Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 157 (2006) 147–166
www.elsevier.com/locate/entcs

How Recent is a Web Document?

Bo Hu[1] Florian Lauck[2] Jan Scheffczyk[3]

Universität der Bundeswehr München, Munich, Germany

Abstract

One of the most important aspects of a Web document is its *up-to-dateness* or recency. Up-to-dateness is particularly relevant to Web documents because they usually contain content origining from different sources and being refreshed at different dates. Whether a Web document is relevant for a reader depends on the history of its contents and so-called external factors, i.e., the up-to-dateness of semantically related documents.
In this paper, we approach *automatic management* of up-to-dateness of Web documents that are managed by an XML-centric Web content management system. First, the *freshness* for a single document is computed, taking into account its change history. A document metric estimates the distance between different versions of a document. Second, *up-to-dateness* of a document is determined based on its own history and the historical evolutions of semantically related documents.

Keywords: Web site management, up-to-dateness, content management systems, document metric, semantic links

1 Introduction

The WWW has been designed for dynamic information from the very beginning [1]. *Up-to-dateness* [4] is one of the most significant characteristics of Web documents, because a Web site typically contains numerous Web pages origining from different sources and evolving at different rates. Unlike books in a traditional library, Web pages continue to change even after they are initially published by their authors [2]. In this paper, we distinguish between *freshness*,

[1] Email:bo.hu@unibw.de
[2] Email:florian.lauck@unibw.de
[3] Email:jan.scheffczyk@unibw.de
[4] Also called "recency" or "freshness."

1571-0661/$ – see front matter
doi:10.1016/j.entcs.2005.12.052

which depends on the history of a single document, and *up-to-dateness*, which also takes into account semantically related documents.

Many works have explored measures of Web documents "from a search engine perspective" [18]. It has been found out that usually Web documents change trivially or in their markup only [5]. On the other hand, news pages containing "breaking news" change their content frequently and significantly.

This update heterogeneity severely impacts Web content management systems (WCMS), which should alleviate the continual maintenance of Web documents. More often than not, WCMS pretend an increased "freshness" of a Web page that changed gradually only. Worse, this notion of freshness is not application specific. Changes may be of syntactic or semantic nature. Syntactic changes can reflect editing efforts or improved readability (although the semantics is not changed). Semantic changes can increase the relevance of a document to specific purposes. We find related areas for semantic methods in text classification, retrieval, and summarization [19,13,9,17,7]. Since the date of the "last-modifying" is used for filtering and sorting, it is just fair to authors and readers if the WCMS computes the *freshness* of a Web document automatically using an algorithm that takes into account the *degree* of changes w.r.t. the application at hand.

Due to the importance of Web sites and the increasingly complex and collaborative Web publishing process, versioning of Web documents is an essential feature of WCMS [11]. Since the history of each document is available in such a system, a history-aware metric of changes can be implemented. This metric is essential if the freshness of a document should be estimated automatically or some versions should be vacuumed to free space [4].

In this paper, we present an approach to calculate the freshness of a document automatically based on its history. An important parameter is a *document metric*, which measures how much a document has been changed. This metric may be of syntactic or semantic nature and can be tuned to specific applications. By our syntactic metric not only the plain text information but also the markups are compared to each other. In this way, the real human resource usage for a Web document can be reflected. Our semantic metric employs Latent Semantic Analysis (LSA) as implemented in the General Text Parser (GTP) [7]. That way, we can analyze how much the content of a Web document has changed, regardless of presentation style modifications. We have implemented our approach in our WCMS [10], in which an XML structure represents a whole Web site, where each leaf stands for a Web page, containing further XHTML markup. Since XSLT pre- and post-processing are involved, the document metric can be easily adapted to special situations for creating and updating the document. Particularly, we have applied our

Pattern	Change Frequency	Change Extent	Change Content	Usage
News page	hourly	large	text / markup	commercial
Home page	monthly / yearly	small	text / markup	private
Boards	minutely / hourly	large	text	private
Online stores	minutely / hourly	large	text	commercial
Enterprise site	monthly / yearly	small	text / markup	commercial
WCMS	minutely / hourly	medium	text	private / comm.

Table 1
Change patterns for Web documents

approach to Chinese Web documents.

Whether a Web document is relevant for a reader depends not only on the document's own history but also on so-called external factors, i.e., the historical evolutions of semantically related documents. This proves useful, e.g., for news pages that change frequently. In our setting, semantic relations between documents [8,14] cover aspects like "is translation," "provides background information," "is essential part," or "is recommended reading." Therefore, we also calculate the *up-to-dateness* of a document w.r.t. the historical evolutions of semantically related documents.

The aim of our approach is to provide a language- and topic-independent algorithm that determines real up-to-dateness of documents in a WCMS. In addition, old versions of a document without significant contribution to up-to-dateness of the current version (or any version in the future) might be vacuumed to free space. The contribution of this paper is a flexible approach to calculate the up-to-dateness of documents based on their own history and on the history of semantically related documents. The major enabling factors are version control and explicit semantic links. The most significant parameter is a document metric, which can be tuned to specific applications.

From here, we proceed as follows. In Sect. 2 we introduce the running example for this paper by which we illustrate our approach. We address the freshness of a single document in Sect. 3. Sect. 4 describes the implementation of our approach in our XML-centric WCMS. In Sect. 5 we approach up-to-dateness of a document w.r.t. semantically related documents. We conclude this paper and sketch directions for future research in Sect. 6.

