Modeling and Managing Content Changes in Text Databases

Panagiotis G. Ipeirotis* Alexandros Ntoulas† Junghoo Cho† Luis Gravano§

Abstract

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 need to change 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 paper, 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.

1. Introduction

A substantial amount of information on the web is stored in databases and is not indexed by search engines such as Google. One way to provide one-stop access to the information in text databases 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* [15] (or, alternatively, as *resource descriptions* [3]), usually include the *frequency* of the words that appear in the database, plus perhaps other simple statistics such as the number of documents in the database. These summaries, which provide sufficient information to decide which databases are the most promising for evaluating a given query, are the focus of this paper.

So far, database selection research has largely assumed that databases are static. However, real-life databases are not always static and 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 so that content summaries are kept up to date but without overloading databases unnecessarily to regenerate summaries that are already (at least close to) up to date.

In this paper, 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. Our study shows that summaries indeed change and that old summaries eventually become obsolete, which then calls for a content summary update strategy. To model content changes, we resort to the field of statistics named "survival analysis." Using the Cox proportional hazards regression model [10], we show that database characteristics can be used to predict the pattern of change of the summaries. Finally, we exploit our change model 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. To conclude the discussion, we report the results of an extensive experimental evaluation over our 152 real web databases, showing the effectiveness of our update strategies.

In brief, the contributions of this paper 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

* New York University
† University of California, Los Angeles
§ Columbia University

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

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

Table 1. A fragment of the content summaries of two databases.

change of database content summaries.

- In Section 5, we show how to update content summaries by exploiting our change model. The resulting strategies attempt to contact the databases only when strictly needed, thus avoiding wasting resources unnecessarily.

Finally, Section 6 discusses related work, while Section 7 provides further discussion and concludes the paper.

2. Background

This section introduces the notation and necessary background for this paper. We first define the notion of a "content summary" for a text database and briefly summarize how database selection algorithms exploit these summaries (see [18] for an expanded version of this discussion). Then, we review how to obtain database content summaries via querying.

Definition 1: *The* content summary $C(D)$ *of a database* D *consists of:*

- *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 1: *Consider the query* [cassini saturn] *and two databases* D_1 *and* D_2*. Based on the content summaries of these databases (Table 1), 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 [14]). Unfortunately, the so-called *hidden-web* databases [16], 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, other solutions have been proposed to automate the construction of content summaries from hidden-web databases that do not export such information.

Callan and Connell [4] presented an algorithm for building (approximate) content summaries of hidden-web text databases via document sampling. This algorithm first extracts a document sample (of about 300 documents) from a given database D via single-word queries. The document sample is then treated as a small database whose content summary is used to approximate that of D's. (Alternative query-based techniques [17] use different querying strategies.) In this paper, we use the document sampling and content summary approximation strategy from [4], and we use the "hat" notation to refer to an approximate content summary:

Definition 2: *The* 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 *as described in [4].*

Next, we present the results of our study that examined how 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 paper 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 later in the paper, 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 [23]. The databases were –roughly– the five top-ranked web sites in a subset of the topical categories of the Google Directory, which, in turn, reuses the hierarchical classification of web sites from the Open Directory Project. (Please refer to [23] 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.

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

Table 2. Domain distribution in our dataset.

Also, since we wanted to study content changes, we only selected databases with crawlable content, so that every week we can retrieve the databases' full contents using a crawler. A complete list of the sites included in our experiments is available at `http://webarchive.cs.ucla.edu/`. Table 2 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*).

We downloaded the contents of the 152 web sites every week over one year, up to a maximum of 200,000 pages per site at a time.[5] 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 treat 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 *approximate* content summaries. For this, we used query-based sampling (see Section 2) to create every week an approximate content summary $\hat{C}(D)$ of each database D.[6]

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 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.

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

6 To reduce the effect of sampling randomness in our experiments, we create five approximate 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.

