Quality-Aware Retrieval of Data Objects from Autonomous Sources for Web-Based Repositories (Technical Report)

Houtan Shirani-Mehr, Chen Li, Gang Liang, and Michal Schmueli-Scheuer
University of California, Irvine, CA 92697, USA
{hshirani,chenli,liang,mshmueli}@uci.edu

Abstract

In the age of Internet, the performance of many Web-based applications critically depend on the quality of local repositories. Due to the dynamics of remote Web sources, applications have to crawl information periodically to ensure the high quality of repositories. In this paper, we use the general quality-aware principle to strike balances between the quality of repositories and the resources for synchronization. The contributions of this paper are: 1) An extensive empirical study is conducted on six Web sites from four domains (cars, books, job postings, and forums), with about 140 *thousand objects collected during a period of 1.5 years. Our study provides valuable insights into the dynamics of Web sources; 2) An arrival-survival approach is proposed to model behavior of individual objects at remote sources, and the estimation of various quality metrics are derived; 3) A general adaptive framework based on time series analysis is proposed. This novel approach allows the underlying model be adaptive to the dynamics of remote sources; 4) An object partitioning approach is proposed based on object attributes to improve the accuracy of quality metric prediction. At last, our approaches are validated on real Web datasets, and good results are obtained on both quality prediction and resource allocation.*

1 Introduction

As the Web becomes the dominant medium for information delivery and commerce, an increasing number of applications have their services heavily rely on data repositories in utilizing information collected from various Web sources. The following example illustrates such a trend.

Example 1 *Comparison shopping service providers like MySimon.com offer shopping recommendations, buying advice, and side by side price comparisons for various products. They search and index products from a large number of merchant Web sites, and allow users to compare prices from competing stores by collecting a substantial amount of information from many autonomous sources on the Web. To provide high-quality services, they need to identify relevant Web sources, retrieve information from pages of these sources, and extract information from pages to populate their databases.*

Figure 1 shows the general architecture of such applications. The application service is powered by a backend repository with data retrieved from some Web sources, including "Deep Webs" whose information is hidden behind search forms. The middle layer extracts, cleans, and stores extracted information into local repositories.

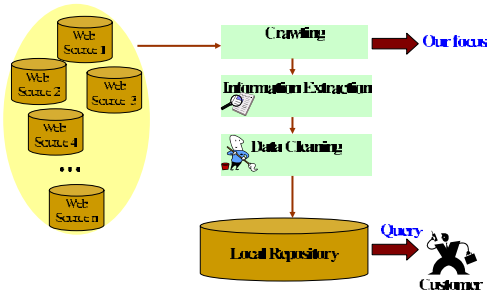


Figure 1. A Web-based repository architecture.

One of the most significant obstacles of building an efficient repository is that remote Web sources are volatile: new data are added; existing data are deleted or updated frequently. Furthermore, most remote Web sources are not collaborative in the sense that applications will not be notified for any change. As a consequence, applications have to query remote sources periodically in retrieving the latest information. Under such a setup, the local repository cannot mirror the remote sources completely; hence, *quality metrics* are used to quantify the goodness of the local repository

with respective to remote sources.

1.1 Related Work

Improving the quality of data from non-collaborative Web sources was extensively studied in the literature. The problem of quality assurance is viewed from various angles, such as Web page importance [10] and topic based crawling [6, 12].

From the perspective of quality assurance over time, [7, 8, 20] studied the problem of how to crawl Web pages to maximize their quality in a local repository. [5] extended the techniques by incorporating the information of user preference on Web pages. Their main quality metrics are freshness (a boolean value indicating whether an object is up to date) and age (elapsed time since last time an object was refreshed). Differently, our research focuses on structural data (objects) rather than Web pages or word frequencies in text databases, and the main quality metrics used in our study are precision and recall for a collection of objects.

In order to retrieve information more efficiently from non-collaborate sources, it is of importance to detect remote changes or to model remote source dynamics. Some prior work focuses on using modeling [9] and sampling [11] to detect changes on remote sources. A recent work [14] used survival analysis to model content changes in text databases, and [13] proposed to model insertions, deletions and updates of records directly in relational databases. [18] studied the problem of poll scheduling to monitor the dynamism of Web pages through utility maximization. [15] proposed a general methodology for characterizing the access patterns of Web server requests using a time series analysis method.

1.2 Contributions

The ultimate goal of this paper is to develop a framework for designing efficient repositories for Web applications. The central theme of our approach is to employ statistical methods to predict quality metrics. These prediction quantities can be used to answer questions related to applications, such as: How soon should the local repository be synchronized to have at least 90% precision with certain confidence level? Suppose the local repository was synchronized three days ago, how many cars have been deleted at the remote source since then approximately? And so on.

There are four main contributions in this paper. First, an empirical analysis is conducted on datasets from 6 Web sites in 4 different domains. Our empirical study shows the decay properties of quality metrics at remote sources, and that the decay rate varies from source to source, from category to category, and from time to time. All these observations demonstrate the challenges in modeling Web dynamics.

Our second contribution is a survival-arrival approach for analyzing behaviors of individual objects in remote Web sources (Section 3). The changes in the remote sources are decomposed into two independent processes: arrival and removal. By analyzing both processes, we derive equations for estimating future quality measures.

Our third contribution is an adaptive framework to track the dynamism of the remote sources (Section 4). A time-series analysis is used to model decay properties of quality metrics, and the adaptivity of the framework allows us detect to react to changes in remote sources automatically.

