Adapting Web Archive Catalogues for Dynamic Change

Paul Wu Horng-Jyh
Nanyang Technological University
31 Nanyang Link, SCI Building,
Singapore 637718
(65)-67906110
hjwu@ntu.edu.sg

Ichsan Tamsir Putra
Nanyang Technological University
31 Nanyang Link, SCI Building,
Singapore 637718
(65)-67906110
Ichsan@ntu.edu.sg

Nguyen Ngoc Giang
Borland Software Corporation
2 International Business Park
#02-30/36, Singapore 609930
(65)-67256846
giang.nguyen@borland.com

ABSTRACT
Web archives are an important source of information. However, before a Web archive can be properly utilized, it needs to be catalogued. This is to ensure that the accessed materials yield the historical understanding intended by the researcher. At the same time, the dynamic nature of the Web will easily render these catalogues outdated, and there is a constant need to monitor when the Web catalogues become irrelevant upon change of the Web content. This means a substantial amount of human effort is required to maintain the catalogue records for the Web archives, adding additional burden to any institutions that maintain it. In this paper, we propose an automatic mechanism to monitor changes in Web content, so that human workload can be reduced. The system combines two component technologies to make this possible: (1) a contextualized annotation module and (2) an evidence change detection module. Contextualized annotation enables the cataloguing process to link content on the Web page (*the evidence*), to the value assigned for an element of a metadata schema. Thus, the metadata is "supported" by certain Web content that functions as evidence for a cataloguing decision. Regardless of changes in the webpages outside of the evidence, the metadata remains valid as long as all the evidence remains the same. In order to achieve evidence-specific change detection, we need to extend the traditional Longest Common Subsequence (LCS) based Diff engine using a Page Coordinate translation algorithm, which we argue, through a survey, is the first among many other Web content monitoring approaches.

Categories and Subject Descriptors
H.4 [Information Systems]: INFORMATION SYSTEMS APPLICATIONS

General Terms
Algorithms, Design.

Keywords
Web archives, evidence-based cataloguing, change detection.

IWAW'07, June 23, 2007, Vancouver, B.C.

1. INTRODUCTION
In our earlier paper, we demonstrated a Web archive cataloguing system, called Web Annotation for Web Intelligence (WAWI) [1]. In that paper, we highlighted four important qualities of a Web archive catalogue system using Web annotation. They are:

- Relating semantic content in the metadata to Web content
- Rendering agreement, disagreement and different granularity of evidence
-
- Providing flexible and precise annotation of the evidence
- Relating ontology to metadata in relational metadata.

We argued in the paper that these qualities provide the necessary data structure and processes to organize Web archive materials for historical research; a Singapore Ministry of Manpower Web archive was used as an example to demonstrate how that can be achieved.

We also observed that, in the context of a publicly-accessible Internet, these qualities are also compatible with the collaborative, post-custodial mode of cataloguing like the social tagging of techorati and social bookmarking of del.icio.us.

On the other hand, in the institutional context, systems incorporating these qualities can enable librarians and their managers to establish peer catalogue processes as demonstrated in the following Fig. 1:

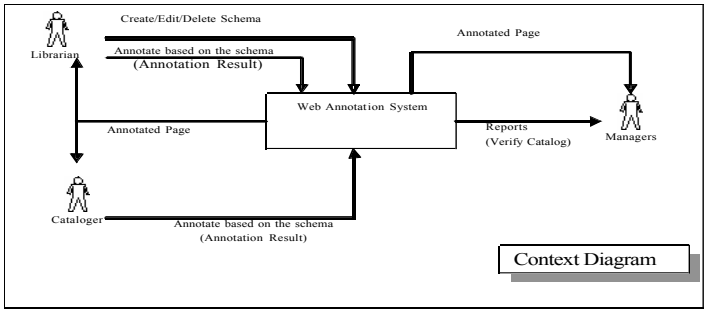

Figure 1. Context Diagram of WAWI Annotation and Cataloguing System

As shown in Fig. 1, there are three types of actors interacting with the annotation system. The Librarian is the actor who creates, edits, and deletes annotation schemata, in this case, Dublin Core

metadata. The Librarian can also catalogue and annotate the Web archive materials.

The Cataloguer is the actor who annotates the webpages based on the annotation schema created by Librarian. They can also retrieve and view the annotations and modify them. The last actor is the Manager. S/he can view the annotations done by cataloguers, and generate a management report from the annotation data; most importantly, s/he also ensures the quality of the catalogue by retrieving the annotated page and verifying that the evidence of the cataloguing decision is highlighted. The detailed processes and interaction between each actor in the annotation system is shown in Fig. 2, the Use-Case diagram.