2 News Pages — A Challenge for Up-to-dateness

In our experiments we found typical patterns regarding the modifications of Web documents (see Tab. 1). For this paper, we choose the news page pattern, which is particularly suitable because it is characterized by an hourly change

Version 1:
Ukrainians Hit Polls to Elect President
KIEV, Ukraine - Rival candidates Viktor Yushchenko and Viktor Yanukovych faced off Sunday in a repeat election triggered by a fraudulent runoff vote and massive protests that resulted in an unprecedented third round in Ukraine's fiercely waged presidential contest. ...

Version 2:
Ukraine Elects President in Runoff Vote
KIEV, Ukraine - Rival candidates Viktor Yushchenko and Viktor Yanukovych faced off Sunday in a repeat election triggered by a fraudulent runoff vote and massive protests that resulted in an unprecedented third round in Ukraine's fiercely waged presidential contest. ...

Version 6:
Ukraine Holds Presidential Vote a 3rd Time
KIEV, Ukraine - Rival candidates Viktor Yushchenko and Viktor Yanukovych faced off Sunday in a repeat election that all sides hoped would resolve Ukraine's fiercely waged presdential contest after fraud wrecked one vote and prompted massive protests that deeply divided the nation. ...

Version 7:
Exit Polls Give Yushchenko the Presidency
KIEV, Ukraine - Three exit polls projected Ukrainian opposition leader Viktor Yushchenko the winner by a commanding margin over Prime Minister Viktor Yanukovych in Sunday's fiercely fought presidential rematch. ...

Fig. 1. Version history of an article about the Ukrainian President's Vote

combined with a large extent of textual alteration. In addition, news pages show extensive semantic interrelation. The usage of our approach is, however, not limited to news pages. Up-to-dateness of any document under version control can be estimated. In a collaborative authoring environment even the partial contribution of each author to a document section can be calculated as well.

Our running example was taken right out of practice and hence reveals large actuality: The President's Vote in the Ukraine on December, 26 2004. This day constituted of a large amount of breaking news worldwide, concerning forecasts, results, background information, and opinions. Starting hourly from 12.00 CET to 21.00 CET the sources of five different news pages (CNN.com, MSNBC.com, YahooNews.com, USAToday.com, derstandard.at) were downloaded and saved to a database, in order to later apply our approach. Each download represents a different version of a news page.

Fig. 1 illustrates the version history of the first paragraph of the YahooNews.com breaking news Web page about the Ukrainian President's Vote. [5] Changes between versions range from correcting typos or changing the layout towards dramatic changes in content. For example, we see a slight change between versions 1 and 2, whereas between versions 6 and 7 the article was rewritten almost completely. Clearly, the freshness of this news page should

[5] URL story.news.yahoo.com/news?tmpl=story&u=/ap/20041226/ap_on_re_eu/ukraine_election&e=1&ncid=

represent these facts.

3 Freshness of a Single Document

Up-to-dateness of a document depends on two aspects: its own freshness and the historical evolutions of semantically related documents. In this section, we only deal with the freshness of a single document, which can be seen as a document property based on the historical development. Up-to-dateness w.r.t. semantically related documents is dealt with in Sect. 5.

3.1 Approaching Freshness

For a document with only a single version the freshness is given by the time t_1 at which the document was saved. If a document has multiple versions $1, \ldots, n$, its freshness t^* can be expressed as a weighted average time stamp of all versions:

$$t^* := \frac{\sum_{i=1}^{n} t_i \cdot c_i}{\sum_{i=1}^{n} c_i}$$

where c_i, the *contribution* of the version i to the end version n, is still to be determined. A common and trivial way is to set $c_n = 1$ and for all previous versions $c_i = 0$ $(i < n)$. That way, we have the "last-modified" time stamp, which ignores the contributions of all past versions.

To take into account the contributions of all versions, one must know how much a document has been changed from one version to another one, which we call the *distance* $D_{i,j}$ between versions i and j of a document. Since all Web pages are under version control in our WCMS, document metrics can be easily employed for calculating such distances. Recall that our approach is parametric in the document metric D. For example, we can calculate the syntactic distance between two versions of an XML document by analyzing the modifications of XML nodes using an XML diff implementation such as Microsoft XMLDiff [15]. By simply defining $D_{i,j}$ as the number of XML node modifications to change the content of version i to that of the version j, we have successfully measured the editing effort of students. On the other hand, a semantic metric, e.g., based on LSA, reflects real content changes. For example, semantic metrics can be used to indicate whether a news article is worth reading again. We use LSA because, in the literature, it has been shown that LSA is superior to other semantic similarity measures [12]. See Sect. 4.2 for further details about our implementation.

Based on the metric D, we find two possible definitions of the contribution of a given version to the current version of a document. The basic idea is illustrated in Fig. 2, where nodes represent the versions of a Web document

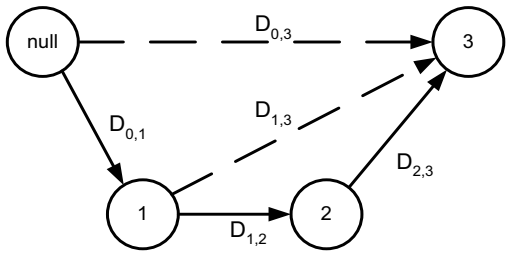

Fig. 2. Distances between different versions of a document

and edges represent the distance. The empty document, from which the first version origins, is denoted as null. The contribution of version 2, for example, may be defined as $D_{1,3} - D_{2,3}$ or "how much the distance to the end version 3 has decreased from version 1 to version 2." Alternatively, the contribution can be defined as $D_{1,2}$ or "how much the document has been changed from version 1 to version 2". Notice that the contributions of some versions may be negative in the first case.

