Using Visual Pages Analysis for optimizing Web Archiving*

Myriam Ben Saad
Advisor: Stéphane Gançarski
LIP6, University P. and M. Curie Paris, France
Myriam.Ben-Saad@lip6.fr

ABSTRACT

Due to the growing importance of the World Wide Web, archiving it has become crucial for preserving useful source of information. To maintain a web archive up-to-date, crawlers harvest the web by iteratively downloading new versions of documents. However, it is frequent that crawlers retrieve pages with unimportant changes such as advertisements which are continually updated. Hence, web archive systems waste time and space for indexing and storing useless page versions. Also, querying the archive can take more time due to the large set of useless page versions stored. Thus, an effective method is required to know accurately when and how often important changes between versions occur in order to efficiently archive web pages. Our work focuses on addressing this requirement through a new web archiving approach that detects important changes between page versions. This approach consists in archiving the visual layout structure of a web page represented by semantic blocks. This work seeks to describe the proposed approach and to examine various related issues such as using the importance of changes between versions to optimize web crawl scheduling. The major interesting research questions that we would like to address in the future are introduced.

Keywords

Web archiving, change detection, visual page analysis, web crawling

1. MOTIVATION

Archiving the web has become crucial for preserving useful source of information. For this reason, it has become an issue for many national archiving institutes around the world. However, the web is highly dynamic, evolving over time (pages change frequently). Most often, web archiving is automatically performed using web crawlers. Web crawlers visit web pages to be archived and build a snapshot and/or index of web pages. In order to maintain the archive up-to-date, crawlers must revisit periodically the pages and update the archive with fresh images. However, the crawler can not revisit a site and download a new version of a page all the time because it usually has limited resources (such as bandwidth, space storage, etc.) with respect to the huge amount of pages to archive. Thus, the archiving system must estimate the behavior of a site in order to guess when or with which frequency it must be visited. This can avoid downloading duplicated versions during the web crawl.

Several works [9, 8] have focused on estimating the change frequency to improve the Web crawlers. However, it's frequent that crawlers waste time and space to download a new page version with unimportant changes such as advertisements which are continually updated. Thus, an effective method is required to know accurately when and how often important changes between versions occur. Up to now, approaches that estimate the frequency of changes only take into account the amount of detected changes. But, they do not consider the *importance* of changes that have occurred. If we can predict important changes frequencies much more accurately, web crawl can be optimized.

In order to estimate the frequency of updates, changes between already retrieved versions of documents must be detected. Many existing algorithms [15, 10, 19] have been specially designed to detect changes between semi-structured documents (XML and HTML). However, there is no method that detects and distinguishes relevant/irrelevant changes from useful/useless (noisy) information.

Our goal is to propose an approach for (1) detecting important changes between versions and exploit it to (2) optimize web crawling, to (3) efficiently index/store pages versions and to (4) temporally request archived pages and changes occurred between them. Our approach will be applied on a repository for the French National Audiovisual Institute (INA). One of the missions of INA is to create a legal deposit which preserves French radios and televisions web pages and related pages. A strong requirement for this project is that the visual aspect of the pages must be preserved. Thus our idea is to use a visual page analysis to assign importance to web pages parts, according to their relative location.

Previous works [5, 12] show that a page can be partitioned into multiple segments or blocks and, often, the blocks in a page have different importances. In fact, different regions inside a web page have different importance weights according to their location, area size, content, etc. Typically, the

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

EDBT 2010, March 22-26, 2010, Lausanne, Switzerland

most important information is on the center of a page, advertisement is on the header or on the left side and copyright is on the footer. Once the page is segmented, then a relative importance must be assigned to each block. This can be achieved automatically using for instance the algorithm of [18], or though a supervised machine learning method. Then, we can compute the importance of changes between two page versions, based on (i) the relative importance of blocks and (ii) the relative importance of operations (insert, delete, update, etc.) occurred in those blocks, detected by comparing the two versions. Thus, our work seeks to combine those concepts to address issues related to detecting/analyzing important web pages changes. Then, the analysis results can be exploited for an optimized web crawling, an adequate index/storage and an efficient temporal querying.

