	...	0.23	...
0	1	3	2	0.5	12.3	1	...	0.23	...
1	1	4	3	0.5	12.3	1	...	0.23	...
1	1	5	4	0.5	12.3	1	...	0.23	...
⋮									
0	5	8	3	0.8	12.3	0	...	0.09	...
⋮									

Fig. 5. Example training vectors, used to train a classifier that detects whether a database content summary needs to be updated.

mates for the lifetime of each content summary, and then we exploit this information to schedule updates according to the available resources.

An alternative approach for updating content summaries is to treat scheduling as a binary classification problem. Specifically, we define a function $U(D_i, t_1, t_2)$ for a database D_i so that $U(D_i, t_1, t_2) = 1$ if the content summary of D_i extracted at time t_1 should be updated at time t_2, according to Definition 4.1, and $U(D_i, t_1, t_2) = 0$ otherwise. Following Section 4, we want the prediction to rely on *current* database features and perhaps also on a few *evolution* features of the database. Similarly to our survival analysis approach, we use leave-one-out cross-validation to train and test a "black-box" classifier that predicts the update status for a given unlabeled vector x: the training data for a database D_i consists of the vectors for all the other databases except for D_i. The classifier that we learn predicts whether we need to update the content summary of D_i.

5.2.1 *Creating the Training Set.* Given a database D_i and a change sensitivity threshold τ, we train a classifier for U as follows. For the duration of our study, spanning 50 weeks,[14] we consider each week pair $\langle p, q \rangle$ with $p < q$ and define $U(D_i, p, q) = 1$ if the KL-divergence between $C_q(D_i)$ and $C_p(D_i)$ is larger than τ, where $C_t(D_i)$ is the content summary of D_i at time t. The training set features are the *current* and *evolution* features that we defined in Section 4.3.2. Figure 5 shows a few example training vectors for database D_i. In this figure, the first vector corresponds to weeks 1 and 2. For change sensitivity threshold $\tau = 0.5$, $C(D_i)$ at week 2 is not substantially different from $C(D_i)$ as computed in week 1, hence $U(D_i, 1, 2) = 0$ for this week pair. The features associated with this database indicate that D_i is a *.com* database, that D_i contains 219,695 documents (i.e., $\ln(|D_i|) = 12.3$), and so on. Similarly, the third vector corresponds to weeks 1 and 4. For change sensitivity threshold $\tau = 0.5$, $C(D_i)$ at week 4 *is* substantially different from $C(D_i)$ as computed in week 1, hence $U(D_i, 1, 4) = 1$ for this week pair.

In our study, we had 152 databases, studied over a period of 50 weeks, with 10 possible values of the change sensitivity threshold τ. This resulted in a data set with 1,862,000 vectors. These vectors represent points in an m-dimensional

[14] We use two weeks' worth of our 52-week data to compute the κ_1 values.

space, where m is the number of *current* and *evolution* features. A *binary classifier* decides whether a vector, represented using m features (see above), belongs to the class $U = 1$ or not. A *binary linear classifier* makes this decision by calculating, during the training phase, m weights $w_1, \ldots, w_m$ and a threshold b determining a hyperplane such that every point $t = \langle t_1, \ldots, t_m \rangle$ in the hyperplane satisfies the equation:

$$\sum_{i=1}^{m} w_i \cdot t_i = b \tag{5}$$

This hyperplane divides the m-dimensional space into two regions: the region with the vectors that belong to the class in question, and the region with all other vectors.

5.2.2 *Selecting Classifiers.* Binary linear classifiers work best when the manifold that optimally separates [Duda et al. 2000] the two classes is a hyperplane. The results from Section 4.3 indicate that, in our case, the boundaries between the two classes cannot be described by a linear equation in the original m-dimensional space. Instead, the function that optimally separates the two classes involves products of the available features, and uses the logarithm of the content summary age, $\ln(age)$, as a feature. (See Appendix A for details.)

Based on this result, we initially added the $\ln(age)$ value as a separate feature and used SVMs with polynomial kernels to solve this classification problem. (SVMs with polynomial kernels implicitly expand the feature space to include extra features that correspond to the products of the "basic" features.) Unfortunately, polynomial SVMs cannot fully explore the expanded feature space. For example, all the terms that correspond to the products of the "basic" features are given equal weight [Hastie et al. 2001, page 384], a clearly undesirable property for our problem. (See Appendix A for details.) Therefore, we decided to experiment with SVMs with linear kernels. Since we used a linear classifier to discover a non-linear separating manifold, we had to expand our training set manually to include the needed quadratic features, which could be easily extracted from the existing training data. Specifically, instead of the vector $\mathbf{x} = \langle x_1, \ldots, x_m \rangle$, we used the enhanced vector $\mathbf{x}' = \langle x_1, \ldots, x_m, x_1^2, \ldots, x_m^2, \ldots, x_1 x_2, \ldots, x_{m-1} x_m \rangle$. The SVM classifier took more than a week to train, showing that classifiers with superlinear complexity are not suitable for data sets of this size. (The running time remained high even after training on a small random sample of the training set.) Furthermore, even after this long training time, the resulting classifier was a trivial classifier that always predicted $U = 1$ or $U = 0$, depending on the class distribution in the data set. Even after long efforts to use smaller data sets for training, the behavior of SVMs remained unsatisfactory and the training time remained prohibitively high. Based on these results, we abandoned the idea of using SVMs.

The next step was to use a Naive Bayes classifier [Duda et al. 2000], an extremely efficient classifier with time complexity linear in the size of the training set (i.e., a single scan over the training data suffices for training the classifier). Since Naive Bayes is a linear classifier, we again expanded our training set manually to include the needed quadratic features, which could be easily extracted from the existing training data. As we mentioned above, we use leave-one-out cross-validation to determine the update schedule for our content summaries.

One important shortcoming of the classification-based approach for scheduling updates is its inability to adapt to different levels of update resources. The only way that we can increase or decrease the required resources is to modify the change sensitivity threshold τ. This is in contrast with the survival analysis approach, which can adapt to the available resources even without modifying the change sensitivity threshold for the underlying databases.