Our fourth contribution is an algorithm for partitioning objects based on attributes (Section 5). The motivation is to improve quality prediction by partitioning objects into groups of similar decay rates. The grouping information can also be used to improve the efficiency of resource allocations.

2 Objects and Quality Metrics

2.1 Data Objects

In this paper, each Web source is viewed as a collection of objects. Each object (e.g., a car) can have multiple attributes such as vehicle ID, make, model, price, etc. The local repository stores the object information retrieved by either following links of Web pages, or posing queries on search forms for deep-Web sources [21]. The remote Web sources have to be re-crawled periodically due to their frequent changes over time.

A critical issue in data management is to assign a unique identifier for each object. We can choose an attribute of the object as a key attribute. For instance, vehicle ID for cars. Many Web sources have a built-in identifier for objects in the internal database. The identifier can be either explicitly associated with the object on an HTML page, or implicitly embedded in the URL for the object. More advanced techniques, such as record linkage [16], could also be used.

There are three possible events during the life-span of an object: *insertion*, *deletion*, and *update*. Note the occurrence of update events may depend on definition of the specific application. For instance, when the price of car changes, it can also be viewed as a deletion of the "old car" followed by an insertion of a "new car." In this paper, update events are *not* viewed to be a combination of deletion and insertion events.

2.2 Quality Metrics

In the literature, several quality measures have been proposed for individual objects, such as age and freshness [7]. Additionally, two quality metrics of importance to describe the remote source as a whole are *precision* and *recall*. As

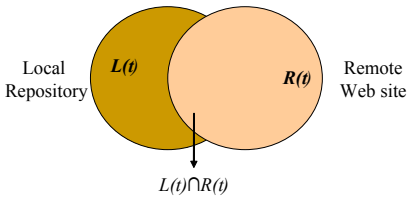


Figure 2. Local and remote objects at time t.

illustrated in Figure 2, let $L(t)$ and $R(t)$ be the set of objects at the local repository and remote Web source at time t respectively. We define

$$\text{precision}(t) = \frac{|R(t) \cap L(t)|}{|L(t)|}; \quad (1)$$

$$\text{recall}(t) = \frac{|R(t) \cap L(t)|}{|R(t)|}. \quad (2)$$

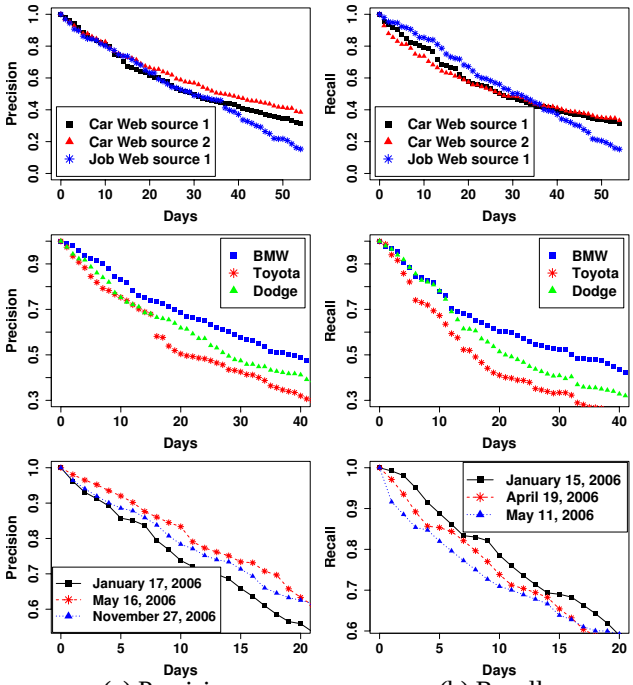
The precision is the proportion of objects in the local repository also exist at the remote source at time t. It is affected by deletions but not by insertions at the remote source, and its value decreases monotonically before the next synchronization. The recall is the proportion of objects at the remote source are in the local repository. It is affected by both insertions and deletions at the remote source. Both precision and recall can also be defined over a subset of objects, such as "BMW cars made after 2002."

It is assumed in this paper that crawls can only take place at some pre-fixed time grids (daily in our case). We also ignore the duration for completing crawls. Our goal is to minimize the number of crawls while ensure that the quality metric of interest is above a certain threshold determined by the application. It is possible to impose more resource constraints on both remote sources or the local repository. Here, we would like to illustrate our main ideas under a reasonably simplified setup.

It is also important to note that the quality metrics above are actually defined with respect to a reference time t_0, i.e., the time of the last synchronization of the local repository. We assume that the local repository is fully synchronized each time, so $L(t) = R(t_0)$ always.

2.3 An Empirical Study

We have conducted a thorough empirical study on real data sets collected data on a daily basis from 6 web sites in 4 domains: cars, job postings, books, and online forums. An overview of all data sources is summarized in Table 1. For each source, its provided query interface is used to retrieve data information. For instance, a typical query for the car domain is "Find Ford Focus cars." For each car Web site, we issued queries on 10 different car models. In order to get as much data as possible, we did not specify any restriction



(a) Precision (b) Recall

Figure 3. Decay of quality metrics. The first row: different Web sources; the second row: different car makes (car Web source 1, synchronization: Apr 28, 2006); the third row: different synchronization dates (car Web source 1).

on the price and year attributes. For all the Web sources, we relied on the URL of each page to extract a unique identifier of the corresponding object.