This paper is structured as follows. Section 2 defines our research problem and identifies all related issues. Section 3 describes the proposed approach including the web archive architecture. In Section 4, some related works are discussed. Section 5 describes, in more details, the different steps of our approach. Some preliminary results are presented in Section 6. Finally, Section 7 concludes and discusses issues that we would like to address in future works.

2. RESEARCH PROBLEM

The design of web archiving systems presents many challenges; (i) an optimized web crawling, (ii) an adequate index/storage and (iii) an efficient temporal querying. By pointing out the problem of downloading useless/unimportant page versions, we can considerably improve the effectiveness of those three points. Thus, web crawlers must carefully decide what page should be refreshed/archived and with which priority. They must also decide how frequently pages should be revisited in order to keep the web archive as up-to-date as possible. In fact, it is impossible to maintain a complete archive of the whole Web, or even a part of it, containing all the versions of all the pages because the web is evolving over time and allocated resources are usually limited. Thus, the problem can be stated as follows: how to optimize the crawling in order to download the "most important versions", so that the minimum of useful information is lost? An important version is the version that has "important" changes since the last one archived. Of course, this problem must be solved without any help from web sites managers. Thus, an effective method is required to know accurately when and how often "important" changes between versions occur. Up to now, existing approaches that estimate the frequency of crawlers do not take into account the importance of changes between versions. In fact, it's frequent that crawlers download pages with unimportant changes such as advertisements which are continually updated. In order to estimate an appropriate frequency of crawlers, changes between already retrieved versions must be detected and analyzed. Although various algorithms have been designed to detect changes between documents, there is no method that detects and distinguishes important/unimportant changes from useful/useless (noisy) information.

In this thesis, we address some of these important challenges: (1) How can archiving systems detect useful changes between already archived versions and how they can assess their importance? (2) How can crawlers select the most urgent/important version to be refreshed in case of limited resources with the respect to the huge amount of document to be archived? This can be achieved by either developing crawl scheduling strategies or by estimating the frequency of changes. (3) How can the analysis result of changes importance be exploited to decide indexing or storing the different versions? (4) How can the archive be efficiently queried about the different page versions and the changes occurred between them? The proposed approach described in the next section seeks to address the first and the second challenges. The questions (3) and (4) remain open for future works.

3. PROPOSED APPROACH

In this section, we present our proposed approach which points out the issue of efficiently archiving web pages. As mentioned in the introduction, the visual aspect of the web pages must be preserved in our project. Thus, the idea of our web archiving approach is to analyze/archive the visual structure of documents and to assign importance values to web pages blocks, according to their relative location. In other words, page versions are restructured into blocks according to their visual representation. Detecting changes on such restructured page versions gives relevant information for understanding the dynamics of the web sites. Also, it enables to distinguish relevant/irrelevant changes from useful/useless information. Thus, the proposed approach combines three concepts; visual page analysis (or segmentation), change detection and the importance of web pages' blocks in order to optimize web crawling. Those concepts are not new. However, as far as we know, they had never been combined for archiving the web. To better understand our approach the architecture of the web archive is described in greater details.

Our archiving system consists of four major components: the *web crawlers*, the *freshness component*, the *storage component* and the *query engine*. Figure 1 presents an overview of the system.

The Web Crawler. The web crawlers harvest the web by

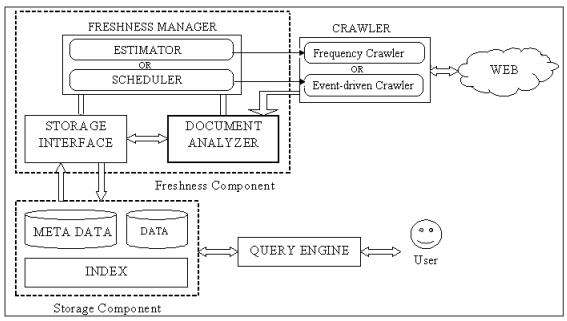

Figure 1: An Overview of the Web Archive.

iteratively downloading documents referenced by URLs. We consider two types of web crawlers. The first one downloads automatically web pages according to a certain frequency given by an estimator. The second one is an event-driven crawler. It downloads the most urgent document to be refreshed, as computed by a scheduler.