5.3 Sampling-based Method for Updating Content Summaries

Cho and Ntoulas [2002] presented an approach for defining a recrawl schedule for web sites, with the goal of keeping a web search engine index up to date, subject to available "update resources." Their approach randomly samples a few pages from every web site to estimate the *fraction of changed pages* in each site since the last crawl. Based on these estimates, a greedy recrawl schedule is established, as follows. First, the site with the largest number of changed pages is crawled, followed by the second-most-changed site, and so on, until all available update resources are exhausted. We will refer to this update policy as *Sampling*.

The *Sampling* policy has been shown to work well to maintain a centralized index of the web. Unfortunately, this policy can only be applied to *crawlable* databases, because it assumes that we can retrieve the same set of documents repeatedly during sampling, which cannot be guaranteed for non-crawlable databases. In the following section, we will then use *Sampling* only for the complete content summary experiments, where we assume that the databases are crawlable (see Section 2). Throughout the experiments, and based on the analysis of [Cho and Ntoulas 2002], we use samples consisting of 20 documents.

6. EXPERIMENTAL RESULTS

We now report the results of the experimental comparison between the variations of our survival analysis approach against the alternative techniques. Section 6.1 describes the quality of the content summaries, while Section 6.2 focuses on the accuracy of the update schedules generated by the different techniques.

6.1 Quality of Content Summaries

We examine each update policy by measuring the average (weighted and unweighted) precision and recall, as well as the average KL divergence of the generated *approximate* summaries. For the survival analysis approaches and for the *Sampling* technique, we consider different values of T, where T is the average number of weeks between updates. The *Bayes* technique, as mentioned in Section 5.2, does not have a mechanism for adapting to different resource availability scenarios and the only way to increase or decrease its resource needs is to vary the change sensitivity threshold τ.

6.1.1 *Recall.* Our recall measurements indicate that the survival analysis approaches perform better than the alternatives. Figure 6 shows the average weighted and unweighted recall of the complete content summaries obtained under the scheduling policies that we consider. Specifically, our survival analysis techniques are always significantly better than the *Naive* policy, and outperform the *Sampling* policy for $T \leq 30$. For large values of T, we observe that *Sampling* performs roughly as

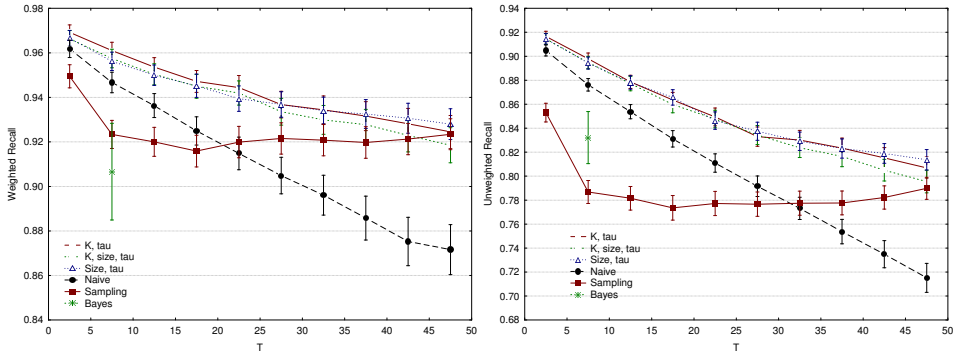


Fig. 6. The weighted and unweighted recall of "old" complete content summaries with respect to the "current" ones, as a function of the time T between updates and averaged over each database D in the data set, for different scheduling policies ($\tau = 0.5$).

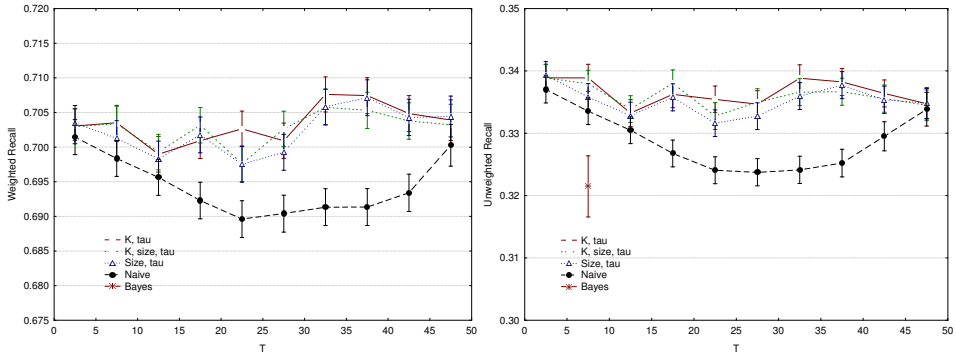


Fig. 7. The weighted and unweighted recall of "old" *QBS* content summaries with respect to the "current" ones, as a function of the time T between updates and averaged over each database D in the data set, for different scheduling policies ($\tau = 0.5$).

well as our survival analysis policies. The high performance of *Sampling* for large T values is mainly due to the fact that a small number of databases are responsible for the majority of the overall changes. By design, *Sampling* dedicates all of its resources to the small set of databases that have changed most, so it is able to capture the majority of the changes coming from few databases even when T is large. Unfortunately, as T gets smaller, *Sampling* fails to allocate the additional resources efficiently to the remaining databases; even if two databases have similarly changed, *Sampling* still dedicates all remaining resources only to one of them first. Due to this inefficiency, *Sampling* does not show much performance improvement as T gets smaller, until it can update all databases frequently. The *Bayes* approach results in a policy that, on average, updates databases every 7 weeks (i.e., $T = 7$). The results for *Bayes* are consistently worse than those for the survival analysis alternatives at this level of resource constraints: the average weighted recall under *Bayes* is 0.902 ± 0.025 and the average unweighted recall under *Bayes* is 0.83 ± 0.03. The *Bayes* approach only outperforms *Sampling*, which does not perform well for low values of T, as discussed above.