	Website (# categories)	Period (month)	#objects
Car	Car Web source 1 (10)	10 (2006/01 - 2006/12)	43,597
Car	Car Web source 2 (10)	13 (2005/10 - 2006/11)	37,173
Book	Book Web source 1 (3)	3 (2005/08 - 2005/12)	3,349
Job	Job Web source 1 (2)	3 (2006/12 - 2007/03)	2,959
Job	Job Web source 2 (7)	4 (2006/06 - 2006/10)	31,223
Forum	Forum Web source 1 (4)	2 (2006/12 - 2007/02)	25,370

Table 1. Overview: Web sources.

Figure 3 shows how the precision and recall decay over time. Take the first row of Figure 3 as an example. Assume the repository is synchronized on July 2, 2006 for car Web source 1 and car Web source 2, and December 20, 2006 for job Web source 1 (day 0 in the figure). The figure shows the decay of quality metrics over time. Take the precision for car Web source 2 as an example: its precision dropped to about 80% after 10 days, and about 66% after 20 days. Similar patterns are observed in other datasets as well. The

second row of Figures 3 shows quality decays of three car makes. The quality metrics of Toyota cars decayed faster than BMW. It is due to the different popularities of different car makes. The third row of Figure 3 shows the decay of quality metrics of three different starting dates for the car dataset from car Web source 1. It shows that the recall decayed the fastest for the synchronization date May 11, 2006, while the slowest for the synchronization date Jan 15, 2006.

In summary, our empirical study shows that quality metrics decay over time, and the decay rate varies from Web site to Web site, from category to category, and from time to time. An application needs to take all these facts into the modeling of Web dynamics in order to determine a good synchronize schedule.

3 Modeling Object Behaviors

In this section we state a general framework for applications to derive synchronization schedules. As illustrated in Figure 4, the application crawls Web sources to collect enough data, then analyzes the collected data to derive a model for quality metrics. Based on the model derived, the application can decide the least crawl frequency which achieves certain quality guarantee. The approach is static in the sense that the derived model is not to be changed.

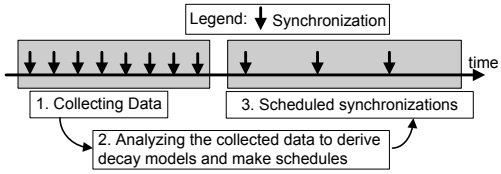


Figure 4. Modeling quality decays statically.

Within this framework, we propose an arrival-survival approach to model the behavior of *individual objects* at remote sources. The survival analysis has already been used in database research community to model the information lifetime [18, 14]. One of our contributions is the introduction of the arrival process: it completes the picture of the whole procedure, and enables us to derive estimation of quality metrics.

One important assumption of our approach is that the arrival is independent of survival process. Such an assumption greatly simplifies the modeling and prediction of quality metrics. From a practical point of view, the dependence between arrival and survival is weak; hence, the independence assumption serves as a reasonable approximation.

There are several advantages associated with this approach. First, understanding behaviors of individual objects can lead to insights into the application. Second, censored information can be naturally incorporated into the survival analysis. Third, it is natural to incorporate attributes of objects (such as car make, etc) in the survival model to improve the estimation accuracy.

3.1 Arrival Analysis

This arrival analysis is to analyze the behavior of arrival counts at remote sources. Figures 5 (the first row) show the histograms of the daily arrival count for several Web sites. Given the shape of the histograms, Gamma distributions are used to model the arrival counts. In the literature, Gamma distributions have been used to model different arrival counts, such as arrival traffic characteristics in distributed or hierarchical Web caching architectures [1]. Its density function is

$$f(x;\lambda,\theta) = \frac{x^{\lambda-1}}{\Gamma(\lambda)}\theta^{-\lambda}e^{-x/\theta}, x > 0,$$

where $\lambda > 0$ is the *shape parameter*, $\theta > 0$ is the *scale parameter*, and $\Gamma(\cdot)$ is the Gamma function. The parameters can be estimated by the maximum likelihood method [19] based on collected data. For our datasets, the fitted density curves are shown against the histograms. The curves show a good fit to raw arrivals, and justify the usage of the Gamma distribution.

It is known that the expected value of a Gamma distribution, i.e., the expected arrival counts, is $\lambda\theta$. Under the assumption that the daily arrival counts are independent, summation of Gamma's is still Gamma distributed. Hence, it is straightforward to compute the distribution of arrival counts during some period by looking up a Gamma table.