The Freshness Component. The freshness component allows maintaining the archive up-to-date. It consists of three main modules: The *freshness manager* enables the

optimization of allocated resources, so that less information is lost. It consists of either a *change frequency estimator* or a *scheduler*. The estimator computes the best change frequency for the first type of crawlers. The scheduler chooses the most urgent page to be downloaded by the event driven crawler in order to maintain the archive as up-to-date as possible. It manages a list of documents ordered by a freshness urgency function. This function estimates, for each page, how it is urgent to refresh it at a given date. Both estimator and scheduler depend on the changes already detected and quantified by the document analyzer on previously archived versions. They take also into account the estimation of changes importance occurred between successive downloaded versions. Based on the freshness urgency function, the scheduler can organize and order the list of pages to be urgently refreshed.

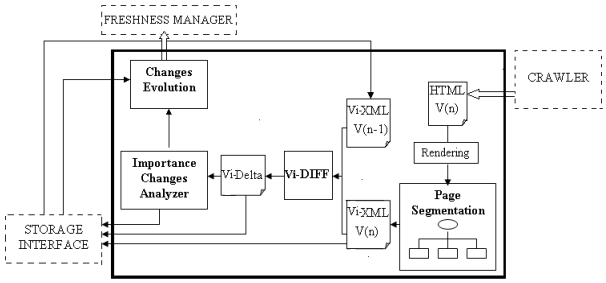

Figure 2: The Document Analyzer.

The *Document Analyzer* allows detecting and analyzing retrieved web page versions. We give here more details on the document analyzer since it is the core of our approach. It consists of several sub-modules corresponding to the various phases of the page analysis. It is depicted in Figure 2. The document analyzer interacts with the crawler to get the current version of the HTML page to be archived. Then, the page is treated by a rendering engine in order to retrieve visual information. The main advantage of rendering is providing a real and a complete visual description of the document even if embedded scripts, such as JavaScript, are present. After that, the rendered page is partitioned and the visual page layout structure is built. The used algorithm of page segmentation generates, as output, a *Vi-XML* document that describes the hierarchical content structure of the page. At the end of the segmentation process, a change detection algorithm (*Vi-DIFF*) provides a description of changes that occurred between the new generated Vi-XML version $V(n)$ and the last version archived $V(n-1)$. Changes are gathered in a delta XML file, called *Vi-Delta*, that describes the operations (insertion, deletion, etc.) occurred between two documents. Thereafter, the Vi-Delta file is analyzed by the submodule *Importance Changes Analyzer* to evaluate the importance of detected changes. The result of this change evaluation can be used by the sub module *Changes Evolution* either to improve the estimation of the crawler frequency or to compute the freshness urgency function used by scheduler. At the end, the Vi-Delta, the current Vi-XML version and additional meta-data are stored in the database through *storage interface*. The storage interface interacts with the storage component to store/index page versions, Vi-Deltas and other meta-data obtained during the analysis.

The Storage Component. The storage component consists of data and meta-data storage units. It includes also an index that facilitates querying the archive.

The Query Engine. Through the query engine, users can temporally request the archive about the different versions and the visual changes occurred between them.

4. RELATED WORKS

The approach proposed in Section 3 is related to different areas. We present below related works in these major areas; web archiving, visual web page analysis, changes detection and web crawling.

Web Archiving. There are several projects launched by different archiving institutes (national libraries, historical data archives, etc.) around the world to preserve their country's web heritage. Most of the web archiving initiatives are described at [1]. Some studies [2, 4] focus on the selection of web pages to be archived by defining the web perimeter. Others [9, 8] work on modeling and evaluating the frequency of web changes. They propose change frequencies estimators and various refresh policies to improve the archive freshness. Some researchers [7, 13] address issues concerning the format of information to be stored and indexed by proposing their own storage system. Others studies [2, 3] focus on the control and the representation of changes. They propose a change detection algorithm and/or a delta format for an efficient query and storage of the web archive.