In the first case we have $\bar{c}_i := D_{i-1,n} - D_{i,n}$, where $D_{0,i}$ is the distance between the empty document to version i. Then the *effective freshness* of a document with n versions is given by (since $D_{n,n} = 0$):

$$\bar{t}_n = \frac{\sum_{i=1}^{n} t_i(D_{i-1,n} - D_{i,n})}{\sum_{i=1}^{n}(D_{i-1,n} - D_{i,n})} = \frac{t_1 \cdot \sum_{i=1}^{n}(D_{i-1,n} - D_{i,n}) + \sum_{i=1}^{n}(t_i - t_1)\cdot(D_{i-1,n} - D_{i,n})}{\sum_{i=1}^{n} D_{i-1,n} - \sum_{i=1}^{n} D_{i,n}}$$

$$= \frac{t_1 D_{0,n} + \sum_{i=2}^{n} t_i D_{i-1,n} - \sum_{i=1}^{n-1} t_i \cdot D_{i,n} - t_n D_{n,n}}{D_{0,n} - D_{n,n}} = t_1 + \sum_{i=2}^{n}(t_i - t_1) \cdot \frac{D_{i-1,n} - D_{i,n}}{D_{0,n}}$$

If a document only has a single version ($n = 1$) the effective freshness is t_1, as expected. Each additional version may increase the effective freshness, depending on the time difference between the new version and the first version, and depending on how much the content has been changed comparing with *all* past versions. Using this algorithm when a new version of a Web document is added to the WCMS, a comparison to each past version must be carried out.

In the second case we have $\tilde{c}_i := D_{i-1,i}$. Then the *incremental freshness* of a document with n versions is given by:

$$\tilde{t}_n = \frac{\sum_{i=1}^{n} t_i(D_{i-1,i})}{\sum_{i=1}^{n} D_{i-1,i}} = \frac{t_n D_{n-1,n} + \sum_{i=1}^{n-1} t_i D_{i-1,i}}{D_{n-1,n} + \sum_{i=1}^{n-1} D_{i-1,i}} = \frac{t_n D_{n-1,n} + \tilde{t}_{n-1} \sum_{i=1}^{n-1} D_{i-1,i}}{D_{n-1,n} + \sum_{i=1}^{n-1} D_{i-1,i}}$$

Notice that $\tilde{t}_n$ can be calculated *incrementally* using $\tilde{t}_{n-1}$ and the accumulated $\sum_{i=1}^{n-1} D_{i-1,i}$. If a document only has a single version ($n = 1$) the incremental freshness yields t_1, as expected. Each additional version increases the incremental freshness, depending on the time difference between the new version and the

first version, and depending on how much the content has been changed compared to the *previous* version. A comparison only to the previous version is necessary, which reduces computational complexity substantially.

Unfortunately as pointed out by [3]: "...resemblance is not transitive, ...for instance consecutive versions of a paper might well be 'roughly the same', but version 100 is probably quite different from version 1." Or in the reverse case: If extensive changes made to a version have been undone completely in the following version, no real increase of freshness is achieved whilst a comparison between the consecutive versions might pretend a significant increase. Therefore, we expect that $\bar{t}_n$ resembles our idea of freshness better than $\tilde{t}_n$ at the cost of additional computation.

In practice, an approximate algorithm may be used to reduce computational complexity for $\bar{t}_n$ because the contributions of some versions to $\bar{t}_n$ can be neglected. Notice that $\bar{t}_n$ can also be expressed for any j $(1 < j < n)$:

$$\begin{aligned}\bar{t}_n \;=\; & t_1 + \sum_{i=2}^{j-1} (t_i - t_1) \cdot \tfrac{D_{i-1,n} - D_{i,n}}{D_{0,n}} + \\ & + (t_j - t_1) \cdot \tfrac{D_{j-1,n} - D_{j,n}}{D_{0,n}} + (t_{j+1} - t_1) \cdot \tfrac{D_{j,n} - D_{j+1,n}}{D_{0,n}} + \\ & + \sum_{i=j+2}^{n} (t_i - t_1) \cdot \tfrac{D_{i-1,n} - D_{i,n}}{D_{0,n}}\end{aligned}$$

If the contribution of version j is neglected the inaccuracy of $\bar{t}_n$ can be estimated as

$$\begin{aligned}|\triangle \bar{t}_n| \;&=\; \left| (t_j - t_1) \cdot \tfrac{D_{j-1,n} - D_{j,n}}{D_{0,n}} + (t_{j+1} - t_1) \cdot \tfrac{D_{j,n} - D_{j+1,n}}{D_{0,n}} - (t_{j+1} - t_1) \cdot \tfrac{D_{j-1,n} - D_{j+1,n}}{D_{0,n}} \right| \\ &=\; (t_{j+1} - t_j) \cdot \tfrac{|D_{j-1,n} - D_{j,n}|}{D_{0,n}} \\ &\leq\; (t_{j+1} - t_j) \cdot \tfrac{D_{j-1,j}}{D_{0,n}}\end{aligned}$$

Notice that $|D_{j-1,n} - D_{j,n}| \leq D_{j-1,j}$ because $D_{j-1,n}$, $D_{j,n}$, and $D_{j-1,j}$ form a triangle in the version graph (see Fig. 2). In other words: if an accuracy of $\triangle \bar{t}_n$ is sought the contribution of each version j $(1 < j < n)$ with

$$(t_{j+1} - t_j) D_{j-1,j} \;<\; \tfrac{\triangle \bar{t}_n \cdot D_{0,n}}{n}$$

can be neglected, since it is quite similar to its previous version and was replaced by the next version after a short time period. Moreover, such a version may be vacuumed to save space.