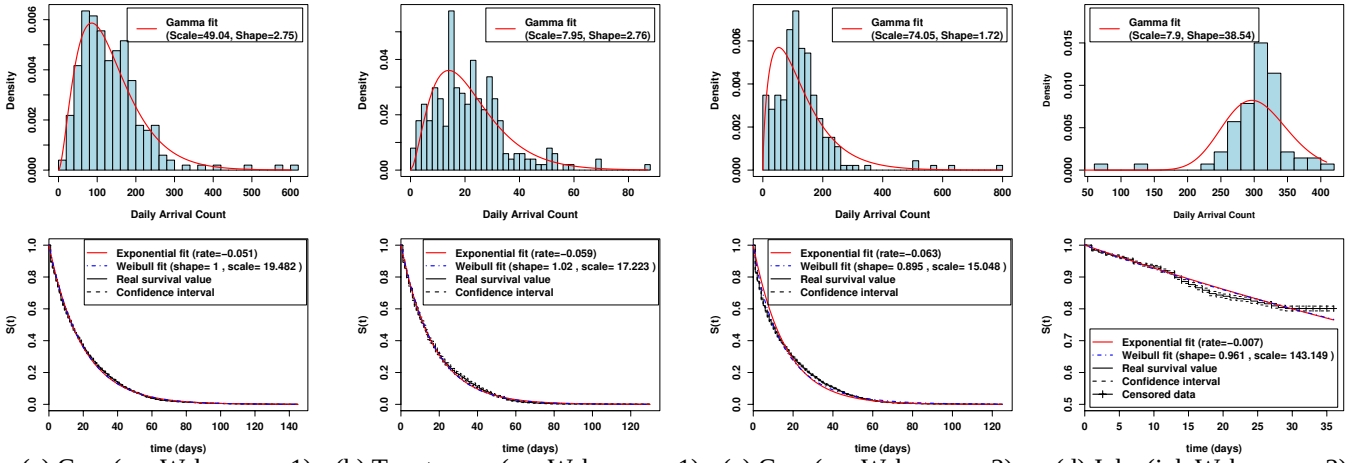
3.2 Survival Analysis

Survival analysis [17] is a statistical approach on studies of the failure events or death of patients. Its idea is to follow objects over time, and study the pattern of failure events of interest. In our study, an object can be viewed as a "patient," and the removal event is a "death" event.

Let T be the life span. The *survival function* $S(t) = P(T > t)$, is the probability that the life span of an object is greater than t. Exponential and Weibull are the two of the most widely used parametric survival functions. The survival function of Weibull is

$$S(t) = e^{-(at)^k},$$

with exponential being a special case ($k = 1$), i.e., Weibull has two parameters while exponential has only one. Both survival functions are fitted with our datasets. The fitted curves are shown in Figures 5 against the empirical ones. The Weibull is slightly better than exponential but the difference is negligible. In other words, the exponential enjoys its simplicity without sacrificing noticeable prediction power. So it is selected as the underlying survival function.



(a) Cars (car Web source 1) (b) Toyota cars (car Web source 1) (c) Cars (car Web source 2) (d) Jobs (job Web source 2)

Figure 5. Arrival/survival distributions.

Given observed data, the maximum likelihood estimate of parameter $\hat{a}$ is simply the inverse of the sample mean. The estimated parameter can be then used to predict the number of objects in a future time.

3.3 Deriving Quality Metrics

Equipped with estimated arrival and survival models, we can estimate the decay of quality metrics over time. Suppose the local repository is synchronized at time t_0. Below, we will derive formula for both precision and recall under the assumption that the arrival and survival processes are independent.

Precision: The precision in (1) can be rewritten as

$$\text{precision}(t) = \frac{|L(t) \cap R(t)|}{|L(t)|} = \frac{|L(t_0) \cap R(t)|}{|L(t_0)|}.$$

The denominator $|L(t_0)|$ is known due to crawling, while the numerator $|L(t_0) \cap R(t)|$ is the number of remaining objects at time t. We can estimate this quantity by its expected value:

$$E(|L(t_0) \cap R(t)|) = |L(t_0)| * e^{-\hat{a}(t-t_0)}. \qquad (3)$$

So we have

$$\widehat{\text{precision}}(t) = \frac{|L(t_0) \cap R(t)|}{|L(t_0)|} = e^{-\hat{a}(t-t_0)}.$$

This result implies that the survival function is equivalent to precision over time. The confidence interval of the precision with confidence probability α is then

$$e^{-\hat{a}(t-t_0)} \pm z_{(1+\alpha)/2} \sqrt{\frac{e^{-\hat{a}\cdot(t-t_0)}(1 - e^{-\hat{a}\cdot(t-t_0)})}{|L(t_0)|}},$$

where $z(\cdot)$ is the standard normal quantile function.

Recall: Similarly, the recall metric at t can be written as

$$\text{recall}(t) = \frac{|L(t) \cap R(t)|}{|R(t)|} = \frac{|L(t_0) \cap R(t)|}{|R(t)|}.$$

Note that both the denominator and numerator of the recall metric depend on the uncertain quantity $R(t)$. We propose to obtain estimates for the denominator and numerator separately, then assemble them together. The estimate for the numerator is already derived in (3). For the denominator, it is more complicated because the quantity is affected by both the arrival and survival processes. Objects in $R(t)$ consist of 1) the objects existing at t_0: they decayed with a constant rate $\hat{\alpha}$; 2) daily arrivals between $[t_0 + 1, t)$. Note that new arrivals also decay with rate $\hat{\alpha}$. In short, we have

$$E\left(|R(t)|\right) = |L(t_0)| * e^{-\hat{a}(t-t_0)} + \sum_{i=t_0+1}^{t} \hat{\lambda}\hat{\theta} * e^{-\hat{a}(t-i)},$$

where $\hat{\lambda}\hat{\theta}$ is the mean daily arrival given by the Gamma distribution. Together, we have

$$\widehat{\text{recall}}(t) = \frac{|L(t_0)| * e^{-\hat{a}(t-t_0)}}{E(|R(t)|)}.$$

A confidence interval for the recall quantity can be obtained by using the delta method [19].

4 Modeling System Adaptivity

The static framework presented in Section 3 is based on an implicit assumption that the observed data can be well

generalized to the future. Such an assumption has its merits in practice, but can be often violated under a dynamic Web environment. For instance, most e-commerce Web sites have much more transactions during Christmas than off-seasons; thus, data from off-seasons might have little predictive power for holiday seasons. Such local dynamism is commonly seen. In order to overcome this issue, an adaptive framework, illustrated in Figure 6, is proposed to automatically adjust to the changes of remote sources. Time-series methods [4] are used as the underlying workhorse of our proposed framework due to the natural time component in our problem.