Figure 7 shows the average weighted and unweighted recall of the approximate

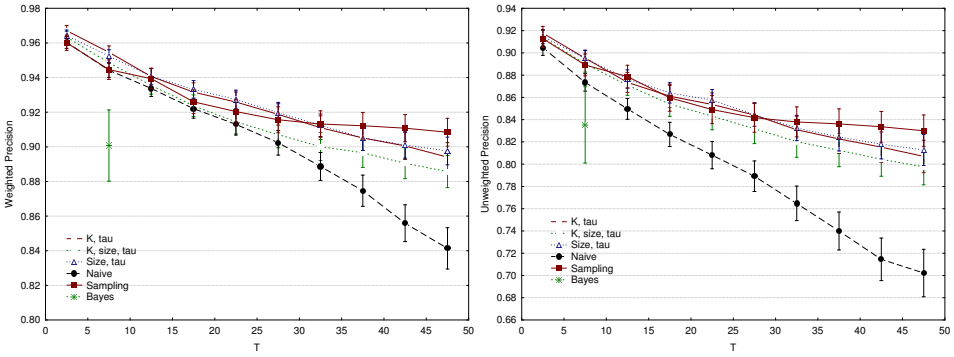


Fig. 8. The weighted and unweighted precision of "old" complete content summaries with respect to the "current" ones, as a function of the time T between updates and averaged over each database D in the data set, for different scheduling policies ($\tau = 0.5$).

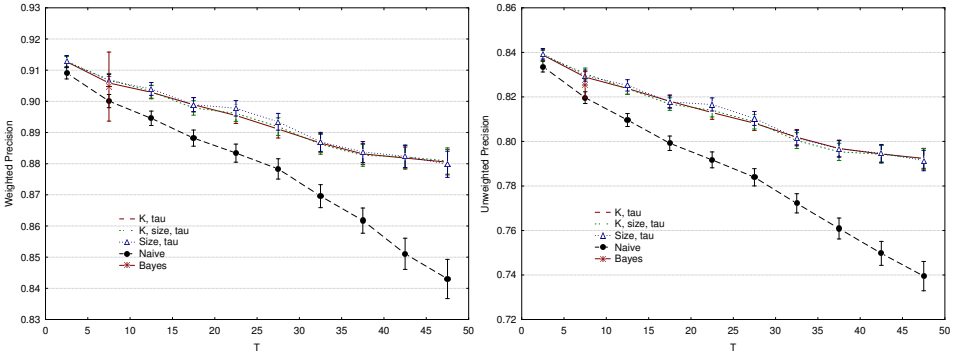


Fig. 9. The weighted and unweighted precision of "old" *QBS* content summaries with respect to the "current" ones, as a function of the time T between updates and averaged over each database D in the data set, for different scheduling policies ($\tau = 0.5$).

QBS summaries[15] for the survival analysis policies, *Naive*, and *Bayes*. (As discussed above, we do not include results for *Sampling*, since *Sampling* requires the ability to retrieve the same set of documents at each sampling instance, which cannot be guaranteed for non-crawlable databases.) The results indicate that, by using any of our survival analysis policies, we can keep the recall metrics almost stable, independently of the resource constraints. We also observe that the alternative approaches, namely *Naive* and *Bayes*, perform consistently worse than our survival analysis techniques.

6.1.2 *Precision.* Our precision measurements for complete content summaries (Figure 8) indicate that our survival analysis approach works significantly better than the *Bayes* alternative. Our survival analysis techniques are also significantly better than *Naive* for all values of $T > 15$. *Sampling* has lower precision than the survival analysis approach for small values of T, while the *Sampling* precision is higher when $T > 30$. In all cases, though, the differences between *Sampling* and our survival analysis techniques are not statistically significant, showing that the

[15]Figure 15 in Appendix B contains the respective results for *FPS* content summaries.

techniques are equivalent in terms of precision. However, as shown above, *Sampling* performs worse than survival analysis under the recall and KL metrics.

Figure 9 shows the average weighted and unweighted precision of the *QBS* summaries.[16] Again, our three survival analysis scheduling policies demonstrate similar[17] performance, and they are all significantly better than *Naive* and *Bayes*. The difference between the survival analysis policies and *Naive* is statistically significant, even when the summaries are updated relatively frequently (i.e., even for small values of T).

6.1.3 *KL-Divergence.* The measurements of KL-divergence for different scheduling policies reveal, again, that our survival analysis techniques can keep the average KL divergence of the approximate summaries almost constant even for a large number of weeks T between updates. Figure 10(a) shows that, for small values of T, *Sampling* is worse than our survival analysis techniques and is almost statistically equivalent to *Naive*. For larger values of T, the performance of *Sampling* improves; for $T > 35$, *Sampling* behaves similarly to the survival analysis techniques. However, our analysis shows that *Sampling* never outperforms our survival analysis techniques, which can also handle non-crawlable databases and work better for small values of T as well. The *Bayes* approach performs well for this metric, achieving performance similar to our policies. Nevertheless, *Bayes* lacks the ability to adjust automatically to environments with constraint resources. The only way to change the performance requirements of *Bayes* is to change the value of the change sensitivity threshold τ (for details, see below). Finally, Figure 10(b) shows the results for *QBS* content summaries.[18] The results are consistent with the behavior that we observed for complete content summaries. The only difference is the increased variance for the *Bayes* method, which indicates that our survival analysis policies are preferable, since they offer the same average performance but with a higher level of consistency across databases.

6.2 Precision of Update Operations

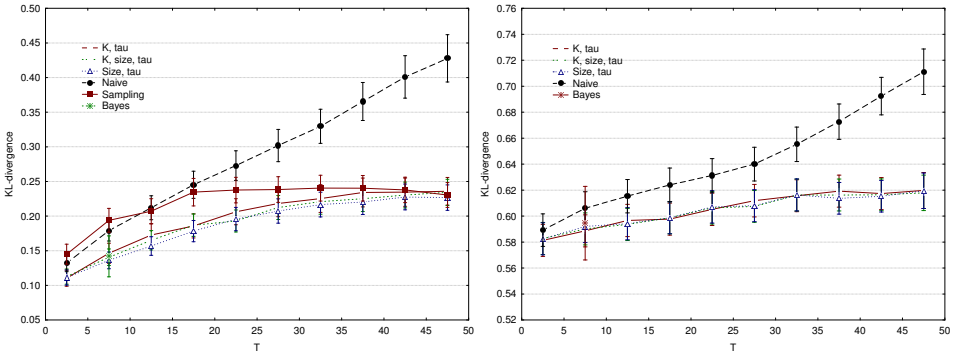
A scheduled update for a content summary might be unnecessary if the underlying database has not changed "sufficiently" since the time the summary was derived. Unnecessary content summary updates are of course undesirable, since they needlessly overload the databases. We now discuss how to characterize our update schedules in terms of whether their updates are necessary or not.