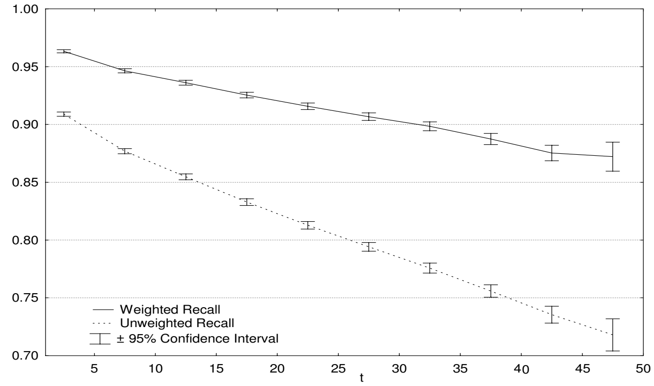

Figure 1. The recall of content summary $O(D,t)$ with respect to the "current" content summary $C(D)$, as a function of time t and averaged over each database D in the dataset.

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.

Figure 1 focuses on complete content summaries and shows the weighted and unweighted recall of t-week-old summaries with respect to the "current" summary, as a function of t and averaged over every possible choice of "current" summary. In Figure 1 (as well as in all subsequent figures), we report our results with a 95% confidence interval. 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 constitute a substantial fraction of the database vocabulary as well.

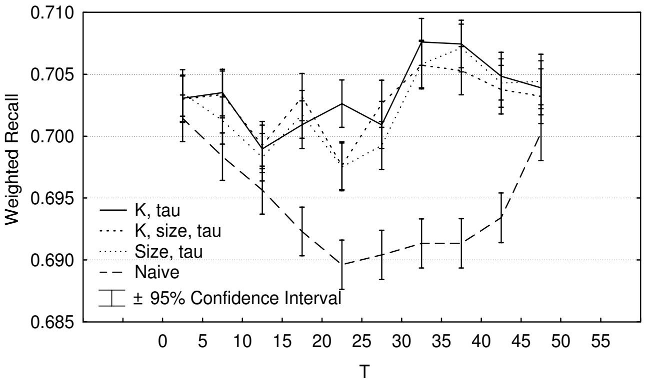

Figure 2. The weighted recall of "old" sample-based content summaries with respect to the "current" ones, as a function of the time T between updates and averaged over each database D in the dataset, for different scheduling policies ($\tau = 0.5$).

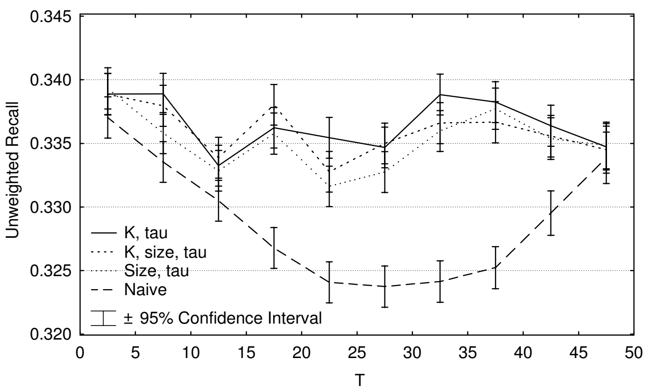

Figure 3. The unweighted recall of "old" sample-based content summaries with respect to the "current" ones, as a function of the time T between updates and averaged over each database D in the dataset, for different scheduling policies ($\tau = 0.5$).

The curves labeled "Naive" in Figures 2 and 3 show the corresponding results for approximate, sample-based content summaries. (Please ignore the other curves for now; we will explain their meaning in Section 5.) As expected, the recall values for the sample-based summaries are substantially smaller than the ones 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 not too accurate 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.

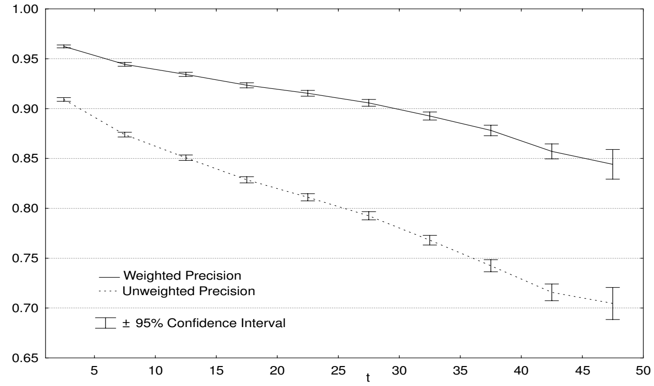

Figure 4. The precision of content summary $O(D, t)$ with respect to the "current" content summary $C(D)$, as a function of time t and averaged over each database D in the dataset.

