Int J Digit Libr (2012) 13:33–49
DOI 10.1007/s00799-012-0094-z

Archiving the web using page changes patterns: a case study

Myriam Ben Saad · Stéphane Gançarski

Published online: 6 July 2012

Abstract A pattern is a model or a template used to summarize and describe the behavior (or the trend) of data having generally some recurrent events. Patterns have received a considerable attention in recent years and were widely studied in the data mining field. Various pattern mining approaches have been proposed and used for different applications such as network monitoring, moving object tracking, financial or medical data analysis, scientific data processing, etc. In these different contexts, discovered patterns were useful to detect anomalies, to predict data behavior (or trend) or, more generally, to simplify data processing or to improve system performance. However, to the best of our knowledge, patterns have never been used in the context of Web archiving. Web archiving is the process of continuously collecting and preserving portions of the World Wide Web for future generations. In this paper, we show how patterns of page changes can be useful tools to efficiently archive Websites. We first define our pattern model that describes the *importance* of page changes. Then, we present the strategy used to (i) extract the temporal evolution of page changes, (ii) discover patterns, to (iii) exploit them to improve Web archives. The archive of French public TV channels *France Télévisions* is chosen as a case study to validate our approach. Our experimental evaluation based on real Web pages shows the utility of patterns to improve archive quality and to optimize indexing or storing.

Keywords Web archiving · Importance of page changes · Pattern · Temporal completeness

M. B. Saad (✉) · S. Gançarski
LIP6, University Pierre and Marie Curie, 4 Place Jussieu, 75005 Paris, France
e-mail: myriam.ben-saad@lip6.fr

1 Introduction

Owing to the growing importance of the World Wide Web, many national libraries and organizations (e.g., Internet Archive,[1] Internet Memory Foundation.[2]) have the mission of archiving portions of the Web to prevent useful content from disappearing. The major challenge of such organizations is to collect, preserve, and enable (possibly far) future accesses to a rich part of the Internet content from around the world. Therefore, future generations of users will be able to access over time and browse offline a collection of sites even after they have disappeared from the Web. Web archiving is typically performed by means of Web crawlers. The crawlers periodically harvest the Web and update the archive with fresh images (or versions). However, maintaining a good quality of the archive is not a trivial task because the Web is evolving over time (generating many page versions) and allocated resources are usually limited (storage space, bandwidth, site politeness rules, etc.). The quality of Web archives can be assessed by various measures such as freshness [16], sharpness [18], coherence [39], etc. Among those measures, the completeness appears to be the most relevant. The completeness is the ability of the archive to contain the largest amount of page versions. To maintain a complete archive (containing all versions of every page), Web pages should be captured continuously whenever there is a (small) change in any page. Of course, this is practically infeasible due the limitation of resources and the large number of pages (and sites) to archive. Nonetheless, it is possible to optimize Web crawlers to capture the largest number of pages, and thus the quality of the archive can be significantly improved. For that, archiving systems must predict the behavior of Websites to

1 http://www.archive.org/.

2 http://internetmemory.org/.

be collected to guess when and at which frequency each page must be visited.

Up to now, there are three major factors that have been considered to learn how often each page should be revisited: (i) the importance/relevance of pages (e.g., PageRank, similarity of keywords to user queries, etc.) [17,33,44], (ii) the longevity[3] of information [32] and (iii) the frequency of changes [14,15,18,19]. Another factor that has been ignored so far is the *importance of changes* between pages versions. It is very frequent that crawlers waste time and space to retrieve page versions with unimportant changes (e.g., advertisements, copyright, decoration, navigation bars, etc.). Moreover, the frequency of change used by the main crawlers strategies is estimated based on the homogeneous Poisson process [16]. Unfortunately, in [8,23,36], researchers demonstrate that Poisson model is only valid when the time granularity of changes is longer than one month which is far from being the common case on the Web.

Our work is applied on the archive of the French National Audiovisual Institute (INA), which preserves radio and television channels sites and related pages. These pages are updated frequently (several times per day), and hence the Poisson model cannot be used as explained above. By monitoring several TV channels sites, we have observed a periodicity of changes over days. For some pages, there are more important changes during the day than in the night and during workdays than in weekends. Therefore, we had the idea to discover periodic patterns from pages. A pattern models the evolution of the importance of the page changes over a period of time. We claim that patterns can be a useful tool to predict changes and thus efficiently archive Web pages. Based on patterns, the most significantly changed pages can be visited "at the right moment" optimizing the use of resources (e.g., bandwidth, storage space, etc.). Patterns have received considerable attention and were used in various domains such as Web mining [4,35,41], networking [37,43], systems [27,28,42,45], etc. They were generally used to predict trends, to simplify data processing, and to improve system performance. However, as far as we know, patterns have never been used in the context of Web archiving. In this paper, we demonstrate through a case study the utility of patterns to improve the effectiveness of the Web archives. The archive of French TV channels "*France Télévisions*" has been chosen, as a case study, to validate our approach. We first present different steps and algorithms used to discover patterns from collected Web pages. Then, we show how discovered patterns can be used to (i) efficiently index or store Web pages and (ii) to optimize Web crawlers to improve archives quality. The major contributions of this paper can be summarized as follows:

– A definition of a quality measure (temporal completeness) to assess the quality of the archive. This measure considers the importance of archived versions.
– A definition of a pattern model that represents the evolution of pages changes over accurate period of time (fine granularity).
– A description of the different algorithms used, through a case study, to discover patterns from pages. Some observations and trends are also reported.
– A pattern-based strategy to optimize (i) archives quality and (ii) pages indexing/storing.
– Experimental results that show the usefulness and the effectiveness of our pattern based approach.

The rest of the paper is organized as follows. Section 2 discusses related work on pattern mining and on Web archiving. Section 3 defines the temporal completeness measure to assess the quality of archives. Section 4 formally defines our pattern model for Web pages. In Sect. 5, an overview of the major benefits of patterns for archives is given. Section 6 presents the different steps followed to discover patterns from Web pages. Section 7 describes a case study where patterns are discovered from French TV channels pages. An approach based on patterns to optimize indexing and storing is proposed in Sect. 8. Section 9 shows how patterns can be used to optimize Web crawling and to improve data quality. In Sect. 10, experiments over simulated and real Web pages are presented and discussed. The obtained results demonstrate the effectiveness of our approach. Finally, Sect. 11 concludes.

2 Related works

As mentioned in the introduction, our work is related to pattern mining and Web archiving. We present below related works in those two areas.

2.1 Pattern mining

There are a lot of research, publications, and applications where patterns are involved mainly in the data mining area. Patterns mining is widely used for different applications such as trajectories of objects, weather forecasts, DNA sequences, stock market analysis, etc. It is impossible to give here a complete coverage on this topic but interested readers can refer to [24] for example. The intent of this section is not to describe all available algorithms for patterns discovery. We rather present here some applications, in various domains, where patterns are used to predict trends to simplify data processing and to improve system performance. Then, a brief overview of major summarization techniques used to discover patterns is given.

[3] Longevity [32] is the lifetimes of content fragments that appear and disappear from Web pages over time.

In [35], frequent patterns are discovered from significant intervals of Web log data and are used to forecast the user behavior. In [43], Risto Vaarandi detects frequent patterns form event log files to help the creation of system profile and the detection of anomalies. In the context of network analysis, Sia et al. [37] propose a RSS monitoring algorithm, using user access patterns, to efficiently allocate resources and to provide subscribers with fast new alerts. Recent studies are closely related to our work, in the sense that they aim at predicting and modeling expected Web changes based on patterns. In [2], Adar et al. use patterns to describe people's revisitation behavior to understand the relationship between the Web changes and the visitation. However, this approach is limited to Web pages with enough knowledge about visitors (e.g., the number of visitors, the time between a visit, and revisits, etc.). In [6], Bogen et al. conclude that the content of blogs have regular patterns of changes. In [7], they identify unexpected patterns of changes using Kalman filters to model the behavior of blogs over time. However, using Kalman filters assumes the linearity of the systems which is not always the case for Web page changes. Despite the diversity of applications using patterns, we notice that patterns have not been exploited in the context of Web archiving to predict page changes.

In general, the methods and algorithms used to discover patterns are developed from several fields such as statistics, data mining, machine learning, and pattern recognition. These methods depend on the data type to be treated such as set, sequence, time series, tree, graph, etc. Patterns discovery is based on summarization techniques such as: statistical analysis, association rule, clustering, classification, etc. Statistical analysis are the common methods to give statistical description (trend and periodicity) of pattern based on frequency, mean, median, etc., like in [37]. Association rules are used to discover interesting relations between objects, events, etc., such as in [4]. Clustering, like in [43], is a technique to group together a set of data having similar characteristics. Classification, as used in [13], is the process of mapping a data into one of predefined classes.

To sum up, patterns are widely used in different fields. Depending on the dataset, many pattern summarization techniques have been proposed, each with different purpose. However, to the best of our knowledge, patterns have never been exploited to optimize Web archiving. In this paper, we show how patterns can be discovered from pages changes and used to improve the quality of Web archives. Most of existing algorithms search for frequent patterns without any prior knowledge about the form of pattern to be discovered. Here, however, we have an idea about the format of the periodic pattern to be discovered. Our idea is inspired from a long period of monitoring observations. We noticed that time series extracted from the same pages have generally similar change evolution over days. As the patterns we want to discover have a simple format, a very simple statistical analysis (i.e., based on changes average) can be used to extract periodic patterns from Web pages.