Consider a database D whose content summary was computed t weeks into the past. An update to this content summary is *precise* if the survival time of D is smaller than t, using Definition 4.1. In other words, we say that an update for D is *precise* if database D (and, correspondingly, its *complete* content summary; see Definition 4.1) has changed sufficiently in the t weeks since its content summary

[16] Figure 16 in Appendix B contains the respective results for *FPS* content summaries.

[17] The performance is statistically equivalent for the schedules based on "size and τ" and "κ_1 and τ" even at the individual database level, according to the Wilcoxon signed rank test [Marques De Sá 2003] ($p < 0.0001$). The performance of these two scheduling policies is *not* statistically equivalent with "κ_1, size, and τ" scheduling at the individual database level, but it is when we look at the average performance.

[18] Figure 17 in Appendix B contains the respective results for *FPS* content summaries.



(a) *CMPL* (b) *QBS*

Fig. 10. The KL divergence of "old" complete and *QBS* content summaries with respect to the "current" ones, as a function of the time T between updates and averaged over each database D in the data set, for different scheduling policies ($\tau = 0.5$).

was last computed.

We measured the *precision* of the update operations as the ratio of the precise updates over the total number of updates performed. Figures 11, 12, and 13 show the precision results as a function of T and for $\tau = 0.5$, where the κ_1 feature is computed using *CMPL*, *QBS*, and *FPS* summaries, respectively. For this value of τ and for the databases in our data set, low values of T (i.e., $T < 10$) are unnecessary, since then the databases are contacted too often and before they have changed sufficiently. A decrease in the value of τ causes the curves to "move" towards the left: the summaries change more frequently and then the updates become more precise. For example, for $\tau = 0.25$ and $T = 10$, precision is approximately 40%, while for $T = 25$, it is approximately 80%.

Interestingly, the update precision can be predicted analytically, using the target function $\bar{\mathcal{S}}$ described in Section 4.4. The average probability of survival (our target function) corresponds in principle to the percentage of non-precise updates. This result is intuitive, since our target function essentially encodes the probability that the summary of a database has changed. Therefore, during scheduling, it is possible to select a value of T that achieves (approximately) the desired update precision.

The results in Figures 11, 12, and 13 indicate that the *Naive* policy —as expected— has worse update precision than the other policies. Also, Figure 13 shows that the policy that uses *FPS* to compute κ_1 and does not use the *size* feature has significantly higher precision than the other techniques: the κ_1 feature computed using *FPS* is then a better predictor than the other variables, verifying the results of Cox regression, which returned a high weight for β_κ for the given policy (see Table IV).

As we mentioned in Section 5.2, the *Bayes* policy cannot adapt to the level of available resources. The classifier predictions depend only on the age of the summaries and hence the used resources vary from week to week. Figure 14 shows the number of updates performed by the *Bayes* policy when the κ_1 feature is computed using *CMPL* summaries, for different values of the change sensitivity threshold τ. (Figures 18 and 19, in Appendix B, show the respective results when the κ_1 feature is computed using *QBS* and *FPS* summaries, respectively.) The results show that the number of updates can vary greatly from week to week. Hence, to accommo-

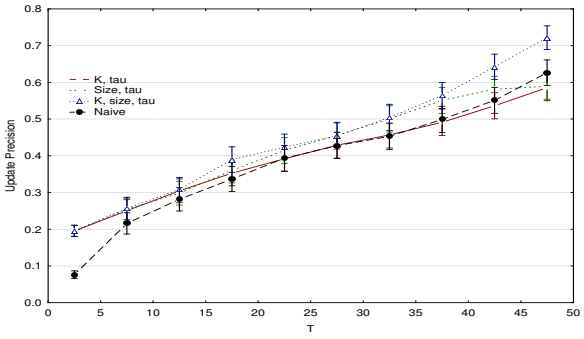


Fig. 11. The precision of the updates performed by the different scheduling algorithms, as a function of the average time between updates T and $\tau = 0.5$, where the κ_1 feature is computed using complete summaries.

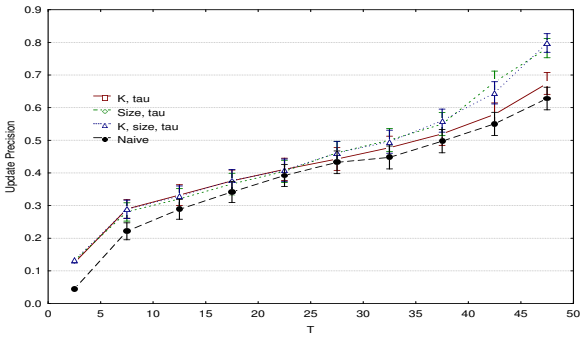


Fig. 12. The precision of the updates performed by the different scheduling algorithms, as a function of the average time between updates T and for $\tau = 0.5$, where the κ_1 feature is computed using *QBS* summaries.

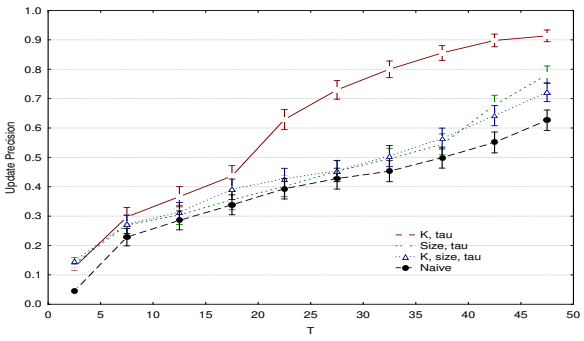


Fig. 13. The precision of the updates performed by the different scheduling algorithms, as a function of the average time between updates T and $\tau = 0.5$, where the κ_1 feature is computed using *FPS* summaries.