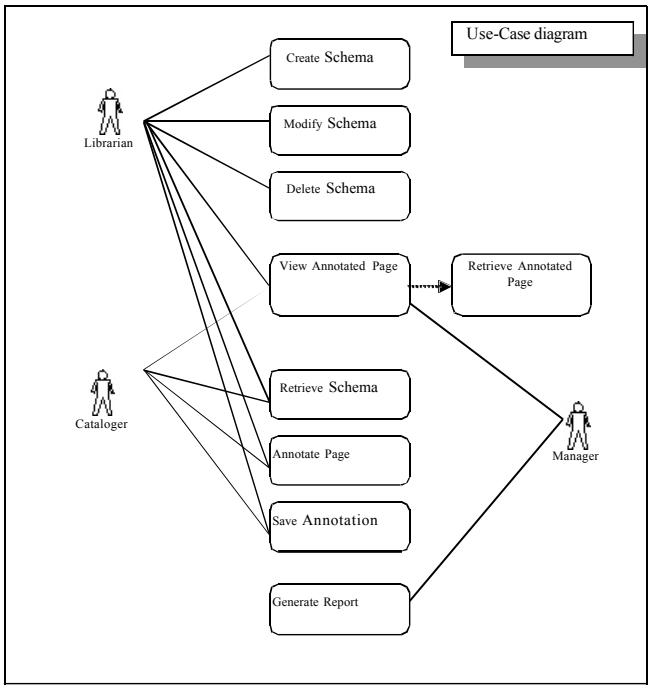

Figure 2. Use-Case Diagram of WAWI Annotation and Cataloguing System

Fig. 3 demonstrates the user interface where, during the verification stage, the left-hand panel displays the overlapping effects of differing evidence (highlighted in yellow). As shown, there are two opposing evidence annotated by two cataloguers in the right-hand frame. The left-hand panel will then render overlapping and non-overlapping highlighted text indicating the disagreement, based on the evidence annotated. By applying his discretion, the Manager may then advise that the annotation be adjusted so the appropriate evidence will be captured and associated with the metadata.

After the evidence associated with the metadata is registered together with the metadata, they can then be used to monitor the changes in the Web content and alert the Librarians. This association between Web content and metadata is unique in WAWI (context-aware), as compared to previous approaches to Web annotation (context-free). In the context-free Web annotation, change monitoring cannot be further specified except at the page level; this may cause too many alarms that are not relevant to cataloguing purpose.

Before we explain how the monitoring of Web content can be done, we shall review previous Web annotation systems, which have seemingly overlooked the need for this feature.

2. Review of Web Annotation Systems

As mentioned, most previous Web annotation systems are not able to perform change monitoring. This is largely because they are not designed for managing Web archives, which have a tendency to be updated frequently. Instead, they are designed for

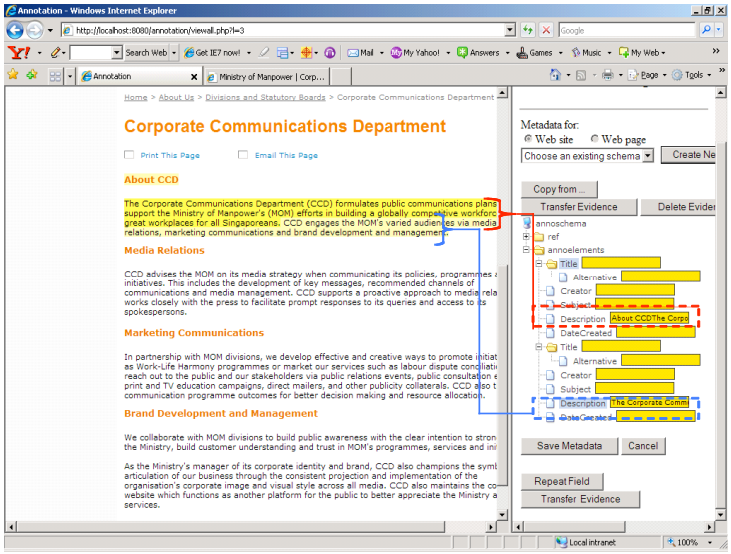

Fig. 3 Overlap annotation reflecting disagreeing evidence in collaborative cataloguing

communication among Web users and for schematising Web content (for Semantic Web). In fact, in the most general sense, Web annotation system is supposed to enable users to take notes on Web documents. Typically, an annotation system enables users to mark a point or portion of a Web page by highlighting it; users can then write down their thoughts with respect to the marked-up portion, which is then shared among others. Specifically, Web annotation system also makes it possible to have collaborative working environments where every member can be writer or reader as they wish [2].