Though interesting, those approaches do not take the visual aspect and the relative importance of pages parts, which are at the core of our approach.

Visual Web Page Analysis. Several methods have been proposed to analyze the visual representation of web pages. Most approaches discover the logical structure of a page by either analyzing the rendered document or analyzing the document code. Gu et al. [14] propose a top down algorithm which detects the web content structure based on the layout information. Kovacevic et al. [12] define heuristics to recognize common page areas (header, footer, center of the page, etc.) based on visual information. Cai et al. [5] propose the algorithm VIPS which segments the web page into multiple semantic blocks based on visual information retrieved from browser's rendering. Cosulshi et al. [11] propose an approach that calculates the block correspondence between web pages by using positional information of DOM tree's elements. Compared to existing methods, VIPS method described above seems to be the most appropriate for our approach because it allows an adequate granularity of the page partitioning. It builds a hierarchy of semantic blocks of the page that better simulates how a user understands the web layout structure based on his visual perception. Thus, VIPS is used to build the visual structure of documents.

Changes Detection. Several algorithms have been designed to detect changes between two (versions of) semi-structural documents (HTML, XML). They find a minimum set of change operations (insert, delete, ...) that transform one data tree to another. These changes operations are often gathered in a delta script or a delta file. The design of diff algorithms depends on the purposes and the requirements (time complexity, operations to be handled, quality of the delta, etc.). Cobéna et al. [10] propose the XyDiff algorithm to improve time and memory management. XyDiff supports move operation and achieves a time complexity of

$O(n * log(n))$. Despite its high performance, it does not always guarantee an optimal result (*i.e.* minimal edit script). Wang et al. [19] propose X-Diff which can detect the optimal differences between two unordered XML trees in quadratic time $O(n^2)$ but it does not handle a move. DeltaXML [15] can compare, merge and synchronize XML documents for ordered and unordered trees by supporting basic operations but it does not detect a move. There are several other algorithms like DTD-Diff [16], etc. After studying these algorithms, we decided to not use existing methods for our web archiving approach because they are generic-purpose. As we have various specific requirements related to the visual layout structure of documents, we prefer proposing our *ad'hoc* algorithm (Vi-DIFF). This algorithm enables for a better trade-off between complexity and completeness of the detected operations set.

Web Crawling. Several existing studies address the problem of optimizing web crawling by either developing scheduling strategies [6, 17] or by estimating the frequency of changes [9, 8]. However, as far as we know, those studies do not consider the importance of changes that have occurred between already analyzed versions. If we can predict important changes frequencies much more accurately, we may avoid indexing unimportant information and wasting space storage. This is the goal of our work through the proposed approach to improve the effectiveness of the web archive system.

5. STEPS OF PROPOSED APPROACH

We describe here, in greater details, the steps of proposed approach which points out the first problem question; (1) detecting/analyzing changes importance between page versions. Also, a crawl scheduling strategy which uses the analysis of changes importance is proposed. This strategy seeks to optimize the event-driven crawler that addresses one part of the second question (2) as mentioned in section 2.

5.1 Visual Page Segmentation

As mentioned in section 4, VIPS [5] is used to segment a web page into nested semantic blocks based on suitable nodes in the HTML DOM tree of the page. It detects the horizontal and vertical separators in a web page. Based on those separators, it builds the semantic tree of the web page partitioned into multiple blocks. The root is the whole page. Each block is represented as a node in the tree as shown in Figure 3. To complete the semantic tree of the whole page, we extended the VIPS algorithm by extracting links, images and text for each block. As illustrated in Figure 3, each block node has additional children nodes: Links, Images and Texts that gather respectively all hyperlinks, pictures and text contained in the block. All nodes of the page are uniquely identified by an ID attribute. This ID is a hash value computed using the node's content and its children nodes content: if matched nodes (nodes at the same position in two successive versions) have different ID values, then their content has been necessarily updated. Leaf nodes have other attributes such as the name and the address for the hyperlink. Our extended VIPS algorithm generates, as output, a Vi-XML document that describes the complete hierarchical structure of the web page. The structure of such a document is shown in Figure 3.