2.2 Web archiving

In recent years, there have been various studies about Web archiving. An overview of the main issues involved by Web archiving is given by Masanès [29]. The researches that have been proposed address different issues of Web archiving. The specification of the Web perimeter was studied in [1,26]. In [12,22], authors address issues concerning the format of information to be stored or indexed in the archive. A technique to extract data objects based on Web feeds is proposed in [31] to improve page change detection. Abiteboul et al. [1] propose a representation of changes at site level (site-delta) for an efficient use of bandwidth and storage space. To improve the freshness of Web pages, efficient crawl strategies were proposed in [14,15] based on a page changes frequency [16]. Brewington and Cybenko [9] estimate how often Web sites must be re-indexed for more fresher indexes. A re-crawl scheduling strategy based on information longevity is proposed in [32] to optimize the freshness of Web pages. For a better quality of the archive, Cho et al. [17] propose a crawl strategy to download the most important pages first. The importance of pages is defined based on different ways: similarity between pages and queries, number of backlinks over the entire site, rank of a page, etc. Similarly, Castillo et al. [11] downloaded the best ranked pages. Recent studies address the issue of the temporal coherence of a site crawl. Spaniol et al. [39] introduce the notion of temporal coherence and propose a crawling strategy to optimize Web sites coherence. In [40], visualization strategies were proposed to help the archivist in understanding the nature of coherence defects. In other studies [18,30], a framework (SHARC) and a system (SOLAR) were implemented to maximize the sharpness of Web archives. Sharpness is a quality measure proposed by authors to reflect how much the site's pages have been changed during the crawl.

All those archiving approaches address the challenge: efficiency [1,12,22], freshness [9,14,15,32], data quality [17,18,30,39,40]. Most of Web crawler strategies download the most important pages (based on PageRank, etc.) and/or retrieve the most frequently changing pages (based on homogeneous Poisson model). However, the importance of changes between page versions has been ignored so far. Moreover, the homogeneous Poisson model used to estimate changes frequency is not valid for page modified very frequently as in our case study. Such models based on a fixed frequency of changes are enabled to predict the period of time accurately where important changes is expected to occur. Similar to our work, Adar et al. [3] propose models and analysis to characterize the amount of change on the Web at finer grain (hourly updates), but they do not propose a method to estimate the importance of changes detected between pages versions. According to them, their change analysis can be used to refine crawler policies by focusing on meaningful changes, but no strategy has been proposed yet. Francisco-Revilla et al. [21] conduct a study to understand how users perceive changes on Web pages and what type of changes (content, presentation and structural) is considered relevant. In [20], they propose a tool which evaluates and informs users when relevant changes occur on Web pages. However, no model has been proposed to describe and predict the relevance of changes over time. In addition, the tool of [20] cannot evaluate the magnitude of changes for pages with active elements like JavaScripts. In this paper, we focus on dynamic Web pages like radio and TV channels pages which have active components (e.g., applet, plug-ins, script, etc).

Though interesting, none of the existing archiving approaches (or models), as far as we know, have used both (i) the patterns and (ii) the importance of changes to improve the quality of archives. By these two concepts for an optimized Web archiving is the core of this paper. We present a case study where patterns are discovered from French TV channels Web pages. Then, we demonstrate how patterns can be useful to improve archives quality and pages indexing.

3 Temporal completeness

The temporal completeness measures the ability of the archive to contain the largest number of important versions. The notion of the completeness is very useful as it is quite common that users cannot access certain versions of Web pages while navigating in the archive. These versions are usually missing because they have disappeared from the Web before they were captured by the crawlers. An archive is considered complete if it contains all the versions of the pages that have appeared on the Web (created by any small change). From a practical point of view, it is impossible to have a perfectly complete archive because resources are usually limited (bandwidth, storage space, etc.) and Web pages change frequently. Therefore, it is important to have a measure that assesses efficiently the temporal completeness of archives. In this work, we focus on archiving initiatives that hold collections of Web pages with a reasonable size, like INA, which preserves radio and television Web pages. In this case, the temporal completeness makes more sense than for other archiving organizations (e.g., Internet Archive) that focus on spatial completeness (horizontal and vertical)[4] by quantifying the number of new discovered resources that have not been archived yet. Unlike spatial completeness, the temporal completeness quantify the number of new important page versions that have been captured by preserving the history of their temporal changes.

In the following, we formally define the temporal completeness measure.

[4] The vertical completeness quantifies the number of new entry points that have been discovered from the Web, whereas the horizontal completeness quantifies the number linked nodes founded for each entry point.

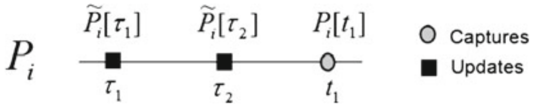


Fig. 1 Example of page versions

3.1 Notations

We assume that $P = \{P_1, P_2, \ldots, P_i, ..P_n\}$ is the list of pages (or Urls) to be crawled from the Web. A version of the page is a capture (or a snapshot) obtained through downloading the page from the Web. We note the version captured from the page P_i at time t by $P_i[t]$. A change that occur on the page P_i at time t is denoted by $\mu_i[t]$. We assume that the importance of the page P$_i$ (e.g., PageRank) is $\omega(P_i)$. The importance of a change $\mu_i[t]$ occurred on the page P_i at time t is denoted by $\omega(\mu_i[t])$.

We note $\tilde{P}_i[\tau]$ the version of the page P_i that have appeared on the Web (created by a change) at time τ. As shown in the Fig. 1, the page P_i has one capture at time t_1 that corresponds to the archived version $P_i[t_1]$. The two versions $\tilde{P}_i[\tau_1]$ and $\tilde{P}_i[\tau_2]$ that have appeared on the Web correspond to the two updates occurred on the page P_i at time τ_1, τ_2. Note that $P_i[t_1] = \tilde{P}_i[\tau_2]$.

3.2 Page version importance

Given a version of a Web page, we define its importance by combining the two following concepts:

1. The importance or the relevance of the pages (e.g., PageRank, similarity of keywords to a user query, etc.)
2. The importance of changes between the current version and the last archived one of the same page. The importance of changes between two page versions is estimated based on the function explained in Sect. 6.3. This function evaluates the importance of changes based on the rate of modifications (e.g., the percentage of deleted images or inserted links, etc.) that occur in the most relevant blocks of the page. It returns a normalized value between 0 and 1. An important value near one (respectively near 0) denotes that changes between the versions are very important (respectively useless e.g., advertisements, decoration, etc.).

Definition 1 (*Version importance*)
Given a version $P_i[t]$ captured from the page P_i, the importance $\omega(P_i[t])$ of the version is the multiplication of the importance of the page $\omega(P_i)$ by the importance of the changes occurred between this current version and the last archived one of the same page.

$$\omega(P_i[t]) = \omega(P_i) * ImpCh(\Delta)$$

where Δ is a delta file saving all changes that occurred between the current version of $P_i[t]$ and the last archived one of the same page.

By introducing this concept of the importance, the strategy of Web crawlers can be optimized to download in priority the most relevant pages that have the most important changes since the last archived version.

3.3 Temporal completeness measure

The temporal completeness assesses the proportion of the importance of the pages versions that have been captured with respect to the total importance of the versions that have appeared on the Web.

Definition 2 (*Complete archive*)
An archive A is complete if it contains all the page versions $\tilde{P}_i[t]$ that have appeared on the portion X of the Web to be captured.

$$\forall \tilde{P}_i[t] \in X, \forall t, \exists t' : P_i[t'] \in A \; et \; P_i[t'] = \tilde{P}_i[t]$$

Definition 3 (*Page temporal completeness*)
The temporal completeness of an archived page P_i is the sum of the weights of versions that have been captured, divided by the total weight of the versions (created by changes) that appear on the Web. Duplicated versions that have been already captured are not quantified by the measure. Let m be the number of versions $P_i[t_k]$ captured at time t_k and p be the number of versions $\tilde{P}_i[\tau_k]$ appeared on the Web at time τ_k, the weighted completeness of archived page P_i is

$$WCompleteness(P_i) = \frac{\sum_{k=1}^{m} \omega(P_i[t_k])}{\sum_{k=1}^{p} \omega(\tilde{P}_i[\tau_k])}$$

where the weight of the version $\omega(P_i[t_k])$ is equal to the last change importance $\omega(\mu_i[t'])$, the weight of the version $\omega(\tilde{P}_i[\tau_k])$ is equal to the last change importance $\omega(\mu_i[\tau'])$, and $\omega(\mu_i[t'])$ and $\omega(\mu_i[\tau'])$ denote the importance of the changes that occurred respectively at t' and τ' just before the capture of the version $P_i[t_k]$ and the creation of the version $\tilde{P}_i[\tau_k]$.

As we measure here the temporal completeness for a same page, the importance of the pages has not been considered in the weight importance of each version. It is included later while computing the temporal completeness of the whole archive.

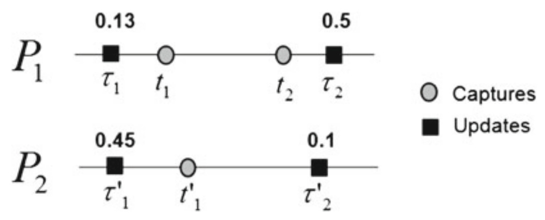


Fig. 2 Completeness measure example

Definition 4 (*Archive temporal completeness*)
The overall completeness of the archive A is the average completeness of all pages $P_i \in A$.

$$WCompleteness(A) = \frac{\sum_{i=1}^{n} WCompleteness(P_i) * \omega(P_i)}{\sum_{i=1}^{n} \omega(P_i)}$$

where $\omega(P_i)$ is the importance of the page P_i.

Example 1 (Archive temporal completeness)
We consider an archive A consisting of two pages P_1 and P_2 as shown in Fig. 2. We assume that $P_1[t_1]$ and $P_1[t_2]$ are the two versions of the page P_1 captured respectively at time t_1 and t_2. We assume that the importance of the two changes that have occurred on the page P_1, at time τ_1 and τ_2 are 0.13 and 0.5, respectively. For the page P_2, there is one capture $P_2[t_1^{'}]$ at $t_1^{'}$. The importance of the two changes occurred on the page P_2 at $\tau_1^{'}$ and $\tau_2^{'}$ are respectively 0.45 and 0.1.