Other perspectives as to how Web annotation systems can benefit the user community are observed by Paul Perry. [3] Web annotiation can provide:

- a trace of use
- third party commentary
- information sharing
- information filtering
- semantic labeling of document content
- enhanced search

These functionalities may be equally applicable for purposes other than collaboration. Indeed, in the past, Web annotation systems have been applied in different domains, which include eLearning [4], Digital Library [5], Semantic Web (as an authoring tool for semantic information) [6], Online Discourse and Forum [7], Community annotation [8], and Web archives [9, 10]. In view of the recent Web 2.0 movement, however, Web annotation is becoming more critical as it empowers the readers with a capability that was previously only available to the authors, thus making Web publication altogether more interactive and participatory.

Among the features available in the above Web annotation systems, the following functionalities are relevant to our design of the WAWI system:

1. Annotation Granularity: Point, Pair-of-Points, Text-node Range, and Open Range.
   Annotation can be marked as a point in the Web page or as a pair of points. It can also be marked as "a range/segment" of text, some restricted to the span of a text node in a DOM-tree, while others allow the segments to start and end at any point in a text node.
2. Annotation Directionality: Uni- or Bi-directional
   annotation can provide a pointer to the Web content; conversely, the Web content can point to the annotation as well. These are uni-directional annotations (between Web annotation and Web content). On the other hand, if the pointers can be provided both from the Web annotation to Web content, and vice versa, then either one can be accessed through their counter part. This system then provides bi-directional accessibility.
3. Annotation Complexity: Simple, Hierarchical, or Nested (Overlapping).
   An annotation can be made to overlap (Nested) with others or be totally subsumed by another (Hierarchical). Or it can be simple, with the annotations not relating with each other.
4. Ability to structure Web content:
   If a Web annotation is designed for communication purposes alone, structuring of Web content is not a priority as communication may take on many forms. On the other hand, if the annotation is designed for giving semantic information to Web content, then there is a need to provide Web content structuring functions based on a "schema" that can capture the essence of a domain.
5. Access to Web annotation: Search and Browse.
   Annotation data can be indexed and made searchable and browsable, as required by many similar database application systems, to provide analysis results or business intelligence.
6. Granularity of Change Detection: Web page or Web annotation alone.
   Web pages are a very dynamic medium and many of them undergo change frequently. Systems can be sensitive to changes at the Web page level; that is, if any part of the Web content changes, the changes will be detected. Alternatively, detection can be set at the Web annotation level, in which case only changes in Web annotation will trigger the alerts.

Table 1. Functionality Comparison of Web Annotation Systems

	Annotea	**CritLink**	**Marginalia**	**WAWI**
1. Granularity	Point & Node Range	Pair of Points	Open Range	Open Range
2. Directionality	Uni - directional	Bi - directional	Bi - directional	Bi - directional
3. Complexity	Simple	Simple Nested	Simple & Nested Hierarchical	Simple & Nested & Hierarchical
4. Access Annotation (Search/ Browse)	Browse	None	Browse	Search & Browse
5. Ability to structure Web Content	None	None	None	Yes
6. Granularity of Change Detection	None	None	None	Page Website

As shown in Row 1 of Table 1, WAWI has the degree of granularity that allows the annotator to specify an arbitrary open range segment of Web content, instead of restricting annotation to just certain HTML-delimited segments/nodes of Web pages. This features combine flexibility and specificity that enable the annotation, and the evidence assignment, to be precise for monitoring the change.

Row 2 of Table 1 demonstrates that WAWI can link Web content to annotation/metadata (Uni-directional), which is required for the purpose of monitoring change in Web archives. It can also link Annotation/Metadata to Web content; this is useful, as demonstrated in Fig. 2, when the cataloguer inspects the annotation done by the librarians.

As demonstrated in Fig. 3 with different shades of the yellow color, Row 3 of Table 1 states that WAWI is able to render complex annotation patterns: simple, nested and hierarchical.

WAWI also provides search and browse front-end as indicated in Row 4 of Table 1.

The schema shown in Fig. 3 is used to structure the Web page content according to it. Thus, Row 5 of Table 1 indicates that WAWI can help structure Web content according to schema.

Row 6 of Table 1 concerns the features WAWI has on change detection. This will be discussed in the following section.

3. WAWI System Architecture

WAWI consists of three main modules, one on the server sides and two, the client sides:

1. Annotation Module (client)
2. Diff View Module (client)
3. Change Detection Engine (server)

This is demonstrated in the following figures Fig 4.1 and Fig 4.2.