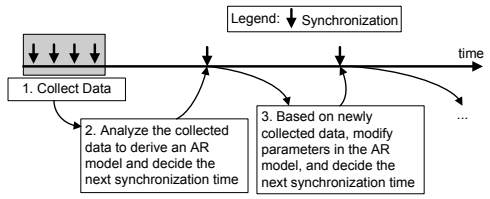


Figure 6. Modeling quality decays adaptively.

Our contributions in the framework are: 1) the introduction of the feedback step. During each synchronization cycle, the parameters of the time series model are updated based on the new information observed through the crawling. An important aspect of the update is to strike a balance between the historic data and the newly observed information; 2) the construction of the time series model. We stress that it is not straightforward to have a time series model on quality metrics. The difficulty comes from the fact that all metrics are implicitly defined using the last synchronization date as a reference. Our solution is to impose a time series model over a closely related quantity, then link it with the quality metric of interest.

Another significant difference between the static framework in Section 3 and the adaptive one is that an *indirect* modeling approach is used by the arrival-survival model, while a *direct* approach by the adaptive framework. In other words, the quality metric is directly modelled by a time series model in the adaptive framework, while it is just a by-product of the modelling of individual objects in the arrival-survival model. There are pro and cons associated with both approaches: a direct approach usually offers better prediction power but lacks the insight into the process itself, while an indirect one generally has better insight but worse prediction power due to the extra complication.

4.1 Time Series Analysis

Assume that crawls can only take place at pre-fixed time grids of the same width, such as one day. Suppose that the local repository is synchronized at time t_0. Let $q(t_0, t)$ be the quality metric (either recall or precision) at time t given there is no synchronization between t_0 and t. Define $d(t)$ be the decay rate of the metric over period $[t-1, t)$:

$$d(t) = \log(q(t-1, t)).$$

The empirical results show that the quality metric is of an exponential form; hence, the quality metric at time t, $q(t_0, t)$, is related to the accumulated decay through

$$q(t_0, t) = \exp\left(\sum_{i=t_0+1}^{t} d(i)\right). \quad (4)$$

Notice the decay rate $d(t)$ is independent of the reference date t_0, so we can impose a time series model on $d(t)$, then the above equation links the decay rate with the quality metric of interest. It also implies that future quality metric can be predicted as long as estimates of $d(t)$ are available.

AR Model. It is of great importance to choose a good time series model in practice even though any model can be used within the general adaptive framework. In this paper, an autoregressive (AR) model [2, 3] is used for the decay rate sequence $d(t)$. Note that the choice of the model is tightly coupled with the application, and should be selected on a case-by-case basis.

Formally, given an *order parameter* k, an AR(k) model can be written as

$$d(t) = \alpha + \sum_{i=1}^{k} \beta_i d(t-i) + \varepsilon_t, \quad (5)$$

where α, β_i's are coefficient parameters, and ε_t's are white noises with variance σ^2. In other words, the quality decay rate at t is a linear combination of the decay rates in the last k periods plus a random error term under an AR(k) model. The AR model is one of the most widely used time series methods due to its simplicity, flexibility, and power. It can be tailored to reflect various properties of the remote sources. For instance, if weekly variation is observed, then we can add a 7th order term, $\beta_7 d(t-7)$, to capture such a weekly trend.

The AR model actually contains a family of time series models. An important question in the AR analysis is to determine the order parameter k based on the observed data. It is called model identification [4] in statistics, and well established procedures exist to choose k based on the observed data. Below are some of model identification results on our datasets: AR(3) model for the precision of car Web source 1 dataset, and an AR(7) for the precision of job Web source 2 dataset.

Given k and the collected data, the maximum likelihood can be applied to obtain parameter estimate $(\hat{\alpha}, \hat{\beta}_i, \hat{\sigma}^2)$. Our goal is to predict future decay rates. Suppose the local repository is synchronized at time t_{-1} and t_0 ($t_{-1} < t_0$,

and no synchronization between them.) The crawled information at t_{-1} and t_0 enables us to interpolate the decay rates $\hat{d}(i)$ for $t_{-1} \leq i \leq t_0$. Then the estimation of $d(t)$ for $t > t_0$ with respective to t_0 can be obtained by recursively applying the following system equation:

$$\hat{d}(t) = \hat{\alpha} + \sum_{i=1}^{k} \hat{\beta}_i \hat{d}(t-i), t > t_0.$$

The estimated decay rates can be plugged in (4) to obtain an estimate of the quality metric $q(t_0, t)$. The confidence interval of $d(t)$ then $q(t_0, t)$ can again be obtained by the delta method.

4.2 Adaptive Autoregressive Model

In this subsection, we present a novel adaptive approach to evolve an AR model based on newly observed information. By doing so, we expect the AR model will slowly adapt to the current dynamics of the remote sources. For simplicity, below we will use a special case of AR(1) model to present the major ideas of our approach. The same idea can be generalized to AR models of any orders.

Suppose the local repository is synchronized at t_0 with parameter $\hat{\alpha}_0, \hat{\beta}_0$, then the model can be written as:

$$d(t) = \hat{\alpha}_0 + \hat{\beta}_0 d(t-1) + \varepsilon_t, \quad t > t_0.$$

Assume another synchronization occurs at time t_1 (no crawls in between). Now we would like to take advantage of the newly observed information to update the parameter estimate. First, we can compute an estimate of the accumulated decays between t_0, t_1 based on the information available at time t_0:

$$l(\hat{\alpha}_0, \hat{\beta}_0) = \sum_{i=t_0+1}^{t_1} \hat{d}(i).$$

Alternatively, the true accumulated decay can be calculated based on the crawled data at both t_0 and t_1: $l(\alpha, \beta)$. In general, these two numbers differ due to the dynamism at the source. If the gap is small, it implies the AR model captures the dynamics of the remote sources well. On the other hand, if the gap is large, it indicates that the dynamics of the remote sources are varying quickly.