3.2 Freshness of a News Page

When we apply our approach to our example news page it turns out that only the time consuming and complex calculation of the effective freshness $\bar{t}_n$ yields useful results.

$D^{\mathsf{XML}}_{i,j}$	1	2	3	4	5	6	7	8
0	919	910	903	904	962	963	1273	1223
1		348	337	336	624	623	1778	1667
2			5	9	287	286	1655	1480
3				4	312	313	1707	1542
4					297	298	1682	1494
5						4	1602	1601
6							1601	1518
7								73

$D^{\mathsf{GTP}}_{i,j}$	1	2	3	4	5	6	7	8
0	1	1	1	1	1	1	1	1
1		0.66	0.66	0.66	0.82	0.82	0.92	0.93
2			0.15	0.15	0.61	0.61	0.88	0.88
3				0.15	0.61	0.61	0.88	0.88
4					0.61	0.61	0.88	0.88
5						0.11	0.89	0.91
6							0.89	0.91
7								0.37

Table 2
Syntactic (D^{XML}) and semantic (D^{GTP}) distances between specific versions of our example news page

For each of the eight versions of the breaking news about the Ukrainian President's Vote, which we found at YahooNews.com, we recorded the last-modified time stamp t_n, and calculated the effective freshness $\bar{t}_n$ and the incremental freshness $\tilde{t}_n$. Both kinds of freshness were calculated w.r.t. our syntactic document metric D^{XML} and our semantic document metric D^{LSA}. D^{XML} counts the number of XML node modifications — the greater the number, the greater the change. D^{LSA} measures semantic similarity, where 0 means semantic equivalence and 1 means that the versions are semantically unrelated. See Sect. 4.2 for a detailed description of our document metrics.

Tab. 2 shows the distances between document versions. Both metrics reflect that the versions 2, 3, and 4 are quite equivalent to each other; the same holds for the versions 5 and 6. We have dramatic changes both in syntax and semantics between versions 4 and 5, and between versions 6 and 7. The semantic metric D^{LSA} shows, however, that the semantic changes are not as drastic as the syntactic changes. Clearly, D^{LSA} is not affected by presentation style modifications and, therefore, can be used to consider questions like: "Does an article reveal new information?"

As shown in Fig. 3, the graphs representing $\bar{t}_n$ and $\tilde{t}_n$, respectively, are below the graph representing t_n. The graph of $\bar{t}_n$ makes a jump towards t_n at the seventh version, which is caused by many changes made to that version. This jump is also visible in the graph of $\tilde{t}_n$, but it does not reveal the significance of the content change. Notice that the graphs computed with our semantic metric D^{LSA} show a smoother transition between version 6 and 7. However, the semantic change between version 4 and 5 is more significant than the syntactic change. Neglecting version 4, which is quite equivalent to version 3, has almost no effect.