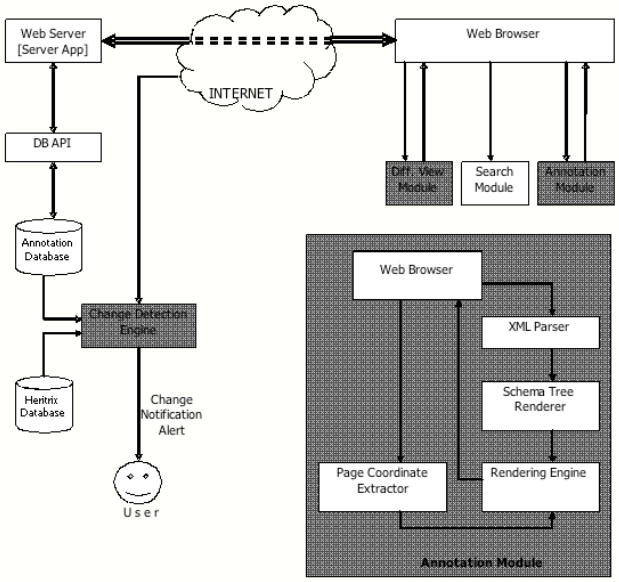

Figure 4.1: WAWI System Architecture (I): Overall and Annotation Module

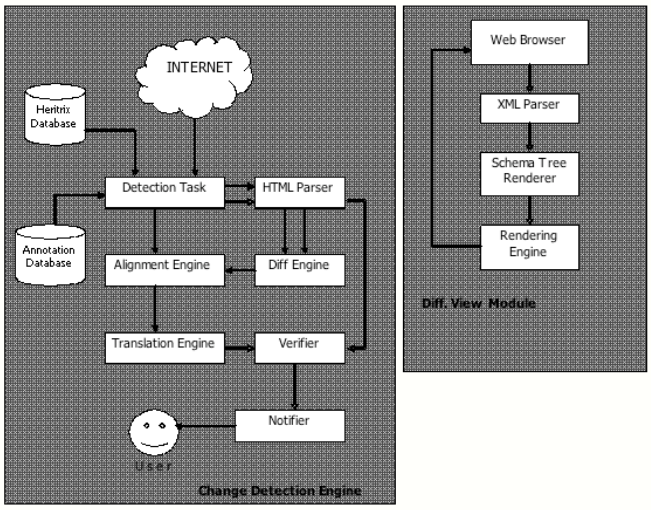

Figure 4.2: WAWI System Architecture (II): Change Detection and Diff View Module

As shown in Fig. 4.1, WAWI takes inputs from the Annotation Database – containing the annotation/metadata records – and the Heritrix Database – containing the archived Web pages. The annotation/metadata records, structured as an XML-document, and the archived Web page, are sent to the client side where the Annotation Module will render the annotation by overlaying it over the archived Web page; the archived Web page remains intact. The annotation/metadata records are rendered in the tree view using a Schema Tree Renderer component. When user annotates the evidence by highlighting Web content, the Page Coordinate Extractor will compute the Start and End positions of the highlighted area along the Page Coordinate (PC) of the Web page – the so called SE PCs of the annotation. The SE PCs of the annotation are passed to the Rendering Engine to render the annotated evidence in yellow-background text as shown in Fig. 3. Before any annotation takes place, the metadata records do not contain any SE PCs, as no Web content (the evidence of annotation) has been made to associate with the metadata records. When the annotation is made, the SE PCs of the evidence, together with the annotation information, will be sent back to the server and saved in the Annotation Database.

A Page Coordinate (PC) is defined over the text array that is constructed through a depth-first traversal the DOM-tree of the Web page. The depth-first traversal omits the web document element structure, and serialize - concatenate one after the other - the text in the text nodes as a text array. Identifying the PCs of the highlighted/annotated text is known as the decoding process. The Rendering Engine works by reversing the decoding operations: given the PCs, it encodes the annotation on the Web page by highlighting the annotation.

As shown in Fig. 4.2, the Change Detection Engine works by taking inputs from Heritrix Database – the archived Web page, Annotation Database – the annotation record associated with the Web page, and the live Web page from INTERNET. The two Web pages, live and archived, are then parsed by an HTML parser where the DOM structures, and subsequently, the Page Coordinates of both Web pages, are constructed. The Page Coordinates of the live and archived Web page are then processed by a Diff Engine where the deltas, in terms of insertion, deletion and movement of text segments, are computed. These deltas will allow the Page Coordinates of both the archived page and the live one to be aligned by the Alignment Engine; as a result, the specific position of one PC (e.g. of the archived page) can be mapped to that of another PC (e.g. of the live page). When this mapping is achieved, the Translation Engine can then translate the SE PCs of the annotation evidence in the archived page to those in the live one. This in turn allows the Verifier Module to identify whether an alert needs to be sent to the cataloguer for potential change required for the catalogue/annotation records. The setting of the change detection rules and the alert messages sent by Notifier Module are shown in Fig. 5.1 and 5.2, while the technical details of how the various engines work will be explained in Section 4.