The idea is to adapt to such changes on the run. More specifically, the principle of the parameter-updating scheme is: *to update AR parameters such that the estimated decay is pushed towards the observed one.* In order to keep a balance between the historic and the newly observed information, a learning rate parameter τ ($0 < \tau < 1$) is introduced to control the speed of adaptation. Our updating equations can be formulated as:

$$\hat{\alpha}_1 = \hat{\alpha}_0 + \tau \frac{\Delta l}{\Delta(\hat{\alpha}_0)}, \tag{6}$$

$$\hat{\beta}_1 = \hat{\beta}_0 + \tau \frac{\Delta l - \Delta(\hat{\alpha}_0)(\hat{\alpha}_1 - \hat{\alpha}_0)}{\Delta(\hat{\beta}_0)}, \tag{7}$$

where $\Delta l = l(\alpha, \beta) - l(\hat{\alpha}_0, \hat{\beta}_0)$, and $\Delta(\hat{\alpha})$, $\Delta(\hat{\beta})$ are the coefficients of the Taylor expansion of the function $l(\alpha, \beta)$:

$$l(\alpha, \beta) \approx l(\hat{\alpha}_0, \hat{\beta}_0) + \Delta(\hat{\alpha}_0)(\alpha - \hat{\alpha}_0) + \Delta(\hat{\beta}_0)(\beta - \hat{\beta}_0).$$

In practice, the learning rate parameter is set to be a small number $\tau = 0.1$ or 0.2 for a slow adaption. We also find that the result is not very sensitive to this parameter.

5 Object Partitioning

Our discussions in the previous sections treat all objects at the Web source as a single group with a single model. As our empirical results show, objects with different attributes decay in different speeds. It prompts us to study the problem of partitioning objects into sub-groups to improve modeling accuracy. It is also possible to have different schedules for different groups of objects. Our proposed approach consists of two components: 1) quantifying the goodness-of-fit of a partition scheme; 2) searching for good partition schemes using a heuristic-based algorithm. Below, we use arrival-survival analysis as an example to illustrate the grouping approach with the exponential survival model.

5.1 Goodness of Partitioning

Let G be a collection of objects. For simplicity, we consider a scheme that partitions G into two groups: G_A and G_B. It can be generalized to more complicated groupings. Usually, the grouping criteria is based on some auxiliary attributes. For example, G_A is the set of Toyota cars, and G_B the set of all other cars. The goodness of the partition scheme can be formulated as a hypothesis testing problem:

$$H_0\text{: partition } G \quad \text{vs.} \quad H_a\text{: partition } G_A \text{ and } G_B,$$

where H_0, H_a are the null and alternative hypothesis respectively. The Pearson's χ^2 statistic [19] is used to quantify the goodness-of-fit of both hypotheses. The life-span is discretized into k time intervals: $t_1, \ldots, t_k$. The discretization makes sure that there are sufficient objects falling in each interval. For a partition scheme, the χ^2 statistic is defined as:

$$T = \sum_{i=1}^{k} \frac{(E_i - O_i)^2}{E_i}, \tag{8}$$

where O_i is the number of objects with their life-span within the range of t_i, and E_i is the estimated number of

objects in t_i given by the fitted model based on the grouping scheme. This statistic is a measure of the difference between the observed quality and the estimated quantity. The smaller the T, the better the model.

Let T_0 and T_a be Pearson's statistics under the null and alternative hypothesis, respectively. It can be shown that a finer partition scheme always performs better than a coarser scheme with the cost of adding more parameters in the model. In other words, $T_a < T_0$. Hence, their difference, ΔT, can be used to quantify the goodness of H_a over H_0. It is known that

$$\Delta T = T_0 - T_a \sim \chi^2_1.$$

The degree of freedom of the χ^2 distribution is 1 because one additional partition increases the number of parameters by one under the exponential survival model. Their difference indicates the amount of improvement of a finer partition over a coarser one. If the difference is large (e.g., greater than a pre-defined threshold), we accept the finer scheme; otherwise, we stay with the coarser scheme. The probability of making a wrong decision is quantified by the χ^2 distribution.

5.2 Partitioning Algorithm

Given the criterion defined in the previous subsection, a good partitioning scheme is one which greatly reduces the Pearson's statistic. We propose a heuristic-based approach to searching for such partitions based on the fitted results of survival analysis. For simplicity, we illustrate our results on a single categorical attribute, and the same idea can be generalized to other cases as well.

As illustrated in Figure 7, the proposed algorithm works in a top-down manner. A general clustering algorithm, such as k-means, is used to find good clusters of categories with similar decay rates. The hypothesis testing procedure is then used to measure the goodness of the proposed partition scheme. If the scheme improves the quality estimation, we accept it, otherwise the algorithm stops. The sub-routine is recursively called to make finer partitions.