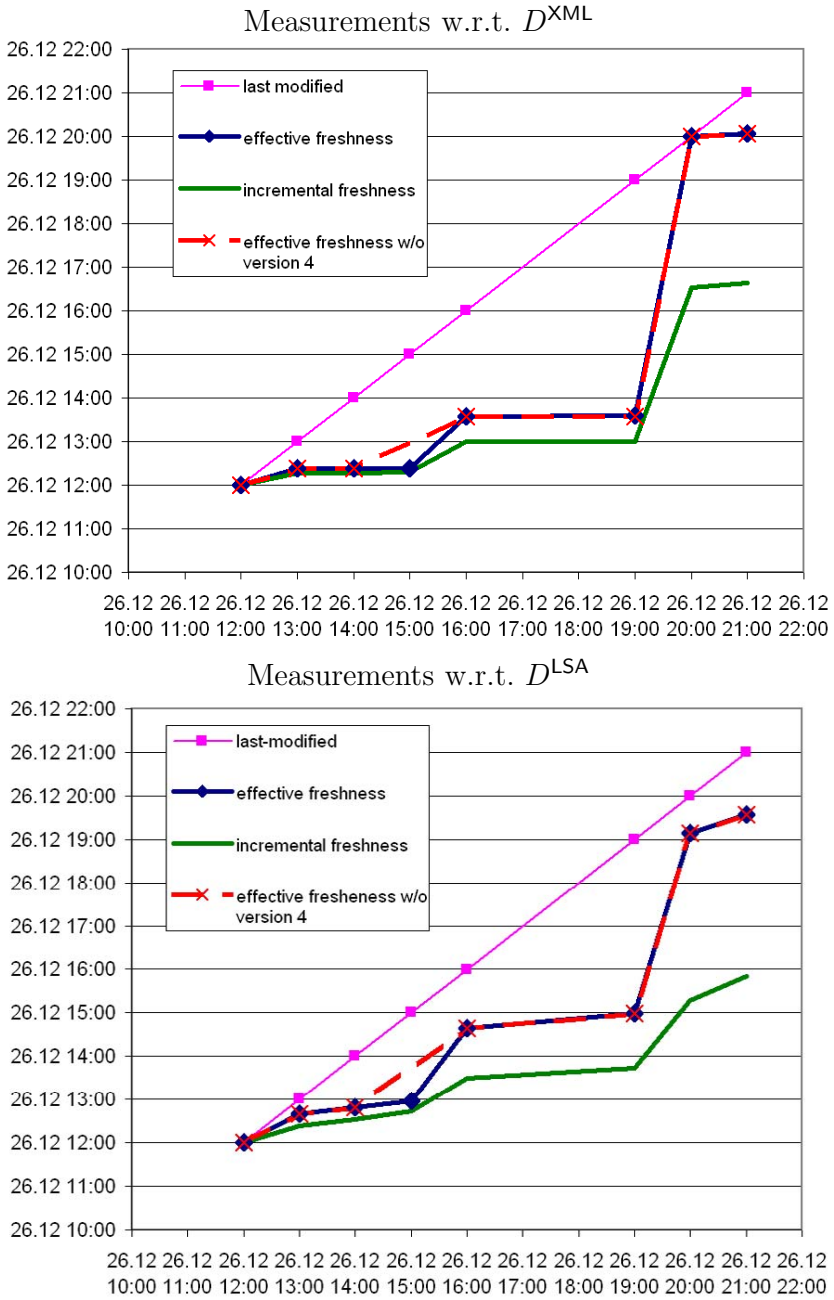

Fig. 3. Recorded and calculated time stamps for our example news page w.r.t. our syntactic metric D^{XML} and our semantic metric D^{LSA}, respectively

At the news site derstandard.at[6] the difference between $\bar{t}_n$ and $\tilde{t}_n$ is even

[6] URL derstandard.at/druck/?id=1901315

more significant since the document has been changed completely many times.

As a matter of fact, if a document has been changed completely (w.r.t. the document metric used) the effective freshness should be set to the last-modified time stamp, as the calculation of $\bar{t}_n$ delivers.

4 An Up-to-dateness Aware WCMS

We have implemented our approach into our XML-centric WCMS, which supports versioning of XML contents [10]. Its open architecture supports easy integration of different document metrics. Currently, our syntactic metric D^{XML} employs Microsoft XMLDiff [15], whereas our semantic metric D^{LSA} uses the General Text Parser GTP [7].

4.1 XML-centric WCMS

Our WCMS is based on Windows Active Server Pages technology. Its architecture is XML centric regarding information processing and data storage. The WCMS uses the Open Source HTML editor RichTextEdit [6], which provides a WYSIWYG user interface and converts HTML information into valid XHTML. The WCMS makes available all the necessary XML data for a Web page and supplies XSLT templates that should be used for generating HTML information from these data.

When a client requests an HTML page, the WCMS responds with XML information from the database. An XML-capable browser translates this information into presentable HTML using the associated XSLT template, which it fetches from the server in a second request. For non-XML browsers the XML information can be translated to HTML on the server side as well.

Because of its XML-centric and simple data model on the database layer, the WCMS is flexible regarding extensions. In most cases, extensions can be realized by making adaptations in XSLT. Besides the current version of each Web document all past versions of the same document, their authors, and "last-modified" time stamps can be retrieved, too.

Based on this conception, it is easy to implement a server-side extension for comparing different versions of a web document. This extension has to be invoked each time when a document has been updated. Alternatively, the functions for comparing different versions of web documents could be implemented on the client-side, using the data cached by a web browser. This means, however, that such a comparison must be carried out much more often; actually, each time when a web page is retrieved by a client. Furthermore, many old versions, if not all, have to be stored by each client that is interested

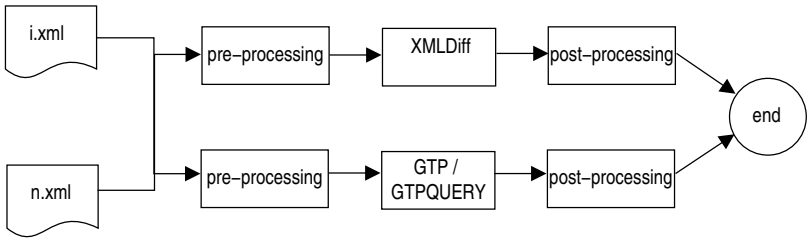

Fig. 4. Integration of application-specific document metrics

in the freshness of a web document. This is much less feasible than the server-side solution. An additional advantage of the server-side implementation is that it can be used for a more effective server-side caching. An intelligent browser may send a request including the information about the cached version and a time tolerance. An intelligent WCMS, in turn, sends a response "304 not modified" even if the Web document under request has been slightly changed but the difference regarding effective freshness between the cached version and the recent version is below the tolerance.

4.2 Using Application-Specific Document Metrics

Fig. 4 gives an overview of the subsystem for comparing different versions of a Web document, i.e., determining the distance $D_{i,n}$ between versions i and n. The subsystem is implemented as a server side system because we intend to integrate the metric into our WCMS. It retrieves all versions of a given Web document from the WCMS and pre-processes them via XSLT. This transformation accomplishes several goals.

Firstly, the WCMS holds internal metadata that should not be compared. For example, there are the URL of the separate image server or the login name of the current user. Since our WCMS is an experimental system there are even debugging and profiling entries included. The pre-processing simply removes these (in this case) superfluous metadata.

Secondly, each particular metric requires the document versions in a specific format. For our syntactic metric D^{XML}, each word in a paragraph is transformed to a node, such that text modifications (including layout) can be detected more accurately. For example, the metric can associate different editing efforts to different kinds of nodes. For processing Chinese texts, it makes sense to divide paragraphs into single "characters" (see Fig. 5) because Chinese word boundaries are not marked using white spaces and cannot be detected using traditional document metrics. For our semantic metric D^{LSA}, however, the pre-processing removes all markup such that the metric deals with the plain text information only.