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 that cover only a fraction of the database vocabulary, and is not observed in the complete summaries, perhaps because they are larger and are not substantially affected by a relatively small number of words.

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 *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 $C(D)$.

Figure 4 focuses on complete content summaries and

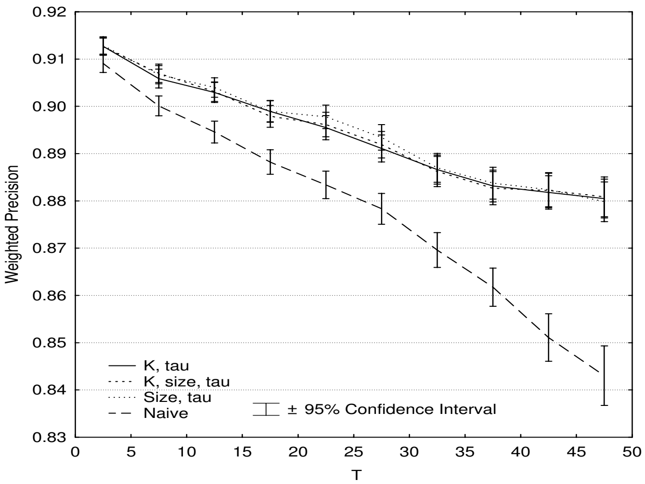

Figure 5. The weighted precision of "old" sample-based content summaries with respect to the "current" ones, as a function of the time T between updates and averaged over each database D in the dataset, for different scheduling policies ($\tau = 0.5$).

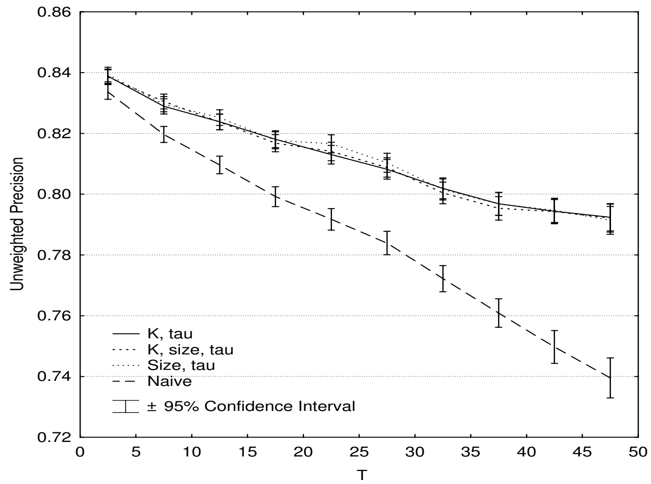

Figure 6. The unweighted precision of "old" sample-based content summaries with respect to the "current" ones, as a function of the time T between updates and averaged over each database D in the dataset, for different scheduling policies ($\tau = 0.5$).

shows the weighted and unweighted precision of 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 curves labeled "Naive" in Figures 5 and 6 show the corresponding results for approximate, sample-based content summaries. (Again, please ignore the other curves for now; we will explain their meaning in Section 5.) As expected, the precision values decrease over time, and do so much faster than their corresponding recall values (Figures 2 and 3). For example, almost 20% of the words in a 15-week-old sample-based content summary are absent from the database. For the precision results, the periodicity that appeared in the recall figures is not visible: the sample-based content summaries contain many more "obsolete" words that do not appear in the database anymore. Hence, a small number of words that appear periodically cannot improve the results.