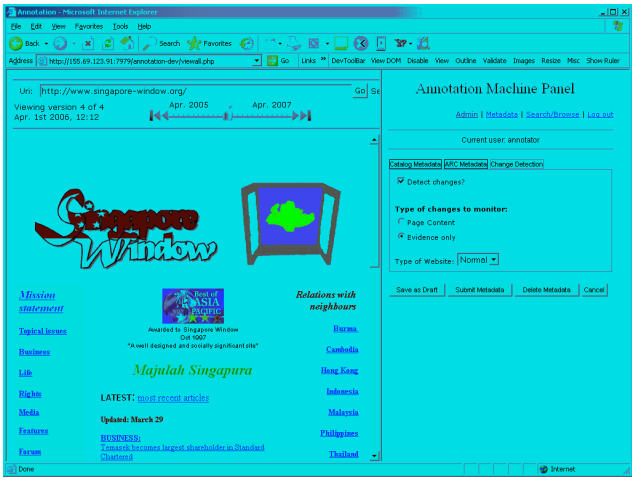

Figure 5.1: WAWI Change Monitoring Interface

Dear Mr Jones,

The following URLs have been changed from the last crawl. You can click on links below to see the differences.

http://www.singapore -window.org/

http://akikonomu.blogspot.com/2006/03/tired -election -strategies.html

This is a computer generated email. Please do not reply.

Figure 5.2: WAWI Email Alert of Change Detected

As demonstrated in Fig. 5.1, the Change Monitoring rule can be set to be sensitive to "Page content" or "Evidence only." If it is set to "Page content," then any update on the Web content will trigger an alert, while if it is set to "Evidence only," then alert will be sent only when the evidence-relevant portion of the content is changed. In this case, the rule is set to be "Evidence only."

The archived and live pages, together with their respective SE PCs, are sent to the client, where the Diff View Module shown in Fig. 4.1 will then render the effects of the Change Detection graphically, as demonstrated in the Fig. 6.

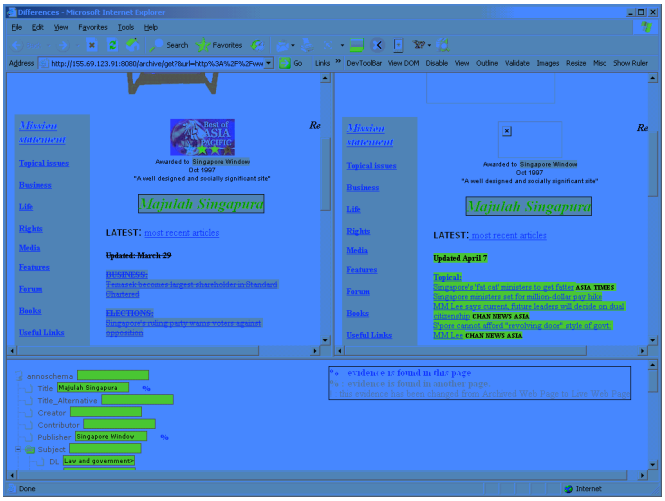

Figure 6: WAWI Change Detection Front-end

As shown in Fig. 6, the upper left-hand panel contains the archived page and its associated evidence highlighted in yellow. Portions of text are also shown to be stricken out; these are the Web content that existed in the old/archived page and have since been deleted from the current/live page. The upper right-hand panel contains the live page. The green portion of the text reflects the Web content that is new in the current/live page but was not in the old/archived page. Evidence of annotation remaining in the live page is displayed similarly as yellow text.

The lower panel displays the annotation/metadata record of the Web page, which is similar that in the Annotation Panel of Fig. 3. The "%" sign by the Title and Publisher elements (of a Dublin Core metadata) indicate that the evidence for them is present in the displayed pages. Similarly, as indicated by the message box at the upper right-hand of the lower panel, a "% !" sign indicates that the evidence of the element is not only present in this Web page, it is also being changed, the effect of which causes the alert to be sent based on the Change Monitoring rule as set in Fig. 5.1.

WAWI's change detection mechanism, as demonstrated Fig. 4.2, is based on Alignment and Translation engines. Previous approaches to change detection, such as AIDE, WebCQ and WebVigiL, are based on a very different design. We shall turn to the discussion on the Change Detection design considerations in the next Section.