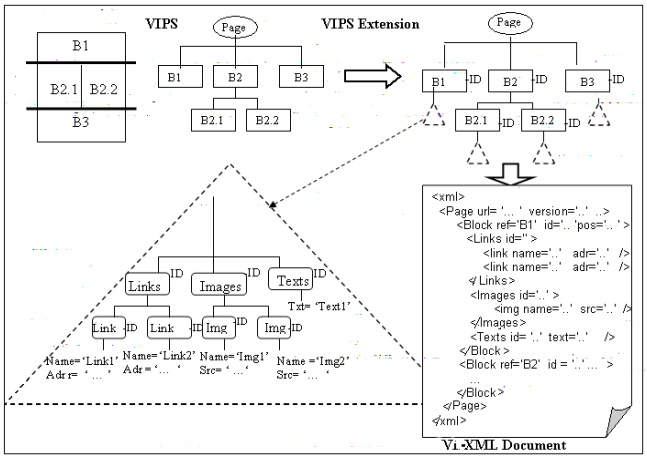

Figure 3: The Extended VIPS Algorithm.

5.2 Changes Detection (Vi-DIFF)

As mentioned in section 4, we have decided to propose our own change detection algorithm because existing ones are generic-purpose. We would like to add some specific criteria for comparing attributes nodes, such as detecting updated links, if one of their attributes (name or address) is modified. We want also to detect an updated text in two matched blocks based on a textual similarity/distance score (*e.g.* number of shared words). Another specificity of our approach is that we need to detect changed elements inside a block and moved elements from one block to another, but detecting moved element inside a same block is useless because no information has been added or deleted in the block. Also, we would like to detect that the structure of the page at level of blocks is changed (inserted/deleted block, etc.) from one version to another. The proposed Vi-DIFF algorithm detects two types of changes; structural and content changes. The structural changes (insert, delete and shift) typically modify the structure of XML document at level of blocks whereas content changes (insert, delete, update and move) modify the textual content at level of links, images and texts. All detected change operations are then described in a Vi-Delta file. If we assume that there is no change in structure then the complexity of Vi-DIFF is log-linear O(n*log(n)) where n is the total number of nodes. If there are structural changes, in the worst case (all the structure is changed), the complexity is quadratic $O(n^2)$ but it is worth to notice that n always remains small. The proposed Vi-DIFF algorithm detects changes between two Vi-XML pages. We intend in future work to extend this algorithm in order to detect changes between two versions of sites instead of two pages. The extended Vi-DIFF must produce, as output, a complete Vi-Delta describing changes between each page of the two site versions.

5.3 Changes Importance

Based on the Vi-Delta produced by Vi-DIFF, we propose a function that evaluates the importance of detected changes. This function depends on three major parameters:

• **Importance of updated block.** Typically, the most important information is on the center and the advertisements are on the header, etc. Thus, the importance of blocks can be assigned according to their relative location in the page.

This can be achieved, for instance, by Song and al. method [18]. Based on extracted spatial and content features of blocks, they use supervised machine learning algorithms to assign automatically importance value for each block. We can, also, take into account other parameters to evaluate the importance of a block with respect to the history of changes on this block. For instance, we can consider that the more frequent a block is changing, the less important it is. We are currently looking for the best technique to estimate the importance of blocks.

• **Operations Importance.** The importance of operations depends on the operation type (move, insert, etc.) and the changed element (link, image, etc.). For instance, insert or delete operations can be considered more important than a move. Also, inserting an image can be more important than inserting a link or a text. Again, we plan to study machine learning methods to choose the best parameters values for each operation type.