Note that in this example, we assume that the importance of pages P_1 and P_2 is equal to 1.

The weighted completeness (WC) of P_1 and P_2 is

$$WC(P_1) = \frac{\omega(P_1[t_1]))}{\omega(P_1[\tau_1]) + \omega(P_1[\tau_2])} = \frac{0.13}{0.13 + 0.5} = 0.2$$

$$WC(P_2) = \frac{\omega(P_2[t_1^{'}])}{\omega(P_2[\tau_1^{'}]) + \omega(P_2[\tau_2^{'}])} = \frac{0.45}{0.1 + 0.45} = 0.81$$

The overall completeness of the archive A is

$$WC(A) = \frac{WC(P_1) * \omega(P_1) + WC(P_2) * \omega(P_2)}{\omega(P_1) + \omega(P_2)}$$

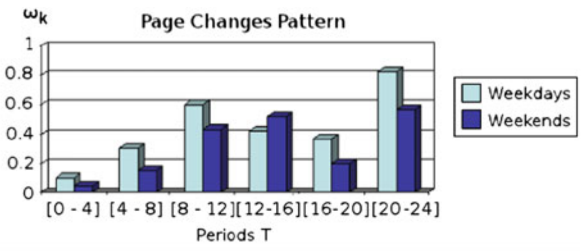
$$WC(A) = \frac{0.2 * 1 + 0.81 * 1}{2} = 0.50$$

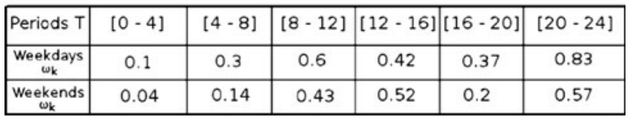
4 Pattern-based model

In this section, we introduce our pattern-based model to describe and predict pages changes behaviors.

4.1 Page changes pattern

A pattern models the evolution of page changes over a period of time (e.g., a day) as shown in Fig. 3. It represents the





Periods T	[0 - 4]	[4 - 8]	[8 - 12]	[12 - 16]	[16 - 20]	[20 - 24]
Weekdays ω_k	0.1	0.3	0.6	0.42	0.37	0.83
Weekends ω_k	0.04	0.14	0.43	0.52	0.2	0.57

Fig. 3 Pattern example

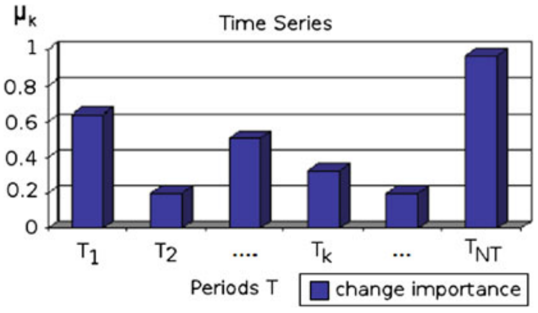


Fig. 4 Time series

changes rate (or *the importance of changes*) over time. It is periodic and may depend on the day of the week and of the hour within a day. Separate patterns can be defined for weekends. We formalize below the definition of pattern based on the importance of changes.

Definition 5 (*Pattern*)
A pattern of a page P_i with an interval length l is a nonempty sequence $Patt(P_i) = \{(\omega_1, T_1); \ldots; (\omega_k, T_k); \ldots; (\omega_{N_T}, T_{N_T})\}$, where N_T is the total number of periods in the pattern and ω_k is the average of the importance of changes estimated in the period T_k. The sum of the time periods, $\sum_{k=1}^{N_T} T_k$, is equal to l. We choose $l = 1$ day as the length of the pattern in our case study.

Example 2 (Pattern)
Let $Patt(P_1)$ = (0.1, [0–4 h]); (0.3, [4–8 h]); (0.6, [8–12 h]); (0.42, [12–16 h]); (0.37, [16–20 h]); (0.83, [20–24 h]) be a periodic pattern of the page P_1 for weekdays as shown in Fig. 3. The values 0.1, 0.3, 0,6, 0.42, 0.37, and 0.83 are respectively the average of the importance of changes for each interval period T_1 = [0–4 h],. . . , T_6 = [20–24 h]. Also, a periodic pattern can be defined separately for weekends as shown in the Fig. 3.

4.2 Time series

Patterns are discovered from a collection of the time series obtained by crawling the page periodically during a long period of time. A time series represents the evolution of the change importance of a specific page during a day.

Definition 6 (*Time series*)
A time series (Fig. 4) of a page P_i with an interval length $l = 1$ day is a nonempty sequence $TSeries(\mathrm{P}_i) = \{(\mu_1, T_1); \ldots; (\mu_k, T_k); \ldots; (\mu_{N_T}, T_{N_T})\}$, where μ_k is the importance of changes detected between two versions captured at the beginning and the end of the period T_k.

5 Pattern benefits

Our model based on patterns can improve the effectiveness of Web archives at the two following points: (i) optimized pages indexing/storing and (ii) improved quality of archives by efficient crawling.

5.1 Optimized storing or indexing

Owing to the limitation of resources such as storage space, bandwidth, site politeness rules, etc., Web archive systems must avoid wasting time and space for storing/indexing some versions with unimportant changes. Based on patterns, one can predict for each page, the period of time when there are important changes on pages. An adequate change importance threshold can be fixed from patterns to decide when it is appropriate to index or store each page. Hence, the optimized index enables a fast and an accurate page retrieval from the archive. The speed and the performance in finding relevant/important Web pages for a search query will be significantly improved.

5.2 Improved archive quality

The role of Web crawlers is to decide what page should be refreshed/archived and with which priority. Based on patterns, Web crawling can be optimized in such a way that it improves the quality of the archive. In this paper, we focus on the completeness as the quality measure. Completeness is the ability of the archive to contain the largest amount of useful page versions. Driven by patterns, the crawler attempts to retrieve the most important versions to maintain the archive as complete as possible. An important version is a relevant version of a page that has important changes since the last one archived. Hence, unimportant changes in the page (e.g., copyrights, advertisements, decoration, etc.) can be ignored and useful information is captured by a single crawl, maximizing the use of resources.

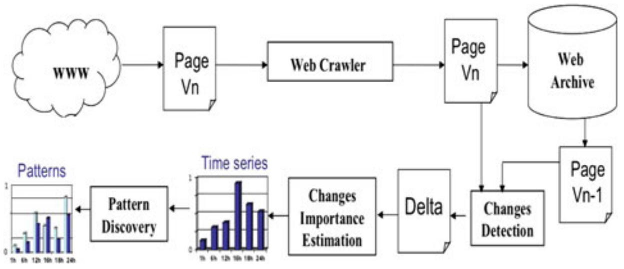


Fig. 5 Pattern discovery framework

6 Pattern discovery phases

In this section, we present briefly the major phases used to discover patterns from page changes.

In order to discover patterns, page changes are observed and monitored for a long period of time. The framework as shown in Fig. 5 illustrates the different phases needed to discover pattern from Web pages. It consists of five major modules: The *crawler* periodically downloads Web pages; the *Web archive* stores and indexes the different versions of pages retrieved by the crawler; the *change detection* module generates delta files describing changes that have occurred between successive page versions; the *change importance estimator* evaluates the importance of changes and represents them by time series; and the *pattern discovery* module mines patterns from extracted time series. More details of pattern discovery phases are given below.

6.1 Pages crawling

The crawler harvests the Web by iteratively downloading new versions of pages. The frequency of the crawler is chosen not to be too long (e.g. hourly crawl) to avoid missing relevant modifications. Each retrieved version is then stored and indexed in the archive.

6.2 Change detection

In this phase, captured versions of pages are rendered and partitioned into multiple blocks by means of an extended visual segmentation algorithm VIPS [10,34]. The advantage of rendering the page is to preserve its visual aspect even if there are some embedded scripts like JavaScripts. The segmentation enables to have a complete hierarchical structure of the page in the same way that it is perceived by users.

Then, each current version of page is compared with the previously downloaded one based on the change detection algorithm Vi-DIFF [34]. Vi-DIFF detects structural and content changes that simulate how a user understands changes based on his visual perception. Structural changes (e.g. delete, insert, etc.) alter the visual appearance of the page and the structure of its blocks, whereas content changes modify text, links, and images inside page blocks. A delta file describing all these changes is generated as output. For more details about the Vi-DIFF algorithm, please refer to [34]. As shown in the example of the Fig. 6, the two page versions are visually segmented on five leaf blocks B_1, B_2, $B_{3.1}$, $B_{3.2}$ and $B_{3.3}$. Content changes i.e., two deleted links and one updated image are detected in the block $B_{3.3}$ between the two versions $V(n-1)$ and $V(n)$ of the page.

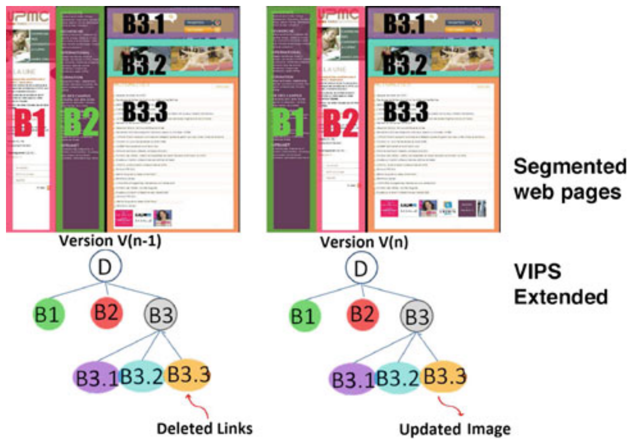


Fig. 6 Visual change detection

6.3 Change importance estimation

In this phase, the successive delta files generated during a day are evaluated by the following function [5] to estimate the importance of changes. This function depends of three major parameters:

- The *importance of each block I(B)* composing the page. It can be evaluated by the method proposed in [38] that exploits spatial (position, area size) and content (links number, etc.) information to assign automatically importance weights to blocks,
- The *percentage of changes PerCh(Op,El,B)* of operations (insert, delete, etc.) occurred on each block,
- The *importance of change operations I(Op)*. For example, insert can be considered more important than a delete.