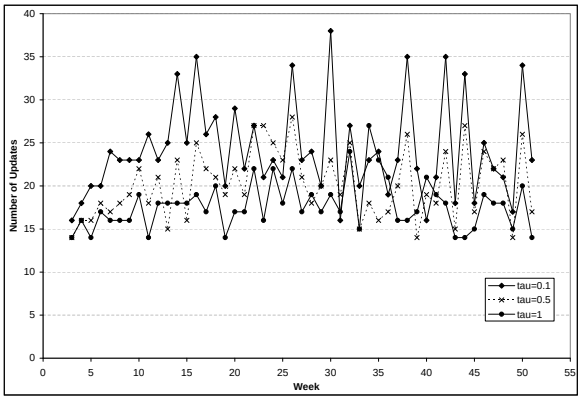


Fig. 14. The number of updates performed by the *Bayes* policy in different weeks, for different values of the change sensitivity threshold τ and for complete content summaries.

date the resource requirements of *Bayes*, we either need to allocate the maximum required resources or we should adopt a queuing policy, to delay updates for weeks in which the resource requirements are reduced. If we adopt a queuing policy, the summaries will be updated slightly later than expected. This will result in a slight deterioration of the content summary quality, which is already significantly lower than the quality of the summaries under the policies that use our survival analysis modeling.

6.3 Conclusion

As a general conclusion, we have observed that the three scheduling policies that are based on survival analysis allow for good quality of the extracted content summaries, even under strict constraints on the allowable update frequency. Our techniques work for both hidden-web and crawlable databases. For crawlable databases, our techniques outperform *Sampling*, an existing state-of-the-art update technique for a different task. Additionally, all techniques for both hidden-web and crawlable databases are significantly better than the other two alternatives that we studied, namely *Bayes* and *Naive*.

An interesting observation is that our three survival analysis policies demonstrate minimal differences in performance, and these differences are not statistically significant. This indicates that it is possible to work with a smaller set of features, without decreasing performance. For example, we may ignore the evolution feature κ_1 and avoid computing the history of a database, which involves frequent sampling of the database for a (small) period of time. We should also note that our survival analysis approach helps predict the precision of the update operations, in turn allowing the metasearcher to tune the update frequency to efficiently keep the content summaries up to date.

7. RELATED WORK

This article expands our earlier paper [Ipeirotis et al. 2005] on modeling content summary changes. In [Ipeirotis et al. 2005], we focused only on hidden-web data-

bases. In this article, we significantly expand the scope of our study to include the important family of *crawlable* web sites as well. Our Cox-regression approach for *crawlable* web sites significantly outperforms an existing state-of-the-art technique for scheduling updates of web search engine indexes [Cho and Ntoulas 2002]. Furthermore, in our previous study [Ipeirotis et al. 2005], we considered only the evolution of summaries extracted through a randomized query-based sampling (*QBS*) algorithm [Callan and Connell 2001]. One key characteristic of the *QBS* algorithm derives from its randomized nature: each execution of *QBS* results in a different summary, even if the underlying database is static. In this article, we now also study the evolution of summaries extracted through a state-of-the-art *deterministic* sampling algorithm, namely the focused probing algorithm from [Ipeirotis and Gravano 2002], *FPS*. The *FPS* algorithm repeatedly derives the same summary of a database if the database remains unchanged. Although we observe that the *FPS* samples are more stable than their *QBS* counterparts, we show that the quality of the *FPS* summaries still deteriorates over time and that our scheduling approach can improve the quality of *FPS* summaries as well. Finally, in this article we develop and evaluate a machine learning approach for updating content summaries. Since Cox-regression exploits only a specific kind of training, an open research question stemming from our previous work was whether alternative machine learning approaches could better exploit the available training data and outperform our survival analysis approach. In a thorough experimental comparison in Section 6, we showed that classification-based approaches for scheduling updates do not work well and we provided substantial evidence for the shortcomings of such approaches in terms of both efficiency and effectiveness.

Beyond [Ipeirotis et al. 2005], we are not aware of other prior work that studied the evolution of text database content summaries over time or how to schedule updates to the content summaries to maintain their freshness. However, several previous efforts have focused on various aspects of the evolution of the web and of the related problem of web crawling. Ntoulas et al. [2004] studied the changes of *individual* web pages, using the same data set as we did in this paper. Ntoulas et al. concluded that 5% of new content (measured in "shingles") is introduced in an average week in all pages as a whole. Additionally, Ntoulas et al. observed a strong correlation between the past and future rates of change of a web page and showed that this correlation might be used to predict future changes of a page. In this article, we investigated this high-level idea formally through survival analysis and modeled the change behavior of web databases using the Cox proportional hazard model. We then used this model for designing the optimal scheduling algorithm for summary updates. Lim et al. [2001] and Fetterly et al. [2003] presented pioneer measurements of the degree of change of web pages over time, where change was measured using the edit distance [Lim et al. 2001] or the number of changed "shingles" [Fetterly et al. 2003] over successive versions of the web pages. Other studies of web evolution include [Brewington and Cybenko 2000a; Cho and García-Molina 2000; Wills and Mikhailov 1999; Douglis et al. 1997; Brewington and Cybenko 2000b], and focus on issues that are largely orthogonal to our work, such as page modification rates and times, estimation of the change frequencies for the web pages, and so on.

Web crawling has attracted a substantial amount of work over the last few years. In particular, references [Cho et al. 2000; Coffman, Jr. et al. 1998; Edwards et al. 2001; Cho and Ntoulas 2002] study how a crawler should download pages to maintain its local copy of the web up to date. Assuming that the crawler knows the exact change frequency of the pages, Cho et al. [2000] and Coffman, Jr. et al. [1998] present optimal page downloading algorithms, while Edwards et al. [2001] propose an algorithm based on linear programming. Cho and Ntoulas [2002] propose the *Sampling* technique (see Section 5.3) for scheduling updates of web search engine indexes. Improved update policies for search engines include [Pandey and Olston 2005] and [Wolf et al. 2002]. The improvement that these policies exhibit over [Cho and Ntoulas 2002] is mainly due to optimizations that exploit particular properties of search-engine ranking functions. Since these optimizations are not directly applicable for content-summary updates, we compare our survival analysis techniques against [Cho and Ntoulas 2002], which assumes a more generic change metric. Our results in Section 6 show that our survival analysis policies outperform the *Sampling* approach for scheduling content summary updates for crawlable web sites.