• **Changes Amount per Block.** The amount of change operations (delete, insert, etc.) occurred inside a block for each element (link, image and text) is deduced from the generated Vi-Delta. This amount represents the percentage of change operations detected for each block divided by the total number of block's elements.

Based on these parameters, we propose the following function $E(v_1, v_2)$ to estimates the importance of changes between versions v_1 and v_2, each composed of blocks Bk_i :

$$E = \sum_{i=1}^{N_{Bk}} I(Bk_i) * [\frac{1}{N_{Op}} \sum_{j=1}^{N_{Op}} I(Op_j) * \frac{1}{N_{El}} \sum_{k=1}^{N_{El}} \frac{N(Op_j, El_k)}{N(El_k, bk_i)}]$$

where:
- Op_j={insert, delete, update, move}
- El_k={link, image, text}
- N_{El}, N_{Op}, N_{Bk} are respectively the number of elements type, operation type and blocks in the page.
- $I(x)$ denotes the importance value of x which can be a block or a change operation. In order to normalize the result of function $E()$, we add the following constraint on the importance of blocks : $\sum_{i=1}^{N_{Bk}} I(Bk_i) = 1; 0 \leq I(Op) \leq 1$
- $N(Op_j, El_k)$ denotes the number of change operation j that occurred on the element k.
- $N(El_k, Bk_i)$ denotes the total number of elements k inside the block i.

The function $E()$ is computed by multiplying the percentage of changes, for each operation (Op_j) and block Bk_i, by the importance of operations $I(Op_j)$ and blocks $I(Bk_i)$. It returns a normalized value between 0 and 1.

Example 1.

Given the blocks importance as shown in Figure 4, the change operations, detected in a delta between two versions of pages, are: (i) an update of a text in block B_1; (ii) an insertion of 4 links in block $B_{2.2}$; (iii) a deletion of 2 images in block B_3. In this example, the operations insert and update are considered more important ($I(ins) = I(upd) = 1$) than a delete ($I(del) = 0.8$). In the old version of the page, the block B_1 has one text element, $B_{2.2}$ has 2 links and B_3 has 4 images. The importance of changes is computed as follows: $E = I(Bk_1) * I(upd) * [N(upd, text)/N(text, Bk_1)] + I(Bk_{2.2}) * I(ins) * [N(ins, link)/N(link, Bk_2)] + I(Bk_3) * I(del) * [N(del, image)/N(image, Bk_3)] = 0.1 * 1 * (1/1) + 0.4 * 1 * (4/(2+4)) + 0.2 * 0.8 * (2 /4) = 0.44$

In future work, we hope to extend this function to evaluate the importance of changes between two site versions by considering others parameters such as the page rank, the page depth in a site, etc.

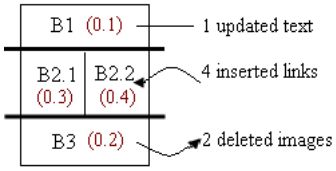

Figure 4: Changes Importance Example.

5.4 Web Crawl Scheduling

One of the goals of our approach is to optimize web crawling through scheduling. The task of the scheduler is to choose the most urgent document to be refreshed according to the history of changes. As mentioned in Section 3, the scheduler manages a list of documents ordered by a freshness urgency function. This function estimates, for each page, how urgent it is to refresh it at a given date. It takes into account the importance of changes (estimated by function $E()$) that have occurred between the original version and the last one archived. We define the following freshness urgency function for the scheduler.

$$U(doc, date) = p * \frac{Avg_E}{date_{lastUpd} - date_{OrigDoc}} * (date - date_{lastUpd})$$

where:
- p: priority of the page
- Avg_E: average of estimated changes importance between two versions of documents
- $date_{lastUpd}$: date of the last refreshed version
- $date_{OrigDoc}$: date of the first version of documents

This function depends on (i) the priority of the page, (ii) the importance of changes of already archived versions and (iii) the date of last refresh. We built a simulator to generate synthesized updates (with different changes importance) on simulated documents. Then, a scheduling strategy using the urgency function is developed and compared with two existing crawl policies (Round Robin [1], Cho [9] [2]). The preliminary results are promising. In fact, the strategy using the urgency function enables the event-driven crawler to retrieve more urgent/important version than the other policies. We hope also to use the importance of changes to assign an appropriate frequency for the first type of crawler (the frequency crawler).