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* [19] 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

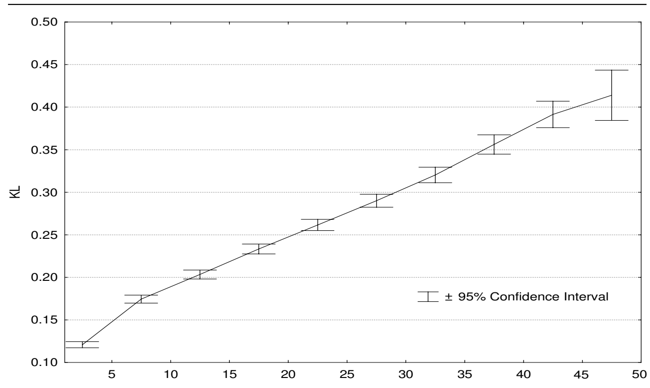

Figure 7. The KL divergence of content summary $O(D,t)$ with respect to the "current" content summary $C(D)$, as a function of time t and averaged over each database D in the dataset.

$C(D)$: $KL = \sum_{w \in W_o \cap W_c} p_c(w|D) \cdot \log \frac{p_c(w|D)}{p_o(w|D)}$, 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 content summaries being compared are equal. Intuitively, KL divergence measures how many bits are necessary to encode the difference between the two distributions.

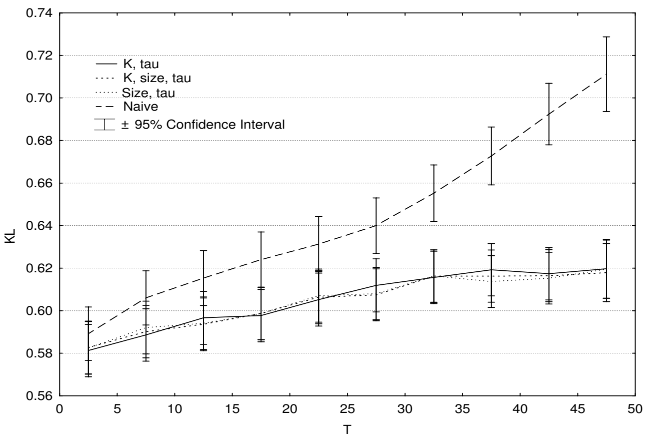

Figure 8. The KL divergence of "old" sample-based content summaries with respect to the "current" ones, as a function of the time T between updates and averaged over each database D in the dataset, for different scheduling policies ($\tau = 0.5$).

Figure 7 focuses on complete content summaries and shows that the KL divergence of old content summaries $O(D, t)$ increases as t increases. This confirms the previously observed results and shows that the word frequency distribution changes substantially over time. The curve labeled "Naive" in Figure 8 shows the KL divergence for sample-based content summaries of increasing age. (Again, please ignore the other curves for now; we will explain their meaning in Section 5.) The KL divergence of the old summaries increases with time, indicating that approximate content summaries become obsolete just as their complete counterparts do.

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 turn now 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 [21]. Then, Section 4.3 shows how we can use this material for our own scenario, to model content summary changes.

4.1. Survival Analysis

Survival analysis is a collection of statistical techniques that help predict the time until an event occurs [21]. 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 is the number of weeks t such that an old database content summary $O(D, t)$ is "sufficiently different" from the current summary $C(D)$. (We define formally 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: $h(t) = -\frac{\frac{dS(t)}{dt}}{S(t)}$. 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 ($h(t) = \lambda\gamma t^{\gamma-1}$). We could use the exponential function to model the database survival time. This choice is reinforced by recent findings that indicate that the exponential function is a good model to describe changes in web *documents* [1, 6]. However, we will see in Section 4.3 that the exponential distribution does not describe accurately changes for *database* summaries, and we will use the Weibull distribution instead.

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 their 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 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 influences 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* [10] is a technique widely used in statistics for discovering important variables that influence survival times. It is a non-parametric model, 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[7] 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 A and B, respectively. Under the Cox model, the hazard functions $h_A(t)$ and $h_B(t)$ can be expressed for databases A and B as:

$$h_A(t) = e^{\beta x_A} h_0(t) \Rightarrow \ln h_A(t) = \ln h_0(t) + \beta x_A \quad \text{(1a)}$$
$$h_B(t) = e^{\beta x_B} h_0(t) \Rightarrow \ln h_B(t) = \ln h_0(t) + \beta x_B \quad \text{(1b)}$$

where $h_0(t)$ is a *baseline hazard function*, common for all the members of the population. The Cox model can be generalized for n predictor variables: $\log h(t) = \log h_0(t) + \sum_{i=1}^{n} \beta_i x_i$, where the x_i's are the predictor variables, and the β_i's are the model coefficients. The algorithm presented by Cox [10] 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" data, from cases that "survived" the whole study period. Excluding these cases from the study would seriously affect the result, introducing 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 [10].)