The importance of changes $ImpCh(\Delta)$

$$= \sum_{i=1}^{N_B} I(B_i) * \left[\frac{1}{N_{Op}} \sum_{j=1}^{N_{Op}} I(Op_j) \right. \\ \left. * \frac{1}{N_{El}} \sum_{k=1}^{N_{El}} PerCh(Op_j, El_k, B_i) \right]$$

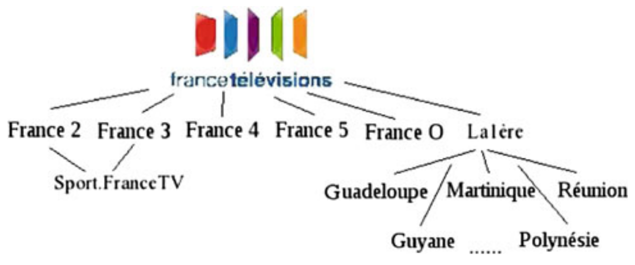


Fig. 7 *FranceTV* Web sites

where the percentage of changes inside each block is

$$PerCh(Op_j, El_k, B_i) = \frac{N(Op_j, El_k)}{N(El_k, B_i)}$$

- N_{Op}, N_B are respectively the number of operations and the number of blocks.
- $Op \in \{insert, delete, update\}$
- $El \in \{link, image, text\}$
- $\sum_{i=1}^{N_B} I(B_i) = 1$
- $N(Op_j, El_k)$ is the number of operations Op_j that modify the element El_k.
- $N(El_k, B_i)$ is the total number of elements El_k inside the block B_i

The larger the amount of changes inside important blocks is, the higher will be the estimated importance of changes.

Then, for each page, a time series (Fig. 4) that represents the evolution of the importance changes during a day is built. The generation of time series is detailed in the case study of the Sect. 7.

Example 3 (Changes importance estimation)
Given the example of the Fig. 6, the change operations detected in the delta between the two versions $V(n-1)$ and $V(n)$ are (i) a deletion of two links and (ii) an update of one image in the block $B_{3.3}$. We assume in this example that the operation update is considered more important ($I(update) = 1$) than a delete ($I(delete) = 0.8$). The importance of the block $B_{3.3}$ is equal to 0.3. We assume also that the block $B_{3.3}$ of the version $V(n-1)$ has 1 image and 10 links (Fig. 7).

The importance of changes is computed as follows:

$$ImpCh(\Delta) = I(B_{3.3}) * \left[I(delete) * \frac{N(delete, links)}{N(links, B_{3.3})} + I(update) * \frac{N(update, image)}{N(image, B_{3.3})} \right]$$

$$ImpCh(\Delta) = 0.3 * [0.8 * (2/10) + 1 * (1/1)] = 0.34$$

6.4 Pattern discovery

In this step, the set of time series is analyzed to extract periodic patterns for each page based on statistical analysis (i.e., changes average). Patterns are discovered by averaging the importance of changes of time series collected during a long period. Some periodic patterns are defined separately for workdays and weekends. Different pages (of the same site or of different sites) with similar behavior of changes can be grouped to share a common pattern. During an online crawl, patterns should be updated periodically to always reflect the current behavior of Web pages. Based on discovered patterns, the evolution of page changes can be predicted over accurate periods of time and exploited to optimize Web archiving. More details of this phase are given in the following case study.

7 Case study: France TV archive

We choose the *France Télévisions* (*FranceTV*) archive as a case study to experiment our pattern-based approach. We claim that the *FranceTV* site is rich enough to reflect the evolution of changes of major radio and TV channels of the French Web. We have monitored, over a period of one month, about one thousand pages composing the *FranceTV* site. Each page is crawled every hour, every day. We first present the characteristics of this site. Then, we describe algorithms used to extract time series and discover patterns from collected pages versions. Some observations and trends deducted from discovered patterns are finally reported.

7.1 Site characteristics

France Télévisions (Fig. 7) is the French public national television broadcaster. It includes the TV channels: *France 2*, *France 3*, *France 4*, *France 5*, *France 0*, and *Sport.FranceTV*. It also includes *La 1 ère* which is the network of radio and television channels operating in France's overseas departments and territories around the world (e.g., Réunion, Martinque, etc.). The *FranceTV* site contains vast and a various amounts of Web pages that have different changes behaviors. There are dynamic pages that change very frequently (several times per day) like news pages. There are also (few) quasi-static pages that change, for example, only twice a week. For some pages, there are significantly higher changes during the day than during the night. Owing to the time zone difference between France and its overseas departments, the periods of work activities are different and this affects the behavior of page changes. Hence, the *FranceTV* archive is chosen as a case study due to the diversity of its Web pages.

7.2 Time series acquisition

As mentioned before, pages of *FranceTV* sites were crawled every hour and saved as a new version. The collected versions of the same page, over a day, are grouped and analyzed to extract time series.

7.2.1 Principle

The acquisition phase of time series from pages changes combines two steps, changes detection and change importance estimation, as presented in Sect. 4. The first step consists in detecting changes between page versions ($v^i_{\eta-1}, v^i_\eta$) by means of Vi-DIFF algorithm (cf. Sect. 6.2) to generate delta files. Then, the changes, described in deltas, are evaluated and used to extract time series for each page every day.

In our case study, the importance of each block was manually estimated. The percentage of changes occurred in each block is computed from the delta file Δ. It represents the amount of change operations detected for each block divided by the total number of elements in the block (text, link, etc.). The importance of operations depends on both the operation type (delete, insert, etc.) and on the type of element (link, image, etc.) affected by the change. In the rest of the paper, we note the importance of an operation *Op* over an element *El* by *I(Op,El)*. Furthermore, we would like to give less importance for an advertisement link or image. An advertisement link (or image) has usually its address (or name) that includes the term "advertisement." If it is not the case, more sophisticated methods based on machine learning algorithms [25] can be used to recognize advertisement content. Also, we want to consider the distance between compared texts (of two versions) when computing the importance of changes. Thus, the following requirements have been introduced to estimate the importance of an operation *I(Op,El)*.

- $I(Op, Text) = distance(Text(v_{n-1}), Text(v_n))$
- $I(Op, El) = \begin{cases} \alpha & \text{if} \quad El\ is\ Advertisement \\ 1 & \text{else} \end{cases}$

  where $El = \{link, image\}$ and $Op = \{insert, delete, update\}$
- The textual distance computes the proportion of distinct words between the two compared texts.
- If an advertisement element (link or image) is modified, a low value α is given to the importance of an operation (e.g., $\alpha = 0.1$). Otherwise, the importance of an operation $I(Op, El)$ is equal to 1.

Note here, we have assigned a low importance weight to advertisement elements (link, image, etc.). However, the same process can be done for others contents that are considered useless or not relevant for users.

7.2.2 Time series acquisition algorithm

We use the Algorithm 1 to extract time series from collected page versions. A new time series is created every day for

Algorithm 1 Time series acquisition

```
Input:
P_1, P_2, ..., P_n - list of pages
$v^i_1, v^i_2, \ldots, v^i_{N_T}$ - versions of page $P_i$
Output:
TSeriesList - time series of pages
Begin
1. for each page $P_i$ i = 1 ..., $n$ do
2. TSeries = newTimeSeries()
3.      for each version $v^i_k$ of period $T_k$ k = 1 ..., $N_T$ do
4.           $\Delta$ = DetectChanges($v^i_{k-1}, v^i_k$)
5.           $\mu$ = EstimateChangeImportance($\Delta$)
6.           TSeries.add($\mu$, $T_k$)
7.      end for
8. TSeriesList.add($P_i$,TSeries,date)
9. end for
10. Return TSeriesList
End
```

each page. At each period (line 3–6), the changes between successive versions of the same page are detected, evaluated and then added to time series. The change detection (*detectChanges*() in line 4) is done by calling the Vi-DIFF algorithm (cf. Sect. 4). A pseudo code of the function that estimates the importance of change in the delta (line 5) is depicted by the Algorithm 2. The importance of changes, as shown in the Algorithm 2, is computed by averaging the percentage of changes in each block (line 5), the importance of change operation (line 6–11) and the importance of block (line 16). At the end of the Algorithm 1, a collection of time series describing the change importance of each page is obtained for a given date (line 8–10).