6. PRELIMINARY RESULTS

Visual segmentation experiments have been conducted over HTML web pages by using our extended VIPS algorithm. We measured the performance of the visual segmentation in terms of execution time and output size. The execution time seems to be a little bit costly but is counterbalanced by the richness of the Vi-XML file that really simulates the visual aspect of web pages. Nevertheless, this time cost must be optimized. The main idea for that purpose is to avoid rebuilding the blocks structure for a page version if no structural change has occurred since the former version. The

[1]The documents are selected in circular order.

[2]The urgent document to be refreshed is the one which has Minimum($date_{lastUpd}$ + T_{CHO}). T_{CHO} [9] is the estimated period of updates.

experiment of output size files shows that the Vi-XML document size is usually about 30 to 50 percent less than the size of the original HTML document (for those sized more than 100 KB). This is interesting for the comparison of two Vi-XML documents since it can help to reduce the time cost of changes detection algorithm.

Experiments were conducted to analyze the performance of our proposed Vi-DIFF algorithm in terms of execution time and delta size. The total execution time is satisfying as it allows to process more than one hundred (currently sized) pages per seconds and per processor (requirements of CARTEC project). Others tests have been realized to measure the size of outputted delta. Results show that the size of the delta is always less than the size of one version of Vi-XML document, which is reasonable. To check the correctness of the delta, we build a simulator that generates synthesized changes on given Vi-XML documents. The simulator takes a Vi-XML file and it generates a new version according to the parameters given as input (proportion of blocks to be changed, for each operation type). It also generates the corresponding delta file that helps checking if the result of the Vi-DIFF is correct. We manually compared the delta generated by the simulator with the delta produced by our Vi-DIFF, with success.

The experiment of crawl simulation shows that the strategy (using freshness urgency function) that takes into account the importance of changes between versions improves considerably the web crawler. In fact, it enables to retrieve more important versions than other existing strategies. This preliminary result is very promising but further experiment must be conducted on real web pages and on complete sites.

7. CONCLUSION AND FUTURE WORKS

This thesis points out the issues of efficiently archiving web pages. Web archiving can waste time and space for indexing and storing unimportant changes on page versions. To address this issue, we propose an approach which combines three concepts; the visual page segmentation, the change detection and the importance of web pages' blocks in order to better detect important changes between versions. Other existing approaches for Web archiving are based only on the frequency of changes. The first step of our approach consists in constructing the visual structure of document based on semantic blocks by extending an existing algorithm VIPS [5]. The second step consists in detecting changes between the last version archived of a page and the former one. The Vi-Diff algorithm we designed for this phase is more adequate to the visual layout structure of documents than existing generic methods. Then, we study issues related to evaluating the importance of changes. This allows us designing a crawl scheduling strategy which uses the analysis of changes importance to optimize web crawling.

Preliminary tests on the segmentation and diff phases show that the execution time is promising. However, the time for segmentation is much higher than the time for comparing. In order to further optimize the system, we must focus on reducing the segmentation time. One idea is to assume that the block structure is evolving very rarely and to process the segmentation only when a change in the structure is detected. Another on-going work is to detect a move, separation and fusion of blocks as structural changes. Also, we intend to extend our approach in order to detect/analyze changes between two versions of sites instead of two pages which raises the issue of temporal consistency of page versions. In fact, further study is necessary to know what documents should be compared according to their depth and their changes in the site. The extended Vi-DIFF must produce, as output, a complete format of the Vi-Delta describing changes between the two site versions. Then, issues related to evaluating the importance of changes between two sites must be addressed. We intend, in a future step, to learn the appropriate change importance threshold to be fixed in order to decide indexing/storing the most useful site versions. In addition, some studies must be done to choose the best strategies to store the Vi-XML versions and the Vi-Delta. For instance, (i) storing only the latest version of the document and all the Vi-Deltas for previous versions or (ii) storing all versions of the documents and computing Vi-Deltas only when necessary.