The Cox proportional hazards model is one of the most general models for working with survival data, 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 dataset 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" (PH, or linearity) assumption, which means that for two individual groups A and B the hazard ratio $\frac{h_A(t)}{h_B(t)}$ is constant over time.

An interesting variation of the Cox model that overcomes the PH assumption is the *stratified Cox model* [26], which is used to account for variables that do not satisfy the proportionality assumption. In this case, the variables that do not satisfy the proportionality assumption are used to split the dataset into different "strata." The β_i Cox coefficients remain the same across the different strata, but each stratum now has different baseline functions $h_0(t)$.

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 its content summary, 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.[8] 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 5, we show that we can define update schedules that adapt to the chosen value of τ.

Definition 3: *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.*

Computing Survival Times: Using the study of Section 3 as well as Definition 3, we computed the survival time of each content summary for different values of 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 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.

7 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.

8 We use KL divergence for our change definition (as opposed to precision or recall) because KL depends on the whole word-frequency distribution. 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.

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 and its DNS domain 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. For the remainder of the discussion, we focus on the features for the important case of approximate, sample-based content summaries. Analogous features can be defined for crawlable databases, for which we can extract complete summaries.

The initial set of *current* features that we used was:

- The threshold τ.
- The logarithm of the estimated size of the database, where we estimate the size of the database using the "sample-resample" method from [25].
- The number of words in the current sample $\hat{C}(D)$.
- 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 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.

To extract the set of *evolution* features, we retrieved sample-based content summaries from each database every week over a period of 10 weeks. Then, for each database we compared every pair of *approximate* 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.[9] The results of the initial run indicated that, from the *current* features, the number of words and the topic of the database are not good predictor variables, while from the *evolution* features precision and recall are not good predictor variables; the KL features are good predictors, and strongly and positively correlated with each other.

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

Features	β_s	β_κ	β_τ
size, τ	0.179	-	-1.313
κ_1, τ	-	8.3	-1.308
κ_1, size, τ	0.094	6.762	-1.305

Table 3. The coefficients of the Cox model, when trained for various sets of features.

Given these results, we decided to drop the number of words and the topic variables from the *current* set, keeping only the threshold τ, the database size, and the domain. From the *evolution* set we dropped the recall and precision features. Also, from the KL features we kept only the κ_1 feature: given its presence, features κ_2 through κ_9 were largely redundant. Furthermore, we reduced the training time from 10 to three weeks. To examine whether any of the selected features –other than threshold τ, which we always keep– are 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.

Training the Cox Model: After the initial feature selection, we trained the Cox model again. The results indicated that all the features that we had selected are good predictor variables[10] 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 *stratified Cox model*, stratifying on domain.[11]

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 3. 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_1 and D_2 from the same domain, we have:

$$\begin{aligned} \ln S_1(t) &= \exp(\beta_s \ln(|D_1|) + \beta_\kappa \kappa_{1_1} + \beta_\tau \tau_1) \cdot \ln S_0(t) \\ \ln S_2(t) &= \exp(\beta_s \ln(|D_2|) + \beta_\kappa \kappa_{1_2} + \beta_\tau \tau_2) \cdot \ln S_0(t) \end{aligned}$$

where $S_0(t)$ is the baseline survival function for the respective domain. The baseline survival function corresponds to a

10 For all models, the statistical significance is at the 0.001% level according to the Wald statistic [21].

11 This meant that we had to compute separate baseline hazard functions for each domain.

Features	Domain	λ_{dom}	γ_{dom}
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
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

Table 4. 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 $|D| = 1$ ($\ln(|D|) = 0$), $\kappa_1 = 0$ and $\tau = 0$.