Algorithm 2 Change importance estimation

```
Input:
$\Delta$ - delta file
Output:
importance - estimated changes importance
Begin
1. for each block B in $\Delta$ do
2.      ImpB = getBlockImportance()
3.      sum = 0
4.      for each operation Op over an element El $\in$ B do
5.           PerCh = getPercentageChanges(Op,El,B)
6.           if (El is Text) then
7.                ImpOp = distance(Text($v_{n-1}$),Text($v_n$))
8.           else if (El is advertisement) then
9.                ImpOp= $\alpha$
10.          else
11.               ImpOp = 1
12.          end if
13.          end if
14.          sum = sum+ImpOp*PerCh
15.     end for
16. importance = importance = (sum/$N_{op}$)*ImpB
17. end for
18. Return importance
End
```

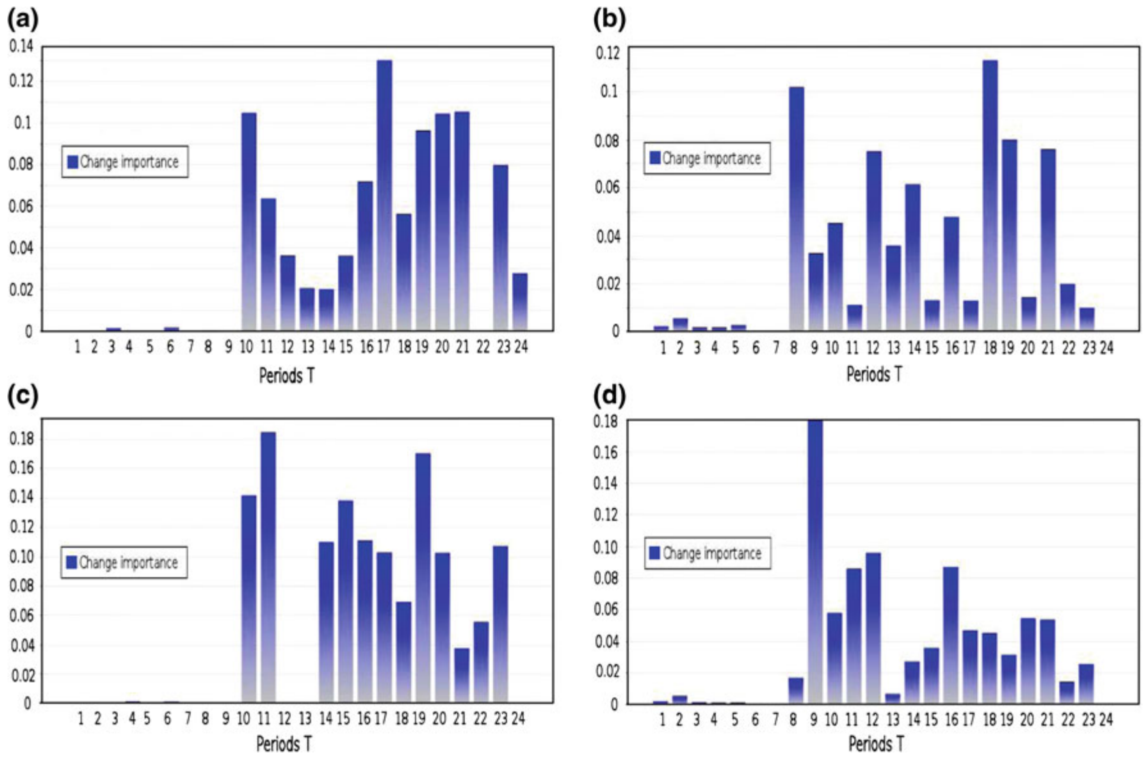


Fig. 8 Samples of time series

Figure 8 shows four time series samples obtained from *Sport.FranceTV.fr* page during weekdays. It clearly appears that there is similar periodicity of changes over the four days. Furthermore, the periodicity of changes for some pages may depend on the time of the day.

7.3 Pattern discovery

The strategy used to discover patterns from already collected time series is described in this section.

7.3.1 Principle

Our approach used to discover pattern is based on statistical summarization technique (i.e., change average). It consists on averaging the importance of changes of each time series collected during a long period. In other words, the importance of changes (at each period) of a pattern is the sum of the changes importance of all time series (at the same period) divided by the total number of collected time series (of the same page). Separate patterns can be learned for a specific day of the week or for weekends. For the pages that change few times per week (or per month), a weekly (or monthly) pattern can be defined.

Assume that $TS = \{TSeries_1, TSeries_2, \ldots, TSeries_{N_{Tseries}}\}$ is the list of time series of the page P_i collected during a long period. Let $Patt(P_i) = \{(\omega_1, T_1); \ldots; (\omega_k, T_k); \ldots; (\omega_{N_T}, T_{N_T})\}$ be the pattern to discover from *TS*. The average importance of changes ω_k defined in $Patt(P_i)$ at the period T_k is computed as follows.

$$\omega_k = \frac{\sum_{j=1}^{N_{Tseries}} \mu_k}{N_{Tseries}}$$

where

- $TSeries_j(\mathrm{P}_i) = \{(\mu_1, T_1); \ldots; (\mu_k, T_k); \ldots; (\mu_{N_T}, T_{N_T})\}$.
- μ_k is the importance of changes of $TSeries_j(P_i)$ at the period T_k.
- $N_{Tseries}$ is the total number of collected times series of the page P_i.

7.3.2 Pattern discovery algorithm

A pseudo code of the pattern discovery algorithm is depicted by the Algorithm 3. For each page P_i, a new pattern is mined from the collection of time series. The average importance of changes ω_k is computed for each period T_k of every pattern (line 3–9). At the end of the algorithm, the collection of discovered patterns of all pages is returned (line 11–13).

The Fig. 9 shows samples of patterns obtained from the following pages; (a) *programmes.France3.fr*, (b) *La1ère.fr*, (c) *Sport.FranceTV.fr* and (d) *documentaire.France5.fr*.

Algorithm 3 Pattern discovery

```
Input:
P_1, P_2, ..., P_n - list of pages
TSeries_1, TSeries_2, ..., TSeries_{N_Tseries} - times series of P_i
Output:
Patterns - list of discovered patterns
Begin
1. for each page $P_i$ i = 1...,$n$ do
2.     Patt = newPattern()
3.     for each period $T_k$ k = 1,..$N_T$ do
4.         imp = 0
5.         for each $TSeries_j$ of $P_i$ j = 1,..,$N_{Tseries}$ do
6.         imp = imp+ TSeries.getImportance($T_k$)
7.         end for
8.         $\omega$ = imp/$N_{Tseries}$
9.         Patt.add($\omega$, $T_k$)
10.    end for
11. Patterns.add($P_i$,Patt)
12. end for
13. Return Patterns
End
```

7.3.3 Observations

We have observed from some patterns significant changes during the day (respectively during the night) and during the weekdays (respectively during weekends). Some pages (e.g., *La 1 ère.fr*, etc.) have similar changes importance during the day and during the night. This is due to the time zone difference between France and its overseas departments that affects the periods of work activities. Separate patterns are also learned for some pages at specific days (holidays) because their change behaviors differ from usual. In addition, we have observed that the pages at the highest level in the site (e.g., home pages) have more significant changes than those at the deepest levels in the site, which remain quasi static. In this study, we focus on discovering temporal patterns. Nonetheless, we have noticed that structural-changes patterns can also be discovered from some Web pages. For example, we have observed that some types of changes occur usually in specific blocks of the page. Discovering patterns from structural-changes (or from content-changes) is out of scope of the paper.

7.3.4 Shared patterns

From the obtained patterns, it clearly appears that some pages (e.g., *France2.culture.fr* and *France3.culture.fr*) have the same template and share similar content information. Also, they present similar change behaviors. Hence, a common pattern can be defined for such pages. Other pages with different content belonging to the same site (or to a different site) have similar changes evolution. Such pages can also be grouped to share a common pattern. Patterns can be learned according to the depth of the page in the site. For instance, a common pattern can be defined for some pages at the deepest level because they change rarely. Further study should be

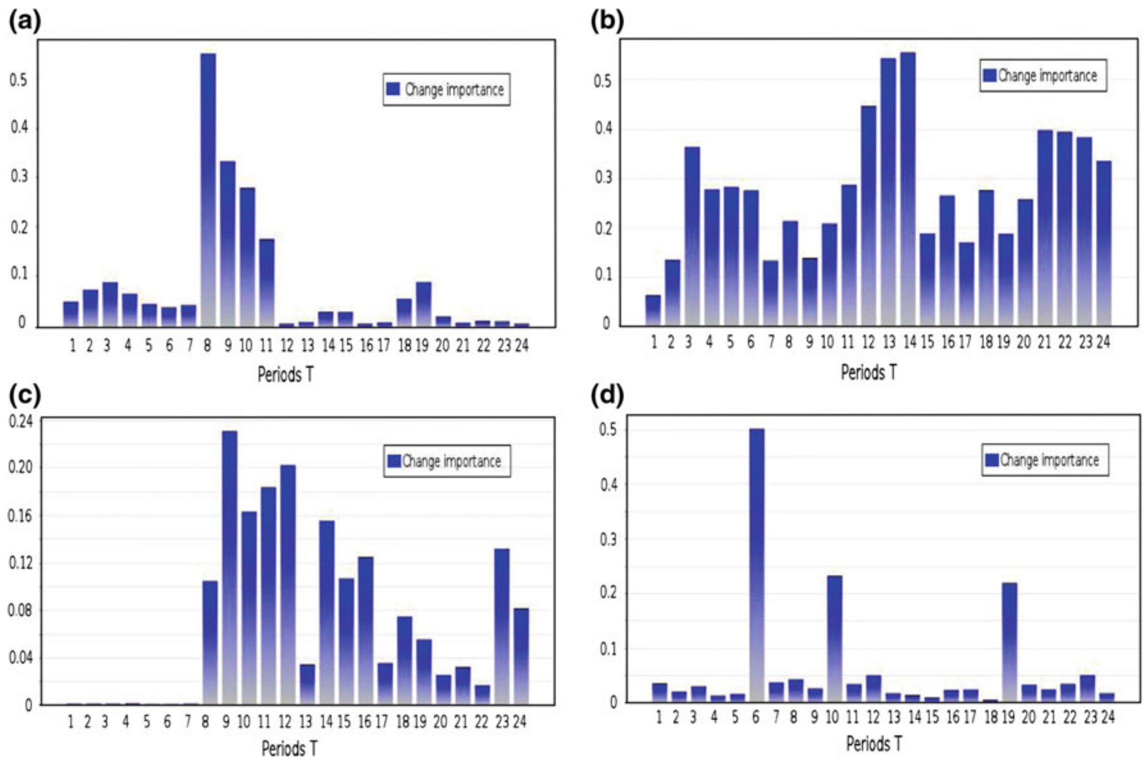


Fig. 9 Samples of discovered patterns

done to learn an efficient methods to create automatically a collection of pages that share common patterns.