Olston and Widom [2002] proposed a new algorithm for cache synchronization in which data sources notify caches of important changes. The definition of "divergence" or "change" in [Olston and Widom 2002] is quite general and can be applied to our context. However, the proposed push model is not applicable when data sources are "uncooperative" and do not inform others of their changes, as is often the case on the web.

8. CONCLUSIONS

In this article, we presented a study —over 152 real web databases— of the effect of time on the database content summaries on which metasearchers rely to select appropriate databases where to evaluate keyword queries. We examined the evolution of both complete and approximate content summaries. We showed that the quality of the content summaries deteriorates over time as the underlying databases change, which highlights the importance of update strategies for refreshing the content summaries. We described how to use survival analysis techniques, in particular how to exploit the Cox proportional hazards regression model, for this update problem. We showed that a short change history of a database can be used to predict the rate of change of its content summary in the future, and that summaries of larger databases tend to change faster than summaries of smaller databases. Based on the results of our analysis, we suggested update strategies that work well in a resource-constrained environment. Our techniques adapt to the change sensitivity desired for each database, and contact databases selectively —as needed— to keep the summaries up to date while not exceeding the resource constraints. Finally, our comparative evaluation shows that our survival analysis techniques significantly outperform an optimized machine learning approach and the current state-of-the-art technique for scheduling updates of web search engine indexes.

REFERENCES

Bergman, M. K. 2001. The Deep Web: Surfacing hidden value. *Journal of Electronic Publishing 7,* 1 (Aug.).

Brewington, B. E. and Cybenko, G. 2000a. How dynamic is the web? In *Proceedings of the Ninth International World Wide Web Conference (WWW9)*. 257–276.

Brewington, B. E. and Cybenko, G. 2000b. Keeping up with the changing web. *IEEE Computer 33,* 5 (May), 52–58.

Callan, J. P. 2000. Distributed information retrieval. In *Advances in Information Retrieval*. Kluwer Academic Publishers, 127–150.

Callan, J. P. and Connell, M. 2001. Query-based sampling of text databases. *ACM Transactions on Information Systems 19,* 2, 97–130.

Chakrabarti, S. 2002. *Mining the web*. Morgan Kaufmann.

Cho, J. and García-Molina, H. 2000. The evolution of the web and implications for an incremental crawler. In *Proceedings of the 26th International Conference on Very Large Databases (VLDB 2000)*. 200–209.

Cho, J. and García-Molina, H. 2003. Estimating frequency of change. *ACM Transactions on Internet Technology 3,* 3 (Aug.), 256–290.

Cho, J., García-Molina, H., and Page, L. 2000. Synchronizing a database to improve freshness. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD 2000)*. 117–128.

Cho, J. and Ntoulas, A. 2002. Effective change detection using sampling. In *Proceedings of the 28th International Conference on Very Large Databases (VLDB 2002)*. 514–525.

Coffman, Jr., E. G., Liu, Z., and Weber, R. R. 1998. Optimal robot scheduling for web search engines. *Journal of Scheduling 1,* 1 (June), 15–29.

Cox, D. R. 1972. Regression models and life-tables (with discussion). *Journal of the Royal Statistical Society B,* 34, 187–220.

Douglis, F., Feldmann, A., Krishnamurthy, B., and Mogul, J. C. 1997. Rate of change and other metrics: A live study of the world wide web. In *1st USENIX Symposium on Internet Technologies and Systems (USITS 1997)*. 16–31.

Duda, R. O., Hart, P. E., and Stork, D. G. 2000. *Pattern Classification*, 2nd ed. Wiley.

Edwards, J., McCurley, K. S., and Tomlin, J. A. 2001. An adaptive model for optimizing performance of an incremental web crawler. In *Proceedings of the 10th International World Wide Web Conference (WWW10)*. 106–113.

Fetterly, D., Manasse, M., Najork, M., and Wiener, J. 2003. A large-scale study of the evolution of web pages. In *Proceedings of the 12th International World Wide Web Conference (WWW12)*. 669–678.

Gravano, L., Chang, K. C.-C., García-Molina, H., and Paepcke, A. 1997. *STARTS*: Stanford proposal for Internet meta-searching. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data (SIGMOD'97)*. 207–218.

Gravano, L., García-Molina, H., and Tomasic, A. 1999. *GlOSS*: Text-source discovery over the Internet. *ACM Transactions on Database Systems 24,* 2 (June), 229–264.

Gravano, L., Ipeirotis, P. G., and Sahami, M. 2003. QProber: A system for automatic classification of hidden-web databases. *ACM Transactions on Information Systems 21,* 1 (Jan.), 1–41.

Hastie, T., Tibshirani, R., and Friedman, J. H. 2001. *The Elements of Statistical Learning*. Springer Verlag.

Ipeirotis, P. G. and Gravano, L. 2002. Distributed search over the hidden web: Hierarchical database sampling and selection. In *Proceedings of the 28th International Conference on Very Large Databases (VLDB 2002)*. 394–405.

Ipeirotis, P. G., Ntoulas, A., Cho, J., and Gravano, L. 2005. Modeling and managing content changes in text databases. In *Proceedings of the 21st IEEE International Conference on Data Engineering (ICDE 2005)*. 606–617.

Jelinek, F. 1999. *Statistical Methods for Speech Recognition*. The MIT Press.

Lim, L., Wang, M., Padmanabhan, S., Vitter, J. S., and Agarwal, R. C. 2001. Characterizing web document change. In *The Second International Conference on Web-Age Information Management (WAIM 2001)*. 133–144.

Marques De SÁ, J. P. 2003. *Applied Statistics*. Springer Verlag.

Moré, J. J. 1977. The Levenberg-Marquardt algorithm: Implementation and theory. In *Numerical Analysis, Lecture Notes in Mathematics 630, Springer Verlag*. 105–116.

Ntoulas, A., Cho, J., and Olston, C. 2004. What's new on the web? The evolution of the web from a search engine perspective. In *Proceedings of the 13th International World Wide Web Conference (WWW 2004)*. 1–12.

Olston, C. and Widom, J. 2002. Best-effort cache synchronization with source cooperation. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data (SIGMOD 2002)*. 73–84.

Pandey, S. and Olston, C. 2005. User-centric web crawling. In *Proceedings of the 14th International World Wide Web Conference (WWW 2005)*. 401–411.