Future works are related to the blocks importance. We are currently looking for the best machine learning technique to get automatically the relative importance of blocks and of change operations. For instance, we can assume that the more frequent a block is changing, the less important it is. For web crawl scheduling, we intend to perform experiments on real web pages and sites. We hope also to use the change importance between versions to assign an appropriate frequencies to the first type of web crawlers. The final goal of the web archiving is the consultation of web pages versions. Thus, further work must be done to temporally query the archive about the different web page versions and also about the visual/structural changes occurred between them. Querying visual/structural changes is important because it gives relevant information for understanding the dynamics of the web sites.

8. REFERENCES

[1] The Web archive bibliography, http://www.ifs.tuwien.ac.at/ aola/links/webarchiving.html.

[2] S. Abiteboul, G. Cobena, J. Masanes, and G. Sedrati. A First Experience in Archiving the French Web. In *ECDL '02: Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries*, 2002.

[3] H. Artail and K. Fawaz. A fast HTML web page change detection approach based on hashing and reducing the number of similarity computations. *Data Knowl. Eng.*, 66(2):326–337, 2008.

[4] D. J. C. Lampos, M. Eirinaki and M. Vazirgiannis. Archiving the greek web. In *4th International Web Archiving Workshop (IWAW04)*, Bath, UK, 2004.

[5] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. VIPS: a Vision-based Page Segmentation Algorithm. Technical report, Microsoft Research, 2003.

[6] C. Castillo and B. Sp. Scheduling algorithms for web crawling, 2004.

[7] W. Cathro. Development of a digital services architecture at the national library of Australia. EduCause, 2003.

[8] J. Cho and H. Garcia-Molina. 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.

[9] J. Cho and H. Garcia-Molina. Estimating frequency of change. *ACM Trans. Interet Technol.*, 3(3), 2003.

[10] G. Cobena, S. Abiteboul, and A. Marian. Detecting changes in XML documents. In *ICDE '02: Proceedings of 18th International Conference on Data Engineering*, 2002.

[11] C. N. Cosulschi M. and G. M. Classification and comparison of information structures from a web page. In *The Annals of the University of Craiova*, 2004.

[12] M. K. Evi, M. Diligenti, M. Gori, M. Maggini, and V. Milutinovi. Recognition of Common Areas in a Web Page Using Visual Information: a possible application in a page classification. In *the proceedings of 2002 IEEE International Conference on Data Mining ICDM'02*, 2002.

[13] D. Gomes, A. L. Santos, and M. J. Silva. Managing duplicates in a web archive. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, 2006.

[14] X.-D. Gu, J. Chen, W.-Y. Ma, and G.-L. Chen. Visual Based Content Understanding towards Web Adaptation. In *Second International Conference on Adaptive Hypermedia and Adaptive Web-based Systems (AH2002)*, 2002.

[15] R. La-Fontaine. A Delta Format for XML: Identifying Changes in XML Files and Representing the Changes in XML. In *XML Europe*, 2001.

[16] E. Leonardi, T. T. Hoai, S. S. Bhowmick, and S. Madria. DTD-Diff: A change detection algorithm for DTDs. *Data Knowl. Eng.*, 61(2), 2007.

[17] C. Olston and S. Pandey. Recrawl scheduling based on information longevity. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 437–446, New York, NY, USA, 2008. ACM.

[18] R. Song, H. Liu, J.-R. Wen, and W.-Y. Ma. Learning block importance models for web pages. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, 2004.

[19] Y. Wang, D. DeWitt, and J.-Y. Cai. X-Diff: an effective change detection algorithm for XML documents. In *ICDE '03: Proceedings of 19th International Conference on Data Engineering*, March 2003.