8 Pattern-based indexing

We show in this section how patterns can be exploited to optimize page indexing.

8.1 Principle

An appropriate change importance threshold is fixed to decide if a retrieved version should be indexed or no. Hence, archive systems avoid wasting time and space for indexing some versions with unimportant changes. As the pattern gives an idea of the evolution of change importance over time, accurate value of threshold δ can be learned for each page. One way to fix the index threshold δ of each page is to compute the average of changes importance of its patterns.

The index threshold of the page P_i is the sum of all changes importance ω_k in $Patt(P_i)$, divided by the number of periods N_T.

$$\delta(P_i) = \frac{\sum_{k=1}^{N_T} \omega_k(Patt(P_i))}{N_T}$$

where $Patt(P_i) = \{(\omega_1, T_1); \ldots; (\omega_k, T_k); \ldots; (\omega_{N_T}, T_{N_T})\}$ The threshold δ can also serve to optimize storing in case of a limited storage space. This threshold can be adjusted with respect to allocated resources by using machine learning algorithms for example.

8.2 Pattern-based indexing algorithm

As depicted by Algorithm 4, if the importance of changes between the current version and the last stored one (line 5–7) is higher than the threshold δ, the page version is indexed in the archive. Otherwise, the estimated importance is cumulated over time (line 8–10). Whenever this cumulated importance of changes becomes higher than the fixed threshold, the current version is indexed/stored (line 5–6).

Figure 10 shows an example of threshold used to index the page *France4.fr*.

9 Pattern-based crawling

We have described in Sect. 6.3 how patterns are discovered from Web pages. In this section, we show how patterns can be exploited to optimize the strategy of Web crawlers.

Algorithm 4 Pattern-based indexing

```
Input:
$\delta(P_1), \delta(P_2), \ldots, \delta(P_n)$ - index thresholds of P$_i$
$v_1^i, v_2^i, \ldots, v_k^i$ - versions of page P$_i$
Begin
1. imp = 0
2. for each new version $v_\eta^i$ of $P_i$ do
3.     $\Delta =$ DetectChanges($v_{\eta-1}^i, v_\eta^i$)
4.     $\mu =$ EstimateChangeImportance($\Delta$)
5.     if $\mu \geq \delta(P_i)$ or imp $\geq \delta(P_i)$ then
6.         index the new version $v_\eta^i$ of $P_i$
7.         imp = 0
8.     else
9.         imp = imp + $\mu$
10.    end if
11. end for
End
```

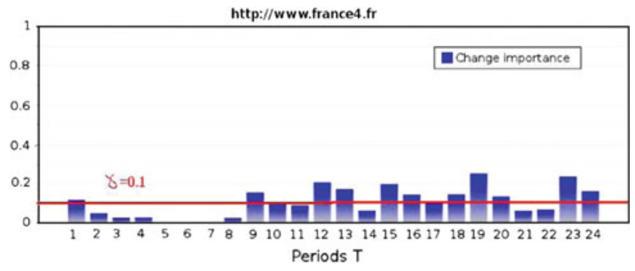


Fig. 10 Example of indexing threshold

9.1 Principle

The pattern-based strategy consists in downloading, in priority, the most important pages to improve the effectiveness of the archive. The scheduler orders the pages to be crawled based on an urgency (or a priority) which depends on the pattern.

The urgency $U(P_i, t)$ of downloading the page P_i at current time t is

$$U(P_i, t) = \omega(P_i) * \omega_k * (t - t_{lastRefresh})$$

where

- $Patt(P_i) = \{(\omega_1, T_1); \ldots; (\omega_k, T_k); \ldots; (\omega_{N_T}, T_{N_T})\}$.
- t is the current time ($t \in T_k$).
- ω_k is the average of change importance defined by the pattern for the period T_k.
- $\omega(P_i)$ is the page's importance.
- $t_{lastRefresh}$ is the last time of refreshing the page P_i. Only the M first pages with the highest urgency are captured at each period T_k.

9.2 Pattern-based crawling algorithm

The pseudo code of the pattern-based crawler is depicted by Algorithm 5. At each period T_k (line 2–5), the urgency of refreshing each page is computed then added to the list

Algorithm 5 Pattern-based crawler

```
Input:
P_1, P_2, ..., P_n - list of pages
Patt(P_1), Patt(P_2), ..., Patt(P_n) - patterns of pages

Begin
1. for each period $T_k$ do
2.    crawlListPages = newList()
3.    for each page $P_i$, $i = 1 \dots, n$ do
4.       compute $U(P_i, t) = \omega(P_i) * \omega_k * (t - t_{lastRefresh})$
5.       crawlListPages.add($P_i$, $U(P_i, t)$)/*descending order*/
6.    end for
7.    for i=1...,M do
8.       $P_i$=crawlListPages.selectPage(i)
9.       $v^i_\eta$ = downloadPage($P_i$)
10.      $v^i_{\eta-1}$ = getLastVersion($P_i$)
11.      $\Delta$ = detectChanges($v^i_{\eta-1}$, $v^i_\eta$)
12.      $\mu$ = EstimateChangesImportance($\Delta$)
13.      Update(Patt($P_i$), $\mu$, $T_k$)
14.      $t_{lastRefresh}$ = t
15.   end for
16. end for
End
```

crawlListPages in descending order. After that, the crawler downloads the M top pages with the highest urgency value. For each page, the current version and the last archived one are compared to build the delta file (line 9–11). The changes occurred in the delta are then evaluated. The result of the importance of change is used to maintain patterns up-to-date. In fact, the average change importance ω_k of patterns should constantly be revaluated to reflect the Web. If more (respectively less) changes are detected in period T_k, the *Update* function (line 13) recomputes the average of changes importance ω_k for patterns. Also, the value of $\omega(P_i)$ is periodically recomputed because the importance of pages (e.g., PageRank) changes over time in the Web.

10 Experimental results

We have conducted experiments to evaluate the efficiency of our pattern-based approach to improve the quality of archives. For that, patterns discovered from *FranceTV* Web pages are exploited to optimize Web crawling. Experiments were conducted on both simulated and on real data. In particular, we compared the temporal completeness obtained by current strategies in use. The weighted completeness measure presented in Sect. 3 is used to assess the quality of archived pages versions.

We start by describing the different strategies considered in this work, named *Relevance*, *Frequency*, *Coherence*, *SHARC*, and *Importance-Pattern*. Their description is the following:

Relevance: under this strategy, the crawler downloads the most relevant/important pages first based on PageRank. Thus, the page are sorted in descending order of their importance and are downloaded in a fixed order. This strategy proposed in [17].

Frequency: under this strategy, the crawler selects the pages to be archived according to their frequency of changes. This policy corresponds to the approach proposed in [15] to improve the freshness of Web pages. The frequency of changes for each page is evaluated under Poisson model based on Cho estimator [16].

Coherence: this strategy corresponds to the approach proposed by [39] to improve the coherence of archives. In this strategy, the crawler works at the granularity of a site. It repeatedly downloads an entire site. For each page, the probability of encountering incoherence during the crawl is computed. The less "risky" pages (i.e., with lowest probability to cause incoherences) are downloaded first.

SHARC: this strategy corresponds to the approach proposed in [18] to improve the sharpness of the archive. Similar to *Improved-Coherence*, the crawler works at the granularity of site by repeatedly downloading the entire site. Under this strategy, the most changing pages are captured as close as possible to the middle of the capture interval.

Importance-Pattern: this is our crawling strategy as described in Sect. 9. Pages are downloaded according to their changes importance predicted by patterns. It combines the two concepts importance of changes and patterns.

10.1 Simulated Web pages

As it is not trivial to capture exactly all page changes which occurred on the Web to measure the completeness, we have simulated the changes importance of Web pages based on patterns discovered from *FranceTV* pages. The performance of different crawl strategies within a controlled test environment are evaluated. Experiments written in Java were conducted on PC running Linux over a 3.20 GHz Intel Pentium 4 processor with 1.0 GB of RAM. At the begining of each experiment, each page is described by a real pattern that has been discovered from *FranceTV* Web pages. The importance of changes of each page is simulated based on the discovered pattern. In addition, the following parameters are set: the number of Web pages, the duration of simulation, the number of periods in patterns N_T, and the allocated resources M. Here, M is the percentage of pages that can be crawled during a period T_k with respect to the total number of pages belonging to the archive.