4. Change Detection Engine Design Considerations

Tracking the evolution of Web resources is critical in Web archive applications. To do so, we need a sophisticated change management system to evaluate the progression of evidence annotated by users. A change management system typically consists of two main components: *difference generation* (performed by the Diff Engine of Fig. 4.2) and *change evaluation* (performed by the Alignment, Translation and Verifier Engines of Fig. 4.2). The first component identifies the differences between documents of interest, while the second one examines whether the differences found in the first step are worth notifying cataloguers.

4.1 Review of Change Detection Approaches

Most previous research on Web content change detection has been based on the technique called Longest Common Subsequence (LCS) [11]. LCS compares two text documents, modeled as two sequences of tokens, and identifies the subsequences that have the longest length in common. The output of this algorithm is a list of Difference elements like the following:

```
<differences>
     <diff type="delete" start="10" length="6"/>
     <diff type="insert" start="34" length="3"/>
</differences>
```

The above means that a 6-characters segment has been deleted from position 10 of the old sequence and a new 3-characters segment is being inserted at position 34 of the old sequence.

Web content change detection systems differ largely in the following two aspects: (1) whether a limit is set to improve performance, by narrowing down the area of concern to be

compared, and then how areas of concern are defined and computed, and (2) how the sequence of tokens is defined when LCS technique is applied.

AIDE (AT&T Internet Difference Engine) [12] treats the entire HTML document as the unit of comparison, thus no limit is used to narrow the area of comparison. It defines the sequence of tokens based on the source of the HTML document, as described in Douglis et al: "the source page of the HTML document is being parsed into a sequence of sentences and 'sentence-breaking' markups (such as <P>, <HR>, <LI>, or <H1>), where a 'sentence' is a sequence of words, and certain (non-sentence-breaking) markups (such as <B> or <A>). A 'sentence' contains at most one English sentence, but may be a fragment of an English sentence. All markups are represented and are compared, regardless of whether or not those markups are 'content-defining'. LCS is then applied to these sequences."

WebCQ [13] attempts to limit the detection areas requiring LCS comparison by applying object-extraction techniques. An object is a portion of the HTML document of concern and WebCQ allows the following types of object change to be defined:

- *Any change:* Any update on the page object
- *Link change:* Any change to links of a page
- *Image change:* Any change to images of a page
- *Words change:* Any change to words of a page
- *Phrase update:* Any change to a selected text phrase
- *Table change:* Any change to the content of a table
- *List change:* Any change to the content of a list
- *Arbitrary text change:* Any change to the text fragment specified by a regular expression

The Object extraction module consists of iterations of regular expression (fuzzy and exact) matching steps.

Further optimization on Object extraction is attempted in WebVigiL [14, 15]. The algorithm CH-DIFF of WebVigiL introduces the concept of window-based change detection, where the window patterns were indexed and then retrieved (rather than matched on-the-fly) when the change detection is being computed, and the area of concern is also identified by Object types. These Object types are extracted from Web pages using Knuth-Morris-Pratt (KMP) string-matching algorithm, and changes are detected as the increase or decrease of objects in a window. If the number of Objects to be detected is large, WebVigiL may not scale well due to the overhead incurred in indexing the Objects.

4.2 WAWI Change Detection Approach

Compared with the previous approaches to Change Detection: AIDE, WebCQ, and WebVigiL, WAWI change detection has the following unique features:

(1) The sequence of tokens for change detection is defined over the Page Coordinate that is generated by depth traversal in the DOM-structure of the Web pages.

(2) The change detection for specific evidence is achieved through alignment and translation of Page Coordinates rather than by Object extraction; as such, no matching or indexing operations are required.

Due to (1), WAWI is different from AIDE, which may detect changes that are not significant for Web archive cataloguing purposes. This is because most of the cataloguing decisions are based on the content of the Web pages rather than the particular tags of the pages. AIDE's sequence of tokens, derived from the HTML source directly, may contain HTML tags.

Due to (2), WAWI is distinct from WebCQ or WebVigiL in that no matching or indexing operations are required. Nonetheless, WAWI obtains the same degree of specificity of change detection that was achieved by WebCQ and WebVigiL. Unlike changes in HTML tags, the specificity of change detection based on evidence annotated into the catalogue records is desirable for change detection in Web archive cataloguing. The specificity is achieved by the Translation module, an innovation among the approaches reviewed in this paper, which is able to map the evidence Page Coordinates in the archived page to those in the live page. Therefore the change can be detected, in terms of insertion, deletion and movement, for the Web page as a whole, or for those specific to the evidence only.