Si, L. and Callan, J. P. 2003. Relevant document distribution estimation method for resource selection. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2003*. 298–305.

Stablein, D. M., Carter, Jr., W. H., and Novak, J. W. 1981. Analysis of survival data with nonproportional hazard functions. *Control Clinical Trials 2,* 2 (June), 149–159.

Wills, C. E. and Mikhailov, M. 1999. Towards a better understanding of web resources and server responses for improved caching. In *Proceedings of the Eighth International World Wide Web Conference (WWW8)*. 1231–1243.

Wolf, J. L., Squillante, M. S., Yu, P. S., Sethuraman, J., and Ozsen, L. 2002. Optimal crawling strategies for web search engines. In *Proceedings of the 11th International World Wide Web Conference (WWW11)*. 136–147.

A. APPENDIX: DERIVING FEATURES FOR CLASSIFICATION-BASED UPDATE SCHEDULING

The survival function $S_i(t)$ is the probability that the content summary of database D_i has not changed at time t. As we discussed in Section 4.3, $S_i(t)$ is computed as:

$$S_i(t) = \exp\left(-\lambda_i t^{\gamma_{dom}}\right), \quad \text{with} \tag{6a}$$

$$\lambda_i = \lambda_{dom}\left(|D_i|^{\beta_s} \cdot \exp\left(\beta_\kappa \kappa_{1i}\right) \cdot \exp\left(\beta_\tau \tau_i\right)\right) \tag{6b}$$

We saw in Section 5 how to cast the core of our content summary update problem as a binary classification task. In this formulation, our goal is to predict when $U(D_i, t_1, t_2) = 1$ for a database D_i, meaning that at time t_2 the content summary of database D_i extracted at time t_1 should be updated.

When the cost of false positives (wrongly predicting $U = 1$ when in reality $U = 0$) is equal to the cost of false negatives (wrongly predicting $U = 0$ when in reality $U = 1$), then an optimal classifier predicts $U = 1$ when $S_i(t) < 0.5$ and $U = 0$ when $S_i(t) \geq 0.5$. Therefore, the optimal manifold for separating the two classes is given by the equation:

$$\begin{aligned} S_i(t) &= 0.5 \Rightarrow \\ \exp\left(-\lambda_i t^{\gamma_{dom}}\right) &= 0.5 \Rightarrow \\ \lambda_i t^{\gamma_{dom}} &= \ln 2 \Rightarrow \\ \ln(\lambda_i) + \gamma_{dom} \ln(t) &= \ln(\ln 2) \end{aligned}$$

From Equation 3b, $\lambda_i = \lambda_{dom}\left(|D_i|^{\beta_s} \cdot \exp\left(\beta_\kappa \kappa_{1i}\right) \cdot \exp\left(\beta_\tau \tau_i\right)\right)$ and:

$$\begin{aligned} \ln(\lambda_{dom}\left(|D_i|^{\beta_s} \cdot \exp\left(\beta_\kappa \kappa_{1i}\right) \cdot \exp\left(\beta_\tau \tau_i\right)\right)) + \gamma_{dom} \ln(t) &= \ln(\ln 2) \Rightarrow \\ \ln(\lambda_{dom}) + \beta_s \ln(|D_i|) + \beta_\kappa \kappa_{1i} + \beta_\tau \tau_i + \gamma_{dom} \ln(t) &= \ln(\ln 2) \end{aligned}$$

Since the values of λ_{dom} and γ_{dom} depend on the value of the domain feature, which we express as a set of dummy binary variables for training the classifier, we set:

$$\begin{aligned}\lambda_{dom} &= I_{com}\lambda_{com} + I_{edu}\lambda_{edu} + I_{gov}\lambda_{gov} + I_{org}\lambda_{org} + I_{msc}\lambda_{msc}\\ \gamma_{dom} &= I_{com}\gamma_{com} + I_{edu}\gamma_{edu} + I_{gov}\gamma_{gov} + I_{org}\gamma_{org} + I_{msc}\gamma_{msc}\end{aligned}$$

Since only one of the binary variables can be equal to 1 for a database, we have:

$$\ln(\lambda_{dom}) = I_{com}\ln(\lambda_{com}) + I_{edu}\ln(\lambda_{edu}) + I_{gov}\ln(\lambda_{gov}) + I_{org}\ln(\lambda_{org}) + I_{msc}\ln(\lambda_{msc})$$

Now we have:

$$\begin{aligned}&\underline{I_{com}}\ln(\lambda_{com}) + \underline{I_{edu}}\ln(\lambda_{edu}) + \underline{I_{gov}}\ln(\lambda_{gov}) + \underline{I_{org}}\ln(\lambda_{org}) + \underline{I_{msc}}\ln(\lambda_{msc})\\ &+\underline{I_{com}}\gamma_{com}\underline{\ln(t)} + \underline{I_{edu}}\gamma_{edu}\underline{\ln(t)} + \underline{I_{gov}}\gamma_{gov}\underline{\ln(t)} + \underline{I_{org}}\gamma_{org}\underline{\ln(t)} + \underline{I_{msc}}\gamma_{msc}\underline{\ln(t)}\\ &+\beta_s\underline{\ln(|D_i|)} + \beta_\kappa\underline{\kappa_{1i}} + \beta_\tau\underline{\tau_i} + \ln(1/\ln 2) = 0\end{aligned} \tag{7}$$

The underlined terms are (functions of) features in the training vectors. A machine learning algorithm that computes the separating manifold should estimate the values of all the non-underlined terms, which correspond to the weights that a machine learning algorithm assigns to these features.

From Equation 7, we can see that a linear classifier does not suffice for separating the two classes optimally. First of all, features $\ln(|D_i|)$ and $\ln(t)$ do not appear among the original features. We could attempt to bypass this problem by adding these values as additional features in the training set. However, the terms $\underline{I.\gamma.\ln(t)}$ involve a product of features, making the manifold a non-linear surface, which is impossible to estimate with a linear classifier. Alternatively, we could add these features manually in the training set and create an augmented feature space. In this augmented feature space, the separating manifold is a hyperplane and a linear classifier can separate the two classes.