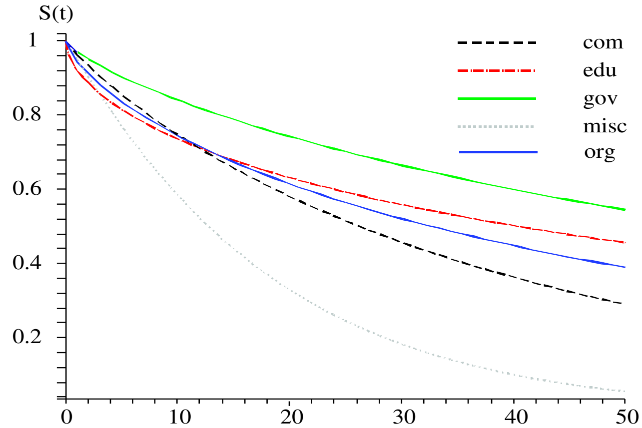

Figure 9. The survival function $S(t)$ for different domains ($|D| = 1,000$, $\tau = 0.5$, $\kappa_1 = 0.1$).

"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 remain unspecified and are defined only by a set of values $S_0(t_1), S_0(t_2), \ldots, S_0(t_n)$. 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 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 (in particular the Levenberg-Marquardt method [22]) to estimate the parameters of the Weibull distribution for each domain. For all estimates, the statistical significance was at the 0.001% level. Table 4 summarizes the results.

An interesting result is that the survival functions do not follow the exponential distribution ($\gamma = 1$). Previous studies [6] 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.

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 content summaries and showed that the survival times of content summaries follow the Weibull distribution, in most cases with $\gamma < 1$ (i.e., they tend to remain unchanged for longer time periods as their age increases). We summarize our results in the following definition:

Definition 4: *The function $S_i(t)$ that gives the survival function for a database D_i is:*

$$S_i(t) = \exp\left(-\lambda_i t^{\gamma_{dom}}\right), \quad \textit{with} \tag{2a}$$

$$\lambda_i = \lambda_{dom}\left(|D_i|^{\beta_s} \cdot \exp\left(\beta_\kappa \kappa_{1i}\right) \cdot \exp\left(\beta_\tau \tau_i\right)\right) \tag{2b}$$

where $|D_i|$ is the size of the database, κ_{1i} is the KL divergence of the samples obtained during the training period, β_s, β_κ, and β_τ are the Cox coefficients from Table 3, λ_{dom} and γ_{dom} are the domain-specific constants from Table 4, and τ_i is the value of the change threshold for D_i (Definition 3).

Definition 4 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 9 shows the shape of $S(t)$ for different domains, for a hypothetical database D with $|D| = 1000$, $\kappa_1 = 0.1$, 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).

5. Scheduling Updates

So far, we have described how to compute the survival function $S(t)$ for a text database. In this section, we describe how we can exploit $S(t)$ to schedule database content summary updates and contact each database only when necessary. Specifically, we first describe the theory behind our scheduling policy (Section 5.1). Then, we present the experimental evaluation of our policy (Section 5.2), which shows that sophisticated update scheduling can improve the quality of the extracted content summaries in a resource-restricted environment.

5.1. Deriving an Update Policy

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 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: $\bar{S} = \lim_{t\to\infty} \frac{1}{t}\int_0^t \sum_{i=1}^{n} S_i(t)dt$. This formulation of the scheduling problem is similar to that in [7] for the problem of keeping the index of a search engine up to date. We formulate our goal as the following optimization problem.

Problem 1: *Find the optimal update frequency f_i for each database D_i such that $\bar{S}$ is maximized under the constraint $\sum_{i=1}^{n} f_i = \frac{n}{T}$.*

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 [7, 24]). Cho et al. [7] investigated a special case of this optimization problem when $\gamma = 1$ (i.e., when the rate of change is constant over time), and observed the following:

1. When λ_i (which can be interpreted as denoting "how often the content summary changes") is small relative to the constraint F, the optimal revisit frequency f_i becomes larger as λ_i grows larger.
2. When λ_i is large compared 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 $\gamma \neq 1$ (i.e., when the rate of change varies over time). As an example, in Table 5 we show the optimal update frequencies for

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 5. Optimal content-summary update frequencies for two databases.

the content summaries of two databases, `tomshardware.com` and `usps.com`. We can see that, when T is small ($T = 10$), we update `tomshardware.com` more often than `usps.com`, since λ_i is larger for `tomshardware.com`. However, when T is large ($T = 40$) the optimal update 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.