Algorithm: PARTITION
Input: Historically crawled data and partition number n
Output: n partitions
(1) Derive a fitted function for each category of objects;
(2) m = 2;
(3) Set of groups G = the entire collection;
(4) WHILE ($m \leq n$)
(5) Choose a group G_i in G as a candidate to partition with the corresponding subgroups G_{ia} and G_{ib};
(6) Test the goodness ΔT of partitioning G_i to G_{ia} and G_{ib} on the estimation accuracy on all the objects;
(7) IF the ΔT is within a predefined threshold, break
(8) Remove G from G, and add G_{ia} and G_{ib} to G;
(9) $m = m + 1$;

Figure 7. Partition Algorithm.

6 Experiments

6.1 Prediction: Adaptive vs Static

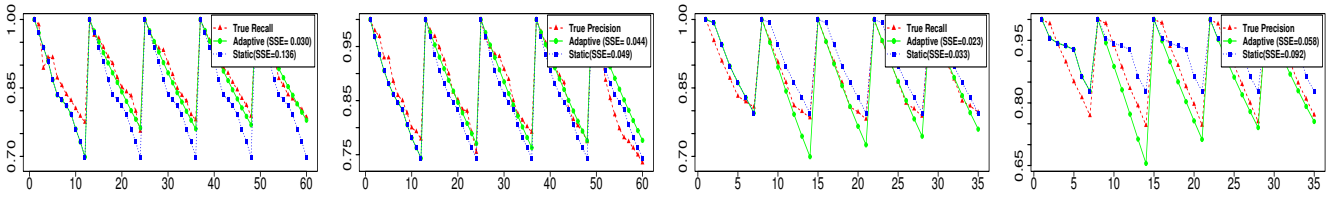
Experiments are conducted to assess the adaptive models in reducing the estimation error of quality metrics. Some experiment results are shown in Figure 8. The training period was April 25, 2006 for 40 days, and the test period was September 25, 2006 to November 24, 2006 for car Web source 1. The training period for job Web source 2 was June 6, 2006 (35 days), and the test period was from August 29, 2006 to October 2, 2006. We crawled car Web source 1 every 12 days and job Web source 2 every 7 days.

Take Figure 8(a) as an example. The remote Web source is crawled every 12 days, and we predict the recall metric at the end of each cycle. The AR model is trained on the data of 40 days starting from April 25, 2006, and the result was tested for a 60-day period from September 25, 2006 to November 24, 2006. The learning rate for updating AR parameters is $\tau = 0.2$. In the plot, the recall curve of adaptive AR model is against the true curve and that of an AR model with static parameters. There is a 78% reduction (from 0.136 to 0.030) in terms of sum of square error (SSE) for the adaptive approach over the static one. The improvement is significant in both relative and absolute scale. From these figures, we can see that the adaptive model yielded more accurate estimates than its static counterpart.

6.2 Resource Allocation

Often the application wants to makes sure that the quality of data in its local repository is greater than a predefined threshold. For example, at any time we want to have at least 90% of cars at the remote source. On the other hand, it is not possible in most cases for the local repository to continuously monitor the changes on the remote source because of limited network resources. We develop a crawling strategy based on our adaptive AR model and empirically verify the improvement.

Our crawling strategy works as follows. Assume that we want the quality of objects in the local repository to be at least a threshold θ. All we need to do is to predict the first time after which the quality of data become less than θ. For example, if $\theta = 0.9$, and we know that the quality of data on the local repository becomes 0.92 after three days, and 0.89 after four days, then we can easily schedule the crawler to synchronize with the remote source every four days in order



Recall (car Web source 1) Precision (car Web source 1) Recall (job Web source 2) Precision (job Web source 2)

Figure 8. Adaptive versus static AR decay models for recall and precision.

to have the required quality. As a result, our main task is to predict such a time using the adaptive AR(k) model.

In order to estimate the goodness of a crawling policy in the above setting, we use the following measure:

$$p = \overline{Q_T}/C,$$

where $\overline{Q_T}$ is the average quality measured over a period of time T, and C is the number of crawls on the local repository. Intuitively if a crawling policy gives us the greater value of p compared to another policy, then the former policy uses fewer synchronization, and has a larger performance value.

Table 2 shows the comparison of two crawling policies (static versus adaptive) for the Web sources we have crawled. The learning rate is $\tau = 0.2$, and the training period consisted of 30 days. The results demonstrate the advantage of using the adaptive policy in resource allocation.

6.3 Partitioning

Table 3 shows the object partitioning result for cars of five different makes from car Web source 2. The estimated decay rates of exponential survival functions is shown first. In the first step, the algorithm partitions the cars into $\langle B, F, H, D\rangle$ and $\langle T\rangle$ – the first letter is used to represent a make. A hypothesis testing shows the goodness testing statistic $\Delta T = 61.34$. This large value shows a large improvement of partitioning into such two groups. Assuming the goodness threshold is $r = 1.5$, the partitioning scheme is then accepted. In step 2, the decay rates are partitioned into three clusters. Again, the partition is accepted, with the goodness $\Delta T = 10.72$. In the third step, the proposed partition is: $\langle B, F\rangle$, $\langle H\rangle$, $\langle D\rangle$ and $\langle T\rangle$. This time, the goodness value $\Delta T = 0.64$, which is less than $r = 1.5$, so the algorithm stops.

The algorithm is also applied to partition cars based on the year attribute, and the results are shown in in Table 4. All these experiments imply that the proposed automatic partitioning algorithm worked well on the real datasets. The algorithm requires no human interactions, and can easily be applied to more complicated partitioning applications.