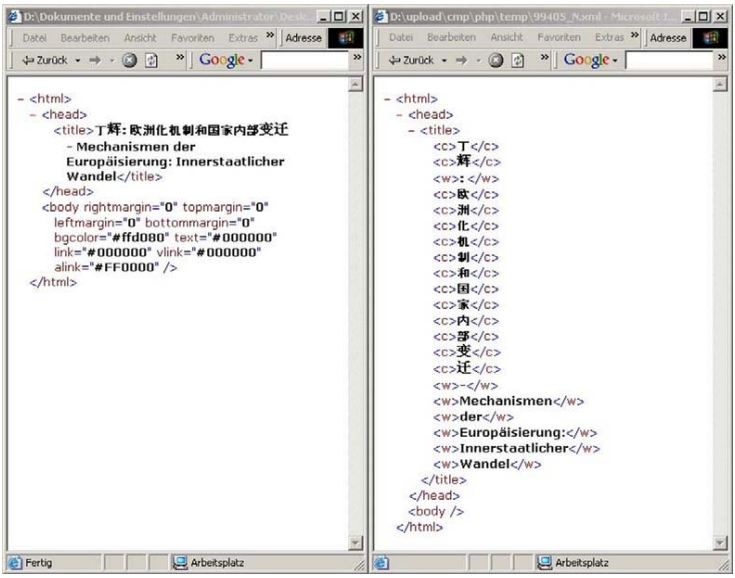

Fig. 5. Pre-processing of Chinese texts

Finally, the HTML editor we use may generate additional entries pretending more additions or changes. For example, when the width of a table column has been adjusted, changes might be detected in each row of the table. The pre-processing suppresses these additional changes.

For measuring editing effort, D^{XML} compares the pre-processed documents using Microsoft's XMLDiff [15], which represents the changes using XDL, a proprietary XML-based language for describing differences between two XML documents. XDL Diffgrams contain information regarding additions, changes, or removals of document content, or content being moved within the XML tree. During the XSLT post-processing, the entries in an XDL Diffgram are compiled and the number of modifications is calculated.

For measuring semantic similarity by D^{LSA}, all document versions are fed into GTP, which builds a document base including the single value matrix decompositions as usual in LSA. Then the new version n is used as a query against this document base. GTPQUERY computes the cosine similarity measure for version n w.r.t. all previous versions that must be taken into account. Since GTPQUERY returns 1 for semantically equal documents and 0 for documents that have no semantic relation, we compute the semantic distance simply by $1 - \text{GTPQUERY}$.

Notice that the metrics do not need to be normalized (except that large values denote a large distance) because freshness computation is based on the relative change of the distance.

4.3 Measuring Editing Effort by a Syntactic Document Metric

Our example has shown that for evaluating news pages, at least, a semantic document metric should be employed. We have found, however, that for measuring editing effort, using a syntactic metric makes sense. Also, a syntactic metric may indicate readability improvements. In this section, we describe an experiment that aims to evaluate the editing effort of student's works. Indeed, this was our original goal. Our WCMS represents a realistic testing environment as it is used by students to write their final theses and term papers and, therefore, contains many different documents. The examined students works were all from the area of business administration. Henceforward, they were characterized by a relatively similar structure especially concerning the relation of text to non-text parts, e.g., tables, links, and pictures (which is important for using syntactic document metrics).

To be able to draw general conclusions based on the empirical results of the examined students works, the main focus lay on several test cases that have been generated. The purpose of these tests was to determine whether the efforts in writing corresponded to the effective change estimated by our syntactic metric D^{XML}. The test cases have been generated based on the variety of different functions available in the built-in text editor RichTextEdit. To cover the whole process of document writing with all its components, the work on objects (e.g., tables, links, or pictures) and the document style was taken into account. XMLDiff distinguishes four kinds of changes (additions, changes, removals, and moving of nodes) within its comparison between the two XML files; so these different types of changes were tested.

Each test case represented a different action, which was derived from the combination of a function with one of the four possible changes (e.g., "add table" or "delete picture"). The average time to perform each action (e.g., delete a picture) was measured using a stopwatch in several independent runs to receive the temporary effort. In the next step, D^{XML} was applied to a document where this specific action had been realized and the resulting value (XML node) was related to the average temporary effort measured.

D^{XML} calculates with nodes, each word representing an individual node. If new words have been added to a document, D^{XML} presents these changes throughout its result, which is a figure of nodes. For text writing (add text), each node measured by D^{XML} corresponds to 1.92 seconds of real editing effort. Since removing a single word or an associated group of words represents little real time effort only (by average 2 seconds per removal), this action is treated by the system as a single node, which leads to the average figure of 2 seconds per node for text removal. The real editing effort of all different actions, using objects like tables, pictures etc. was analyzed, too, and expressed by the

average value of real time effort per node. These values ranged from 2 to 6 seconds per node. In the students works, objects like tables or pictures were relatively rare in relation to the text parts. Therefore, the small deviation of these values from the values of text creation (1.92 seconds) and text removal (2 seconds) has almost no impact on the overall real editing effort of a whole document.

By using this easily adjustable XML-centric system including the pre- and post-processing, the influence of each action and object could be treated and, therefore, lead to a discretional adjustment of the real editing effort value. In dependence on the application area of the text (e.g., business administration or computer science) or the individual author, by using the real editing effort statistics, the real time effort could be personalized or altered w.r.t. the type of text.

5 Determining Up-to-dateness of Multiple Documents

Web pages do not exist on their own. Instead, they are semantically related to each other. Due to these semantic relations, changes in the up-to-dateness of one Web page "somehow" influence the up-to-dateness of related Web pages.