5.2. Experimental Results

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 3 and 4). Now, we use these models to devise three update policies, using the approach from Section 5.1 and the following feature sets:

- κ_1, size, 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 threshold τ. We consider this policy to examine whether we can work with databases without estimating their size.[12]

We also consider the *Naive* policy, discussed above, where we uniformly update all summaries every T weeks.[13]

Quality of Content Summaries under Different Policies: We examine the performance of each updating policy, by measuring the average (weighted and unweighted) precision and recall, and the average KL divergence of the generated *approximate* summaries. We consider different values of T, where T is the average number of weeks between updates.

12 The size estimation method that we use [25] 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.

13 The results presented in this paper focus on sample-based content summaries. We also ran analogous experiments for the complete content summaries, and the results were similar.

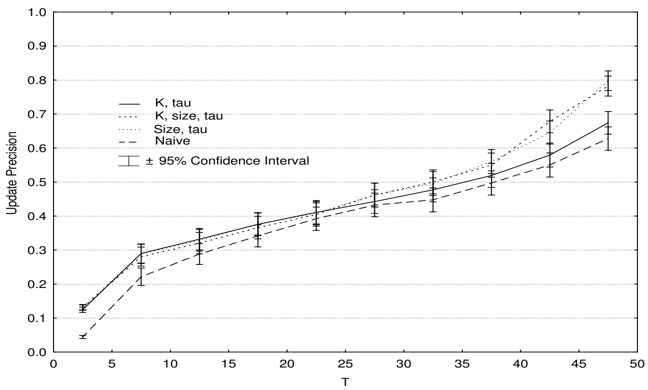

Figure 10. The precision of the updates performed by the different scheduling algorithms, as a function of the average time between updates T and for $\tau = 0.5$

Figures 2 and 3 show the average weighted and unweighted precision of the approximate summaries, obtained under the scheduling policies that we consider. The results indicate that, by using any of our policies, we can keep the recall metrics almost stable, independently of the resource constraints. Figures 5 and 6 show the average weighted and unweighted precision of the approximate summaries. Again, our three scheduling policies demonstrate similar performance, and they are all significantly better than the *Naive* policy. The difference with the *Naive* policy is statistically significant, even when the summaries are updated relatively frequently (i.e., even for small values of T). Finally, Figure 8 shows that our updating policies keep the average KL divergence of the approximate summaries almost constant even for a large number of weeks T between updates.

Interestingly, the three policies that we propose demonstrate minimal differences in performance, and these differences are not statistically significant. Additionally, all techniques are significantly better than the *Naive* policy. 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.

Precision of Update Operations: To measure how "precise" the updates scheduled by our policies are, we define an update as "precise" if it contacts a database when the new summary of the database is different from the existing summary according to the definition of change in Section 4.3. We measured the precision of the update operations as the ratio of the precise updates over the total number of updates performed. Figures 10 shows the precision results as a function of T and for $\tau = 0.5$. For this value of τ and for the databases in our dataset, very 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 cause 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 $\bar{S}$ described in Section 5.1. 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 the database has changed. Therefore, during scheduling, it is possible to select a value of T that achieves (approximately) the desired update precision.

Conclusion: As a general conclusion, we have observed that our scheduling policies allow for good quality of the extracted content summaries, even under strict constraints on the allowable update frequency. Also, our modeling 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.