We have simulated the crawl of about one thousand Web pages from *FranceTV*. The number of periods defined in each pattern is fixed to 24 (i.e., the duration of each period is 1 hour). The resource constraint is modeled by assuming that the crawler can download a rate of M pages in each period

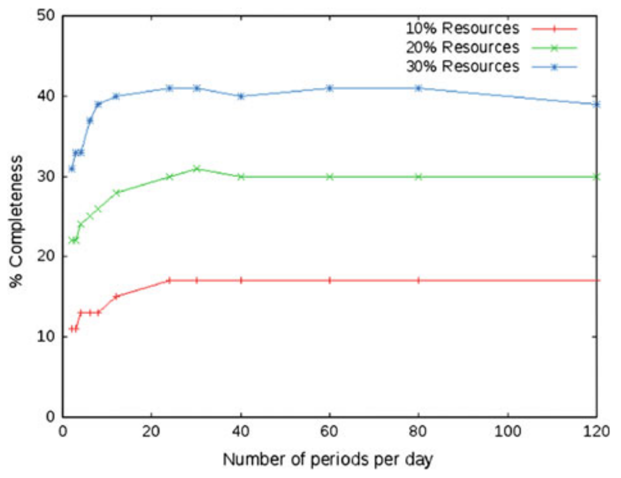


Fig. 11 Pattern time periods

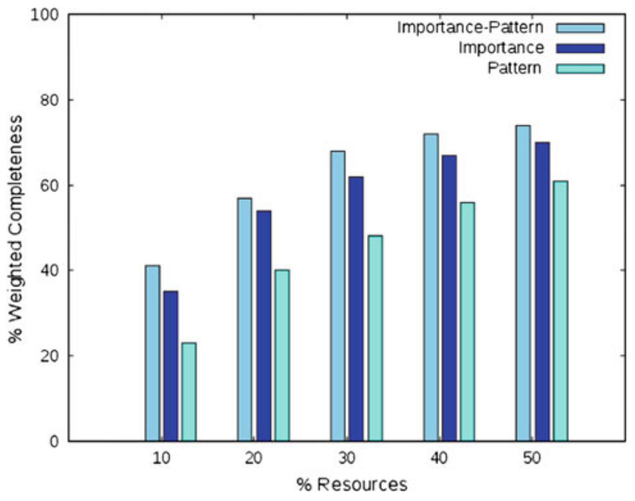


Fig. 12 Ours strategies comparison

T_k. The value of M varies from 10 to 50 % to simulate a Web crawler with limited resources. All experiments that have been conducted to evaluate the different strategies are done under the same conditions.

First experiments were performed to see the impact of the number (or the length) of periods defined in patterns on the effectiveness of the pattern-based strategy. There exists a clear trade-off in deciding how long should be each period T_k defined in the pattern. If the length of periods is too long (i.e., there are few periods), the discovered pattern may be inaccurate and hence our strategy may be less efficient. If the length of periods is too short, the crawling strategy may be very costly in terms of allocated resources.

Figure 11 shows the completeness obtained by our pattern-based strategy under resource constraints $M = \{10, 20, 30\,\%\}$ with respect to the number of periods (N_T) defined in patterns. The length of each period T (in hours) and the corresponding periods' numbers N_T defined in patterns are summarized in the following table.

N_T	2	3	4	6	8	12	24	30	40	60	80	120
T(h)	12	8	6	4	3	2	1	$\frac{4}{30}$	$\frac{1}{10}$	$\frac{2}{30}$	$\frac{1}{20}$	$\frac{1}{30}$

As we can see in the Fig. 11, the completeness (under resources constants 10, 20 and 30 %) increases with the number of periods defined in pattern. When more periods are defined in the pattern (i.e., short period length), better completeness is achieved by our pattern-based strategy. The achieved completeness remains quasi constant for patterns having a number of periods higher than 24, i.e., a period length shorter than one hour. Even if the period length of the pattern is too short (less than 30 min), the achieved completeness is not furthermore improved. Hence, the length of periods defined in patterns should be around one hour (or shorter) to reach the best completeness rate.

To verify the individual impact of the two concepts (patterns and importance of changes) to optimize the completeness of archives, we have first compared our *Pattern-Importance* approach with two other strategies that we have proposed: *Importance* and *Pattern*. *Importance* strategy considers the importance of changes without using patterns. It is based on a fixed average of changes importance ω_k (in the urgency function) which does not depend on patterns. *Pattern* strategy does not consider the importance of changes between pages. It depends on the change rate (in the urgency function) instead of the average importance of changes.

We compared the weighted completeness obtained by our three strategies: *Importance*, *Pattern* and *Importance-Pattern* with respect to allocated resources ($M = 10$–$50\,\%$). Results of experiments are shown in the Fig. 12. From the figure, we can see that *Importance-Pattern* outperforms the other strategies. It improves completeness by respectively 5 and 20 % than *Importance* and *Pattern*. Note that this improvement (5 % of completeness) obtained by *Importance-Pattern* compared to *Importance* can exceed 10 %, if patterns of pages are different from each other (i.e., changes occur in different periods). These experiments have confirmed the utility of considering the two concepts: patterns and importance of changes to maximize the temporal completeness of archives.

In the rest of the paper, the temporal completeness obtained by our strategy *Importance-Pattern* is compared to current policies in use that have been described above: *Relevance*, *Coherence*, *SHARC* and *Frequency*.

The Fig. 13 shows the weighted completeness obtained by the different strategies with respect to the percentage of allocated resources $M = [10$–$50\,\%]$. From the figure, we can see that the temporal completeness increases with the number of allocated resources. Moreover, it is clear that the

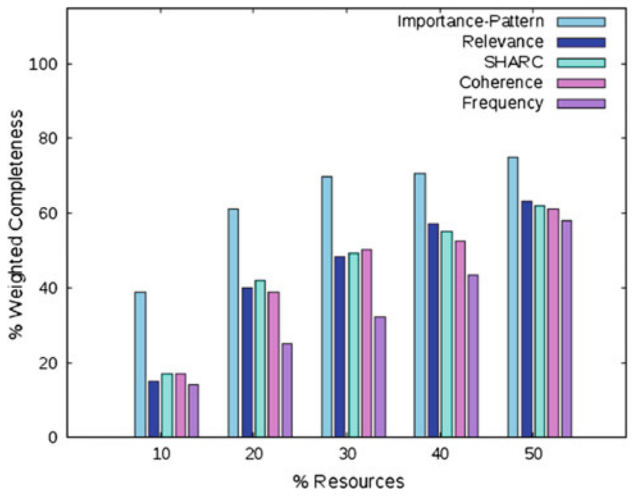


Fig. 13 Weighted completeness (simulation)

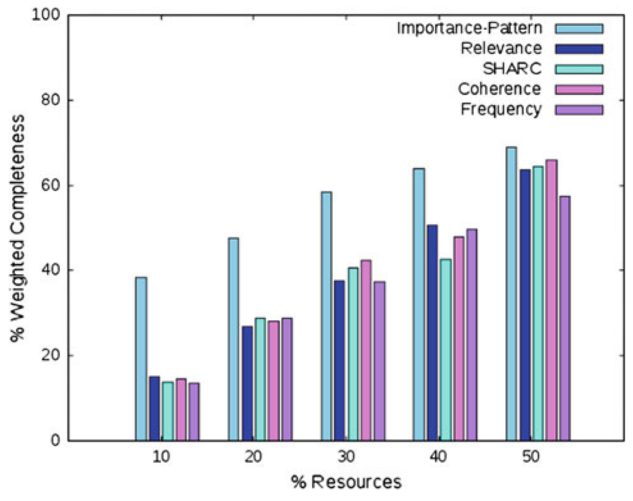


Fig. 14 Weighted completeness (real Web pages)

Pattern-Importance strategy outperforms the other approaches *Relevance*, *SHARC*, *Coherence* and *Frequency*. It improves significantly the weighted completeness by around 20 % in case of limited resources. In other words, it enables to download more important versions (20 % better) than its competitors. Hence, it avoids wasting time and space to download useless page versions.

10.2 Real Web pages

We evaluated also our pattern-based strategy on real Web pages of *FranceTV*. A specific crawler has been implemented for each of these strategies: *Importance-Pattern*, *Relevance*, *SHARC*, *Coherence* et *Frequency*. In addition, we implemented a crawler, named *history*, that visits periodically (every 15 min) Web pages to keep their historical changes over time. Each current version captured by the crawler is segmented and compared with the last archived versions (of the same page) to assess the importance of detected changes. We measured the temporal completeness achieved by each strategy. Is is measured by computing the weight of the versions that have been captured divided by the total weight of versions appeared on the Web and captured by the *history* crawler. Even it is not easy to capture accurately all versions that have appeared on the Web, the crawler *history* enables to keep an approximate history of the importance of changes over time. We can consider that a page content that changes or disappears after less than 15 min corresponds to a negligible modification.

The Fig. 14 shows the weighted completeness obtained by the different strategies with respect to allocated resources. Under limited resources ($M = [10–50\,\%]$), our strategy *Importance-Pattern* improves the weighted completeness of *FranceTV* pages from 20 to 25 % compared to other approaches. Hence, these improvements of completeness obtained over real Web pages confirm the results achieved by the simulated experiments.

10.3 Discussion

The results obtained by our experiments over simulated and real Web page have shown that our *Importance-Pattern* policy performs better than its competitors *Relevance*, *SHARC*, *Coherence*, and *Frequency*. It improves the weighted completeness of archives from 20 to 25 % in case of very limited resources. Even it is impossible to achieve a perfect quality of archives (100 % of temporal completeness), these experiments demonstrate that patterns that describe the importance of changes are very useful to optimize the quality of archives. They allow the crawler to download more important page versions compared to existing strategies. This avoids archiving systems to waste time and storage space to save or to index unimportant page versions. Available resources can be exploited to capture more important versions that have significant changes since the last archived ones. Learning patterns offline is probably the most costly step of our archiving approach in terms of needed resources. However, it is important to note that we are not obliged to discover all the patterns offline. Pattern can be learned online by generating for example random patterns at the beginning of the Web crawl. Common patterns can be also assigned to pages that have similar behaviors of page changes. Then, each pattern of the page is progressively learned and updated during online crawl. The experiments presented in this paper were conducted over small collections of *FranceTV* pages. We intend to perform our strategy over a larger number of media pages and also over other dynamic pages such as online newspapers.

11 Conclusion and future work

The major challenge of Web archiving institutes is to provide a rich archive and to preserve its quality. In this paper, we proposed a novel approach based on patterns to (i) efficiently index/store Web pages and (ii) improve archives quality. The quality of an archive is defined by the completeness measure which insures that the archive contains the largest amount of useful pages versions. As far as we know, this work is the first to use patterns in the context of Web archiving. Through a case study, we presented algorithms used to discover patterns from French Televisions channels pages. A method using patterns is proposed to optimize page indexing and storing. Also, we proposed a crawl strategy based on pattern to improve the quality of archives. Evaluation experiments were performed based on real patterns obtained from *FranceTV* pages. These experiments show that our approach outperforms its competitors, obtaining a completeness gain up to 20 % in case of limited resources. This improvement is also due to considering the importance of changes between versions which has been ignored so far. Our Web archiving approach is not limited to *FranceTV* pages collections, it can also be exploited to optimize the archive of others organisations particularly those who collect frequently changing pages such as on-line-newspapers, pages with dynamic forums, financial Websites, etc.