Consider our running example again: Our breaking news page is hosted at YahooNews.com, which also contains interviews and background information. For the purposes of this paper, suppose that there are an interview and an historical article about the Ukraine at YahooNews.com revealing further background information. In addition, the breaking news page and the interview are part of a "Today's News" summary, which itself is part of a news chronical. Of course, the historical article is part of the chronical, too.

5.1 Semantic Relations

Clearly, we would expect that up-to-dateness of the chronical is influenced by the up-to-dateness of the breaking news page. This is because the chronical is (indirectly) semantically related to the breaking news page.

We use the hypertext model as basis for representing semantic relations, similarly to [14,8]. Semantic relations are represented via *semantic links*, which are treated as first-class entities connecting source documents with target documents. A semantic link can connect multiple source documents to multiple target documents.[7] Usually, such links are stored in a link base [14]. The

[7] Notice that this is different from embedded links in HTML pages. Treating links as first-class entities gives us much more flexibility, e.g., it supports multiple link structures on the same documents without altering the documents themselves.

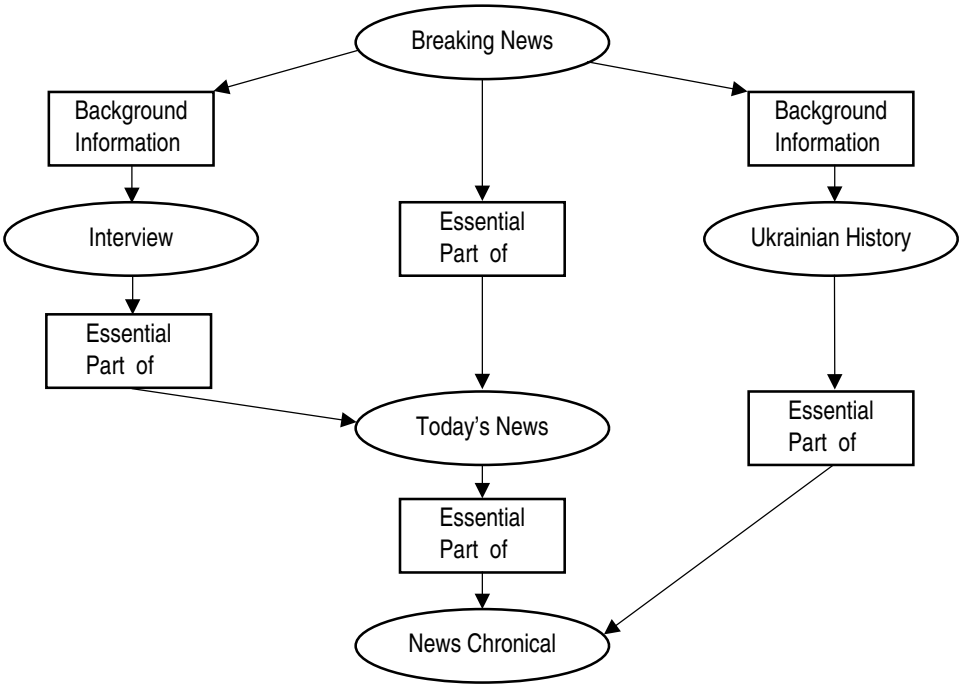

Fig. 6. Semantic structure of our example Web site

semantic structure of a Web site (see Fig. 6) can be represented through a bipartite directed acyclic graph (DAG).[8] Ellipses represent documents; rectangles represent semantic links. Source and target documents, respectively, of a semantic link are denoted by directed edges. Our idea is to propagate changes in the up-to-dateness of the source documents to the target documents. The example contains bidirectional links only; our approach is, however, independent of link arity.

Currently, semantic links are added by hand. Often, they are in reverse direction to the usual link structures in Web pages; e.g., the "Today's News" page links to the breaking news page and the interview. In the future, we plan to derive semantic links, e.g., based on embedded HTML links or semantic similarity via latent semantic linking techniques [14]. Using latent semantic linking requires, however, to assign further properties to links, which we explain below.

Semantic links are typed. A semantic link of type $\tau_{n,m}$ has n source documents and m target documents.[9] This type is associated to an up-to-dateness

[8] For technical reasons, we prohibit circles in the semantic graph.

[9] Due to the simplicity of our example document metrics, we do not need to further type source and target documents. Of course, this may change if we employ other document metrics.

propagation function $[\![\tau_{n,m}]\!] : \mathbb{R}^n \to \mathbb{R}^m$, which given up-to-dateness changes in the source documents calculates up-to-dateness changes of the target documents. On an update of a source document d, we can propagate its up-to-dateness changes along the edges of the semantic DAG. Notice that we can reach a target document d' at different paths origining from d, each of which may require to change the up-to-dateness of d' differently.