4.3 The Page Coordinate Translation Engine

It is crucial to recognize that the concept of evidence is associated with the Page Coordinates of a Web document. Therefore, monitoring the change of evidence requires a mapping from one coordinate system to another. In this section, we present the algorithm to perform such mapping.

As explained in our discussion on WAWI architecture shown in Fig. 4.2, the Translation engine is applied when the differences between two documents have been computed by the Alignment engine. Alignment engine represent the Difference objects in an XML document, for example:

Figure 7: Examples Explaining Translation Engine at Work

```
<differences>
    <diff type="delete" start="10" length="6"/>
    <diff type="insert" start="34" length="3"/>
```

</differences>

For the delete type, the *start* attribute is the position of the first character of the removed string. For insert type, the *start* attribute is the position of the last common character between the archived and live pages; all are based on the Page Coordinate of the archived page. Our objective then in the Translation engine is to compute the coordinates of the evidence based on the Page Coordinate of the live/new Web page.

To elaborate the algorithm, let us consider one simple example. The original string is '*quick brown fox jumps over lazy dog*', with '*quick brown fox*' being highlighted as evidence. In the latest version, the string has become '*quick fox jumps over fat dog*'.

As shown in Fig. 7 above, Rows 1 and 2 concern the archived Web page content and its Page coordinate; Rows 4 and 5, the new Web page and its Page coordinate. The yellow portions are the evidence. Those that are deleted are shown as stricken out sequences, such as "brown" and "lazy" in Row 2. Those that are inserted as shown as green, such as "fat" in Row 5. In other words, the Diff Engine

will report the following 3 change entries, as expressed in the following XML document:

<differences>

<diff type="delete" start="5" length="5"/>

<diff type="delete" start="22" length="4"/>

<diff type="insert" start="21" length="3"/>

</differences>

With this, a *combination line* is constructed - shown in Row 7 - as the array of positions, ordered by the Page Coordinate of the live/new Web page, in the Page Coordinate based on the archived/old Web page. For example, the character '*f*' of "*fox*" occupies the 5th position in the new PC – shown in Row 5; in combination line, it has the value 10, which is its position based on the PC of the archived page; this is because of the deletion of "*brown*", which occupies position 5 to position 9 based on the old PC. As a result, we can map the coordinate pair (0, 12) in old PC to (0, 7) in new PC, resulting in the array (0,1,2,3,4,10,11,12) – the first 8 positions in the combination line. Similarly, by considering the deletion and insertion as shown above one by one, the combination line is thus constructed as (0,1,2,3,4,10,11,12, 13,14,15,16,17,18,19,20,21,21,21,21,26,27,28), where the repetition of 21 by an additional 3 times is due to the insertion of "*fat*" and deletion of "*lazy*".

The Combination line construct is not just useful in change detection; with the extract positions of evidence in the new Page Coordinate computed, the Diff View Module can then display correctly the highlight for the evidence when a side-by-side comparison is displayed to users.

The *combination line construction* algorithm is given in the following pseudo-code:

```
sort all diff entries incrementally by start attribute
current_end := 0
for each <diff type="TYPE" start="s" length="l"/> entry
    if first entry
        init combination := [0, ..., s-1]
    else
        append [current_end, ..., s-1]
    end if
    if TYPE = insert
        if (current_end <= s)
                append l+1 times value s
        else
                append l times value s
        end if
        current_end := s+1
    else
        current_end := s+1
    end if
end for
append [current_end, ..., len1]
```

Further, we need to align the combination line with new document's coordinate system so that the coordinates of the evidence can be translated into those in the new Web document. The *page coordinate translation* algorithm is demonstrated below:

```
for each entry <evidence start="p" end="q"/>
    p1 := 0
    while (p1 < p)
        advance p1 in the combination line
    end while
    if (p1 > q)
        report that this evidence has been completely
        removed
        continue with the next entry
    end if
    q1 := p1
    while (q1 < q)
        advance q1 in the combination line
    end while
    if (q1 > q)
        q1 := previous value of q1
    q2 := position of q1 in the combination line
    insert <evidence start="p1" end="q2"/> to the AG of
    new document
```

Since both the combine line construction and page coordinate translation algorithms are of linear complexity, theoretically the efficiency of the change detection in WAWI is determined by the original DIFF algorithm. In this case, as compared to WebCQ and WebVirgiL, WAWI might be more complex; however, the constant and additional computation routines required may add up for WebCQ and WebVirgiL so that in practice the difference may be limited, especially when more patterns are involved in the

monitoring routine where additional processes in indexing and pattern window matching are incurred. Nonetheless, the benchmark remains to be performed so to ascertain the actual performances of the different algorithms.