As a future direction, we plan to run our pattern strategy over a larger number of dynamic Web pages of different archiving organisations. We plan also to implement our pattern-based indexing/storing strategy by learning appropriate thresholds to decide when each page should be saved (or indexed). Moreover, we intend to explore other pattern summarization techniques such as association rules to predict the behavior of some unexpected changes on Web pages. Another future research direction is to create a method that automatically determines collections of Web pages that share a common pattern. Finally, we are also interested in studying the impact of pattern strategy on other quality measures of archives such as sharpness, freshness, etc.

Acknowledgments This research is supported by the French National Research Agency ANR in the CARTEC Project (ANR-07-MDCO-016).

References

1. Abiteboul, S., Cobena, G., Masanes, J., Sedrati, G.: A First experience in archiving the French Web. In: Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries (2002)
2. Adar, E., Teevan, J., Dumais, S.T.: Resonance on the web: web dynamics and revisitation patterns. In: Proceedings of the 27th International Conference on Human Factors in Computing Systems, Boston, MA, USA (2009)
3. Adar, E., Teevan, J., Dumais, S.T., Elsas, J.L.: The web changes everything: understanding the dynamics of web content. In: Proceedings of the Second ACM International Conference on Web Search and Data Mining. Barcelona, Spain (2009)
4. Baron, S., Spiliopoulou, M.: Monitoring the evolution of web usage patterns. In: Lecture Notes in Computer Science, pp. 181–200. Springer, New York (2004)
5. Ben Saad, M., Gançarski, S.: Using visual pages analysis for optimizing web archiving. In: EDBT/ICDT PhD Workshops (2010)
6. Bogen, P.L. II, Francisco-Revilla, L., Furuta, R., Hubbard, T., Karadkar, U.P., Shipman, F.: Longitudinal study of changes in blogs. In: Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL '07, pp. 135–136. ACM, New York (2007)
7. Bogen, P.L. II, Johnston, J., Karadkar, U.P., Furuta, R., Shipman, F.: Application of kalman filters to identify unexpected change in blogs. In: Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries. JCDL '08, pp. 305–312 (2008)
8. Brewington, B., Cybenko, G.: How dynamic is the web? In: World Wide Web conference (WWW'2000), pp. 257–276 (2000)
9. Brewington, B.E., Cybenko, G.: Keeping up with the changing web. Computer **33**(5), 52–58 (2000)
10. Cai, D., Yu, S., Wen, J.-R., Ma, W.-Y.: VIPS: a Vision-based Page Segmentation Algorithm. Technical Report, Microsoft Research (2003)
11. Castillo, C., Marin, M., Rodriguez, A., Baeza-Yates, R.: Scheduling algorithms for web crawling. In: LA-WEBMEDIA '04: Proceedings of the WebMedia & LA-Web 2004 Joint Conference 10th Brazilian Symposium on Multimedia and the Web 2nd Latin American Web Congress, pp. 10–17. (2004)
12. Cathro, W.: Development of a digital services architecture at the national library of Australia. EduCause, (2003)
13. Cheng, H., Yan, X., Han, J., wei Hsu, C.: Discriminative frequent pattern analysis for effective classification. In: ICDE, pp. 716–725 (2007)
14. Cho, J., Garcia-Molina, H.: The Evolution of the Web and Implications for an Incremental Crawler. In: VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases. (2000)
15. Cho, J., Garcia-Molina, H.: Effective page refresh policies for web crawlers. ACM Trans. Database Syst. **28**(4), 390–426 (2003)
16. Cho, J., Garcia-Molina, H.: Estimating frequency of change. ACM Trans. Interet Technol. **3**(3), (2003)
17. Cho, J., Garcia-molina, H., Page, L.: Efficient crawling through url ordering. In: Computer Networks and ISDN Systems, pp. 161–172 (1998)
18. Denev, D., Mazeika, A., Spaniol, M., Weikum, G.: Sharc: framework for quality-conscious web archiving. Proc. VLDB Endow. **2**(1), 586–597 (2009)
19. Edwards, J., McCurley, K., Tomlin, J.: An adaptive model for optimizing performance of an incremental web crawler. In: Proceedings of the 10th international conference on World Wide Web, WWW '01, pp. 106–113 (2001)
20. Francisco-Revilla, L., Shipman, F., Furuta, R., Karadkar, U., Arora, A.: Managing change on the web. In: Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries, JCDL '01, pp. 67–76 (2001)
21. Francisco-Revilla, L., Shipman, F.M. III, Furuta, R., Karadkar, U., Arora, A.: Perception of content, structure, and presentation changes in web-based hypertext. In: Proceedings of the 12th ACM conference on Hypertext and Hypermedia, HYPERTEXT '01, pp. 205–214 (2001)
22. Gomes, D., Santos, A.L., Silva, M.J.: Managing duplicates in a web archive. In: SAC '06: Proceedings of the 2006 ACM Symposium on Applied Computing (2006)

23. Gruhl, D., Guha, R., Liben-nowell, D., Tomkins, A.: Information diffusion through blogspace. In: WWW ’04, pp. 491–501 (2004)
24. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. Data Min. Knowl. Disc. **15**, 55–86 (2007)
25. Knotek, J.: Information extraction from advertisements. Master’s thesis, Masaryk University (2011)
26. Lampos, D.J.C., Eirinaki, M., Vazirgiannis, M.: Archiving the greek web. In: 4th International Web Archiving Workshop (IWAW04). Bath, UK (2004)
27. Li, Z., Chen, Z., Srinivasan, S.M., Zhou, Y.: C-miner: Mining block correlations in storage systems. In: Proceedings of the 3rd USENIX Conference on File and Storage Technologies, pp. 173–186 (2004)
28. Li, Z., Lu, S., Myagmar, S., Zhou, Y.: Cp-miner: a tool for finding copy-paste and related bugs in operating system code. In: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation, vol. **6**, pp. 20–20 (2004)
29. Masanès, J.: Web Archiving. Springer, Secaucus (2006)
30. Mazeika, A., Denev, D., Spaniol, M., Weikum, G.: The SOLAR System for Sharp Web Archiving. In: Proceedings of the 10 th International Web Archiving Workshop (IWAW), pp. 24–30. Vienna, Austria, September (2010)
31. Oita, M., Senellart, P.: Archiving data objects using Web feeds. In: Proceedings of the 10 th International Web Archiving Workshop (IWAW), Vienna, Austria, September (2010)
32. Olston, C., Pandey, S.: Recrawl scheduling based on information longevity. In: Proceeding of the 17th International Conference on World Wide Web. WWW ’08, pp. 437–446 (2008)
33. Pandey, S., Olston, C.: User-centric web crawling. In: Proceedings of the 14th International Conference on World Wide Web, WWW ’05, pp. 401–411 (2005)
34. Pehlivan, Z., Ben Saad, M., Gançarski, S.: Vi-diff: Understanding web pages changes. In: 21st International Conference on Database and Expert Systems Applications (DEXA’10). Bilbao, Spain (2010)
35. Saxena, K., Shukla, R.: Significant Interval and Frequent Pattern Discovery in Web Log Data. CoRR, abs/1002.1185 (2010)
36. Sia, K.C., Cho, J., Cho, H.-K.: Efficient monitoring algorithm for fast news alerts. IEEE Tran. Knowl. Data Eng. **19**, 950–961 (2007)
37. Sia, K.C., Cho, J., Hino, K., Chi, Y., Zhu, S., Tseng, B.L.: Monitoring RSS feeds based on user browsing pattern. In: ICWSM ’07: International Conference on Weblogs and Social Media (2007)
38. Song, R., Liu, H., Wen, J.-R., Ma.: Learning block importance models for web pages. In: WWW ’04: Proceedings of the 13th International Conference on World Wide Web. (2004)
39. Spaniol, M., Denev, D., Mazeika, A., Weikum, G., Senellart, P.: Data quality in web archiving. In: WICOW ’09: Proceedings of the 3rd Workshop on Information Credibility on the Web, pp. 19–26 (2009)
40. Spaniol, M., Mazeika, A., Denev, D., Weikum, G.: Catch me if you can: Visual analysis of coherence defects in web archiving. In: 9th International Web Archiving Workshop (IWAW 2009): Workshop Proceecdings, pp. 27–37. Corfu, Greece (2009)
41. Srivastava, J., Cooley, R., Deshpande, M., Tan, P.-N.: Web usage mining: discovery and applications of usage patterns from web data. SIGKDD Explor. Newsl. **1**, 12–23 (2000)
42. Ueda, T., Hirate, Y., Yamana, H.: Exploiting idle cpu cores to improve file access performance. In: ICUIMC ’09: Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication, pp. 529–535 (2009)
43. Vaarandi, R.: A data clustering algorithm for mining patterns from event logs. In: IEEE IPOM’03 Proceedings, pp. 119–126. (2003)
44. Wolf, J.L., Squillante, M.S., Yu, P.S., Sethuraman, J., Ozsen, L.: Optimal crawling strategies for web search engines. In: Proceedings of the 11th International Conference on World Wide Web, WWW ’02, pp. 136–147. (2002)
45. Yang, L.H., Lee, M.L., Hsu, W.: Efficient mining of xml query patterns for caching. In: Proceedings of the 29th International Conference on Very Large Data Bases, VLDB ’2003, vol. 29, pp. 69–80 (2003)