B. APPENDIX: ADDITIONAL EXPERIMENTAL RESULTS

In this section, we examine the effect of the different update policies on the quality of the approximate *FPS* content summaries. We measure the average (weighted and unweighted) recall (Figure 15), the average (weighted and unweighted) precision (Figure 16), as well as the average KL divergence (Figure 17) of the generated summaries. Furthermore, we present the number of updates performed by the *Bayes* policy when the κ_1 feature is computed using *QBS* and *FPS* summaries (Figures 18 and 19, respectively) for different values of the change sensitivity threshold τ.

Received July 2006; revised February 2007; accepted March 2007

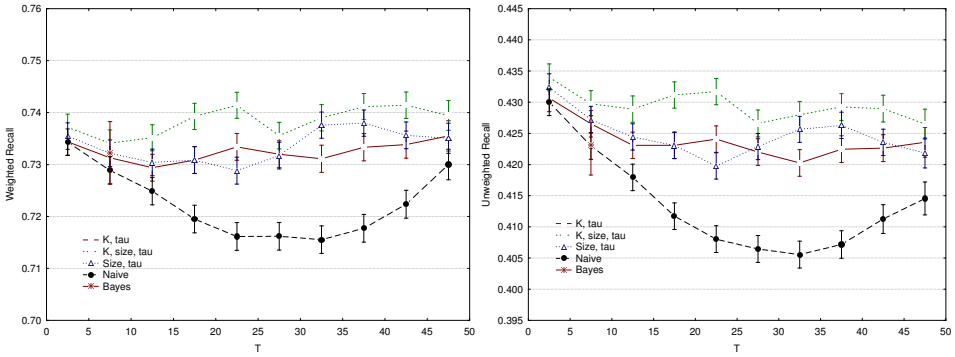


Fig. 15. The weighted and unweighted recall of "old" *FPS* content summaries with respect to the "current" ones, as a function of the time T between updates and averaged over each database D in the data set, for different scheduling policies ($\tau = 0.5$).

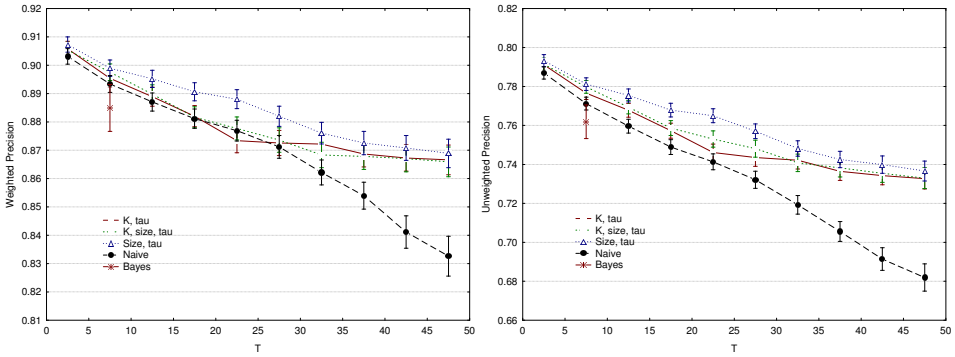


Fig. 16. The weighted and unweighted precision of "old" *FPS* content summaries with respect to the "current" ones, as a function of the time T between updates and averaged over each database D in the data set, for different scheduling policies ($\tau = 0.5$).

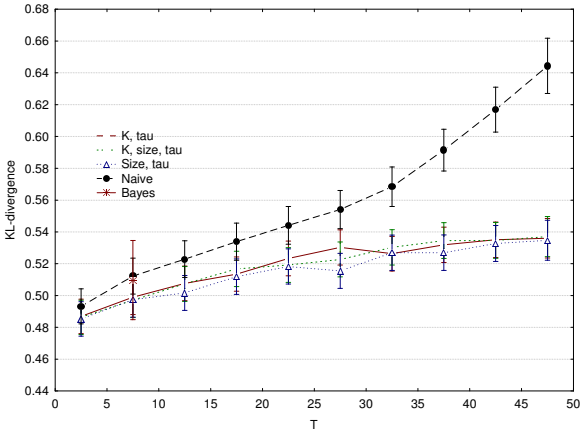


Fig. 17. The KL divergence of "old" *FPS* content summaries with respect to the "current" ones, as a function of the time T between updates and averaged over each database D in the data set, for different scheduling policies ($\tau = 0.5$).

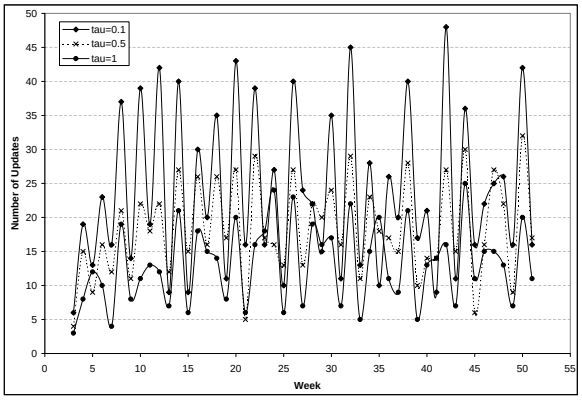


Fig. 18. The number of updates performed by the *Bayes* policy in different weeks, for different values of the change sensitivity threshold τ and for *QBS* summaries.

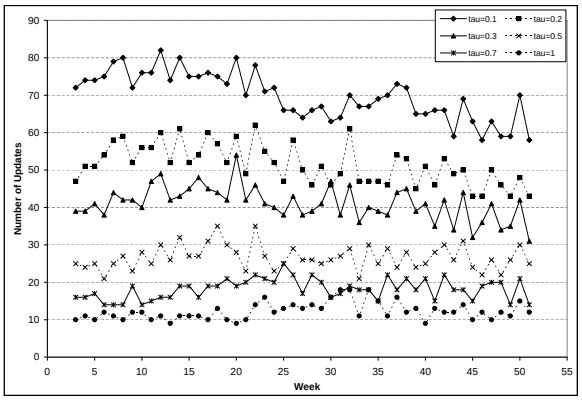


Fig. 19. The number of updates performed by the *Bayes* policy in different weeks, for different values of the change sensitivity threshold τ and for *FPS* summaries.