6. Related Work

We are not aware of prior work to experimentally measure database content summary evolution over time or to schedule updates to the content summaries to maintain their freshness. However, several previous studies have focused on various aspects of the evolution of the web and of the related problem of web crawling. Ntoulas et al. [23] studied the changes of *individual* web pages, using the same dataset 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, [23] observed a strong correlation between the past and the future degrees of the changes of a web page and showed that this correlation might be used to predict the future changes of a page. In this paper (Section 3), we investigated this high-level idea more formally through survival analysis and modeled the change behavior of web databases using the Cox proportional hazard model. This model was then used for designing the optimal scheduling algorithm for summary updates. Lim et al. [20] and Fetterly et al. [13] presented pioneer measurements of the degree of change of web pages over time, where change was measured using the edit distance [20] or the number of changed "shingles" [13] over successive versions of the web pages. Other studies of web evolution include [1, 5, 27, 11, 2], 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 [7, 9, 12, 8] 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 frequencies of pages, references [7, 9] present optimal page downloading algorithms, while [12] proposes an algorithm based on linear programming. Cho and Ntoulas [8] employ sampling to detect changed pages. All this work on web crawling mainly focuses on maintaining a local copy of the web as up-to-date as possible, which requires maximizing the fraction of remote pages whose local copy is up to date. Our goal is different: we want to maximize the freshness of the content summaries that describe the various web sites, so that we produce more accurate database selection decisions.

Olston et al. [24] proposed a new algorithm for cache synchronization in which data sources notify caches of important changes. The definition of "divergence" or "change" in [24] 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 the case on the web.

7. Conclusions

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. Predictably, 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 the 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. Finally, 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.

References

[1] B. E. Brewington and G. Cybenko. How dynamic is the web? In *WWW9*, 2000.

[2] B. E. Brewington and G. Cybenko. Keeping up with the changing web. *IEEE Computer*, 33(5), 2000.

[3] J. P. Callan. Distributed information retrieval. In *Advances in Information Retrieval*. Kluwer Academic Publishers, 2000.

[4] J. P. Callan and M. Connell. Query-based sampling of text databases. *ACM TOIS*, 19(2), 2001.

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

[6] J. Cho and H. García-Molina. Estimating frequency of change. *ACM TOIT*, 3(3), 2003.

[7] J. Cho, H. García-Molina, and L. Page. Synchronizing a database to improve freshness. In *SIGMOD*, 2000.

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

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

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

[11] F. Douglis, A. Feldmann, B. Krishnamurthy, and J. C. Mogul. Rate of change and other metrics: A live study of the world wide web. In *USITS*, 1997.

[12] J. Edwards, K. S. McCurley, and J. A. Tomlin. An adaptive model for optimizing performance of an incremental web crawler. In *WWW10*, 2001.

[13] D. Fetterly, M. Manasse, M. Najork, and J. Wiener. A large-scale study of the evolution of web pages. In *WWW12*, 2003.

[14] L. Gravano, K. C.-C. Chang, H. García-Molina, and A. Paepcke. *STARTS*: Stanford proposal for Internet meta-searching. In *SIGMOD*, 1997.

[15] L. Gravano, H. García-Molina, and A. Tomasic. *GlOSS*: Text-source discovery over the Internet. *ACM TODS*, 24(2), 1999.

[16] L. Gravano, P. G. Ipeirotis, and M. Sahami. QProber: A system for automatic classification of hidden-web databases. *ACM TOIS*, 21(1), 2003.

[17] P. G. Ipeirotis and L. Gravano. Distributed search over the hidden web: Hierarchical database sampling and selection. In *VLDB*, 2002.

[18] P. G. Ipeirotis and L. Gravano. When one sample is not enough: Improving text database selection using shrinkage. In *SIGMOD*, 2004.

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

[20] L. Lim, M. Wang, S. Padmanabhan, J. S. Vitter, and R. C. Agarwal. Characterizing web document change. In *WAIM*, 2001.

[21] J. P. Marques De Sá. *Applied Statistics*. Springer Verlag, 2003.

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

[23] A. Ntoulas, J. Cho, and C. Olston. What's new on the web? The evolution of the web from a search engine perspective. In *WWW13*, 2004.

[24] C. Olston and J. Widom. Best-effort cache synchronization with source cooperation. In *SIGMOD*, 2002.

[25] L. Si and J. P. Callan. Relevant document distribution estimation method for resource selection. In *SIGIR*, 2003.

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

[27] C. E. Wills and M. Mikhailov. Towards a better understanding of web resources and server responses for improved caching. In *WWW8*, 1999.