5. Conclusion

WAWI is a Web annotation system developed specifically to catalogue Web archives. It is designed not only to organize the Web archives for historical study, as it provides annotation that links metadata with Web content, but with the Web content annotated as evidence, WAWI can also be adapted to fulfill a unique capability to automatically monitor the change of Web content that may require an update of the Web metadata records. WAWI achieves this capability by integrating two engines, Alignment and Translation engines, to enable it to perform evidence specific change detection with ease, as compared to previous approaches adopted in WebCQ and WebVigiL, where complex operations of indexing and matching are required. Furthermore, WAWI is also more precise in detecting change as its Translation engine works based on the concept of Page Coordinate rather than on the source of HTML documents. By implementing these two design considerations, we argued that WAWI has effectively been adapted to catalogue Web archives materials, overcoming the substantial manual efforts required to monitor the ever-evolving Web content in the INTERNET.

6. ACKNOWLEDGMENTS

Our thanks are due to National Library Board of Singapore for grants partially supporting this research.

7. REFERENCES

[1] Wu, H-J P. and Heok, K-Y A. Annotating Web Archives – Structure, Provenance and Context through Archival Cataloguing. *New Review of Hypermedia and Multimedia* (Accepted, In Press)

[2] Heck, R.M., Luebke, S.M., & Obermark, C.H. (1999) *A Survey of Web Annotation Systems*. Retrieved June 3, 2005 from Grinnel College Website: *http://www.math.grin.edu/~rebelsky/Blazers/Annotations/Summer1999/Papers/survey_paper.html*

[3] Perry, P. (n.d.). *Web Annotations*. Retrieved June 3, 2005 from PaulPerry Website: *http://www.paulperry.net/notes/annotations.asp*

[4] Desmontils, E., Jacquin, C., & Simon, L. (2004). Advances in Web-Based Learning – ICWL 2004 (Vol 3143). Chapter 8: Dinosys: An Annotation Tool for Web-Based Learning. Heidelberd: Springer Berlin

[5] Silverman (n.d.) *The Annotation Engine*, from Harvard Law School's Berkman Center for Internet and Society, can be found online at: *http://cyber.law.harvard.edu/projects/annotate.html*

[6] Koivunen, M.R. (2005). Annotea Project. Retrieved March 14, 2006 from W3C Website: http://www.w3.org/2001/Annotea/

[7] Yee, Ka-Ping. (2002*). Zest: Discussion Mapping for Mailing Lists. Proceedings of the ACM Conference on Computer-Supported Cooperative Work*. Retrieved May 14, 2006 from CiteSeer Website: http://citeseer.ist.psu.edu/cache/papers/cs/30352/http:zSzzSzzesty.cazSzpubszSzyee-zest-cscw2002-demo.pdf/yee02zest.pdf

[8] Yee, Ka-Ping (2002). CritLink: Advanced Hyperlinks Enable Public Annotation on the Web. Demonstration. Proceedings of the ACM Conference on Computer-Supported Cooperative Work. Retrieved May 14, 2006 from Ka Ping Yee Website: http://zesty.ca/crit/yee-crit-cscw2002-demo.pdf

[9] Paul H. J. Wu, Adrian K. H. Heok, Ichsan P. Tamsir (2006). Annotating the Web Archives – An Exploration of Web Archives Cataloging and Semantic Web.

[10] Paul H. J. Wu, Adrian K. H. Heok, Ichsan P. Tamsir (2006). Applying Context-Sensitive Web Annotation in Evidence-based, Collaborative Web Archives Cataloging. International Workshop on Archiving Web (IWAW 2006). Spain.

[11] James W. Hunt and M. Douglas McIlroy (June 1976). "An Algorithm for Differential File Comparison". *Computing Science Technical Report, Bell Laboratories* **41**.

[12] Fred Douglis, Thomas Ball, Yih-Farn Chen, and Eleftherios Koutso_os. The AT&T Internet Difference Engine: Tracking and viewing changes on the web. *World Wide Web*, January 1998. To appear. Also published as AT&T Labs.Research TR 97.23.1, April, 1997

[13] Ling Liu , Wei Tang , David Buttler , Calton Pu, Information Monitoring on the Web: A Scalable Solution, World Wide Web, v.5 n.4, p.263-304, 2002

[14] Jacob, J, A. Sachde, and S. Chakravarthy, CX-Diff: A Change Detection Algorithm for XML Content and Change Presentation Issues in WebVigiL, in the Proc. of XSDM Workshop, Chicago, 2003, pp. 273--284.

[15] Jacob, J., et al., WebVigiL: An approach to Just-In-Time Information Propagation In Large Network-Centr environments, in Web Dynamics Book. 2003, Springer-Verlag, 2004.