Make	BMW	Ford	Honda	Dodge	Toyota
Decay rate	-0.058	-0.058	-0.065	-0.069	-0.075

Step	Groups	ΔT
1	{BMW, Ford, Honda, Dodge, Toyota}	
2	{BMW, Ford, Honda},{Dodge, Toyota}	61.34
3	{BMW, Ford}, {Honda, Dodge}, {Toyota}	10.72
4	{BMW, Ford}, {Honda}, {Dodge}, {Toyota}	0.64

Table 3. Partition on car makes (car Web source 2).

Year	2000	2001	2002	2003	2004	2005	2006
Decay	-0.073	-0.07	-0.068	-0.063	-0.058	-0.059	-0.059

Step	Groups	ΔT
1	{2000, 2001, 2002, 2003, 2004, 2005, 2006}	
2	{2000, 2001, 2002}, {2003, 2004, 2005, 2006}	25.0
3	{2000, 2001, 2002}, {2003}, {2004, 2005, 2006}	4.6
4	{2000}, {2001,2002}, {2003}, {2004,2005,2006}	0.9

Table 4. Partition on car years (car Web source 1).

7 Conclusions

In this paper we studied the problem of retrieving structural information from remote sources to build a local repository by modeling changes at remote sources. We conducted an empirical study on real Web sites to gain valuable insights about how dynamic Web objects are, and how the changes affect the quality of the local repository. We proposed a framework, in which we statistically model the behaviors of object insertions and deletions using an arrival analysis and a survival analysis. The analyses help us understand these dynamics, and develop crawling strategies to do synchronizations. We presented an approach based on AR(k) time-series models to adaptively change crawling frequencies in the presence of changes of object behaviors. We also studied how to partition objects into groups with similar decay rates, and use the newly partitioned groups to

Training	Test	Improvement	Domain
Nov 4, 2005	Sept 23, 2006 - Oct 30, 2006	15.98%	Recall, car Web source 2
Nov 1, 2005	Sept 23, 2006 - Oct 30, 2006	1.31%	Precision, car Web source 2
April 25, 2006	Sept 25, 2006 - Nov 14, 2006	9.93%	Precision, car Web source 1
April 25, 2006	Sept 25, 2006 - Nov 14, 2006	10.11%	Recall, car Web source 1
June 27, 2006	Aug 25, 2006 - Oct 1, 2006	10.5 %	Recall, job Web source 2
June 27, 2006	Aug 25, 2006 - Oct 1, 2006	0.13%	Precision, job Web source 2

Table 2. Resource allocation for different Web sources ($\theta = 0.9$ and $\tau = 0.2$).

improve quality estimation. The benefits of the proposed techniques have been shown by our experiments on the real data sets.

References

[1] G. Bai and C. Williamson. Time-domain analysis of web cache filter effects. *Performance Evaluation, Special Issue on Distributed Systems Performance*, 58:285–317, 2004.

[2] G. Box and G. Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990.

[3] P. J. Brockwell and R. A. Davis. *Time series: theory and methods*. Springer-Verlag New York, Inc., New York, NY, USA, 1986.

[4] P. J. Brockwell and R. A. Davis. *Time Series: Theory and Methods*. Springer-Verlag, 1991.

[5] D. Carney, S. Lee, and S. B. Zdonik. Scalable application-aware data freshening. In *ICDE*, pages 481–492, 2003.

[6] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1623–1640, 1999.

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

[8] J. Cho and H. Garcia-Molina. Effective page refresh policies for web crawlers. *ACM Trans. Database Syst.*, 28(4):390–426, 2003.

[9] J. Cho and H. Garcia-Molina. Estimating frequency of change. *ACM Trans. Inter. Tech.*, 3(3):256–290, 2003.

[10] J. Cho, H. García-Molina, and L. Page. Efficient crawling through URL ordering. *Computer Networks and ISDN Systems*, 30(1–7):161–172, 1998.

[11] J. Cho and A. Ntoulas. Effective change detection using sampling, 2002.

[12] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori. Focused crawling using context graphs. In *26th International Conference on Very Large Databases, VLDB 2000*, Cairo, Egypt, 10–14 September 2000.

[13] A. Gal and J. Eckstein. Managing periodically updated data in relational databases: a stochastic modeling approach. *J. ACM*, 48(6):1141–1183, 2001.

[14] P. G. Ipeirotis, A. Ntoulas, J. Cho, and L. Gravano. Modeling and managing content changes in text databases. In *ICDE 2005*, pages 606–617, Washington, DC, USA, 2005. IEEE Computer Society.

[15] A. K. Iyengar, M. S. Squillante, and L. Zhang. Analysis and characterization of large-scale web server access patterns and performance. *World Wide Web*, 2(1-2):85–100, 1999.

[16] N. Koudas, S. Sarawagi, and D. Srivastava. Record linkage: similarity measures and algorithms. In *SIGMOD Tutorial*, pages 802–803, 2005.

[17] E. T. Lee and J. W. Wang. *Statistical Methods for Survival Data Analysis*. Wiley, 2003.

[18] S. Pandey, K. Dhamdhere, and C. Olston. Wic: A general-purpose algorithm for monitoring web information sources. In *VLDB*, pages 360–371, 2004.

[19] J. Rice. *Mathematical Statistics and Data Analysis*. Duxbury Press, 2 edition, 1994.

[20] J. L. Wolf, M. S. Squillante, P. S. Yu, J. Sethuraman, and L. Ozsen. Optimal crawling strategies for web search engines. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 136–147, New York, NY, USA, 2002. ACM Press.

[21] Z. Zhang, B. He, and K. C.-C. Chang. Understanding web query interfaces: Best-effort parsing with hidden syntax. In *SIGMOD Conference*, pages 107–118, 2004.