5.2 Propagating Up-to-dateness Changes

Consider that a document d has been updated to such an extent that its freshness changes. Then we immediately update d's up-to-dateness according to the change of its freshness, which naturally determines d's up-to-dateness change $\Delta(d)$. Up-to-dateness changes of other documents are set to zero. Our propagation algorithm traverses the semantic DAG, where it regards d as root. Each edge that has been traversed is marked as "processed." Each document node is marked as "processed" if all incoming edges have been marked as "processed." We traverse the semantic DAG as follows:

- Processing a document node d:
  If all incoming edges are marked as "processed," then update d's up-to-dateness according to its up-to-dateness change $\Delta(d)$, and process all semantic links emanating from d and mark their edges as "processed." Otherwise, process any semantic link targeting d whose edge has not been processed already. Processing a semantic link will update d's up-to-dateness change $\Delta(d)$ and return to d to further process incoming links or emanating links.
- Processing a semantic link node l of type $\tau_{n,m}$:
  First, process all source documents of l that have not been processed already and mark their edges as "processed." Processing these documents will determine their up-to-dateness changes. Second, apply l's up-to-dateness propagation function $[\![\tau_{n,m}]\!]$ to the up-to-dateness changes $\Delta(d_i)$ of the source documents d_i ($i \in \{1, \ldots, n\}$). This results in m up-to-dateness changes $\Delta_l(d'_j)$ for the target documents d'_j ($j \in \{1, \ldots, m\}$). Update the (already calculated) up-to-dateness changes $\Delta(d'_j)$ of the target documents d'_j to the maximum of $\Delta_l(d'_j)$ and $\Delta(d'_j)$. Third, process all target documents d'_j of l and mark their edges as "processed."

Our algorithm is non-deterministic because we do not make any assumptions about the order in which nodes are processed. Since up-to-dateness changes are updated to the maximum, however, our algorithm always yields the same result, no matter in which order nodes are processed. [10] Notice that

[10] Instead of the maximum, any commutative and associative function can be used here.

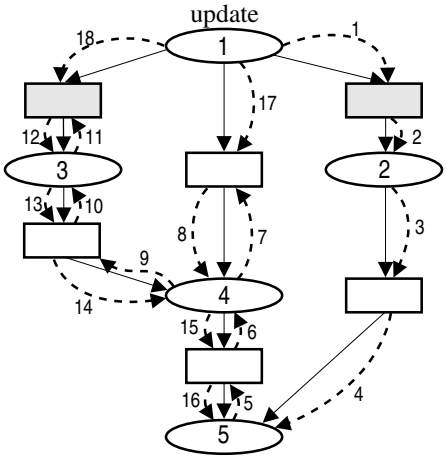

Fig. 7. Up-to-dateness propagation of our example breaking news page

up-to-dateness resembles freshness of "lonely" documents, which are not semantically related to other documents, and as long as no semantically related document has been changed. Otherwise, the up-to-dateness of a document may differ from its freshness.

5.3 Propagating Up-to-dateness of our Example Web Site

For example, consider an update of our example breaking news page. Fig. 7 shows a possible traversal of our algorithm through our example Web site. Ellipses, rectangles, and edges represent the semantic DAG as shown in Fig. 6. "Background Information" links are marked grey, in order to distinguish them from "Essential Part of" links. Dotted arrows show how our algorithm traverses the DAG; they are numbered according to the order in which nodes are processed. Document nodes are numbered according to the order in which they are *marked* as "processed;" i.e., the order in which their new up-to-dateness is determined.

From version 6 to 7 the effective freshness of the breaking news page jumps up by four hours using our semantic document metric D^{LSA} (see Fig. 3). Given the up-to-dateness propagation functions

$$[\![\text{Background Information}]\!](src) = src \cdot 0.25 \qquad [\![\text{Essential Part of}]\!](src) = src \cdot 0.5$$

our algorithm yields the following up-to-dateness changes:

$$\begin{aligned} \Delta(\text{Breaking News}) &= 4.0 \text{ hrs} & \Delta(\text{Interview}) &= 1.0 \text{ hrs} \\ \Delta(\text{Today's News}) &= 2.0 \text{ hrs} & \Delta(\text{Ukrainian History}) &= 1.0 \text{ hrs} \\ \Delta(\text{News Chronical}) &= 1.0 \text{ hrs} \end{aligned}$$

That is, the up-to-dateness of the chronical increases by one hour.

6 Conclusions and Outlook

This paper describes our approach towards automatic management of up-to-dateness of documents that are managed by an XML-centric WCMS. We introduce two measures: the freshness of a document is based on its own history only; the up-to-dateness of a document also employs semantic relations to other documents. *Freshness* of a multiversioned Web document is calculated w.r.t. a document metric, which detects changes between document versions. Currently, we use two general-purpose metrics: Our syntactic metric is based on the modifications of XML-nodes, which roughly reflects editing effort. Our semantic metric employs LSA techniques to determine semantic content modifications. Due to our open architecture, new application-specific metrics can be easily integrated. Since pre- and post-processing using XSLT are included, document metrics can be easily adapted to different human-machine interfaces or user groups. *Up-to-dateness* is based on semantic relations between documents. Changes in up-to-dateness are propagated along these relations.

In the future, we plan to extend our WCMS by recording the document process time automatically. Not only in this way more data on the real effort for document processing by different users should be collected for further validations of the document metrics. We believe that such metrics are key success factors for managing document processes as one of the crucial parts of business process management. We also want to learn more about document authoring patterns for which content management systems and collaborative authoring systems can be optimized. Monitoring the deviation between freshness and up-to-dateness of a document might also contribute to authoring patterns. Currently, semantic links are maintained by hand. We plan to evaluate automatic semantic linking techniques like [14] in order to alleviate semantic link management. In addition, we will implement a plug-in mechanism into our WCMS that supports easy integration of user-defined document metrics. For easier maintenance, we currently stick to a central WCMS. It would be interesting to see how our approach scales to distributed Web sites that use persistent URLs to address Web documents.

In addition, we plan to integrate our up-to-dateness approach with automated consistency management as offered by CDET [20] or xlinkit [16]. Clearly, up-to-dateness changes give rise to consistency constraints based on semantic links. For example, consider a German translation of our breaking news page, which should be as up to date as the English original.

In summary, we think that our approach provides a good basis to manage

real up-to-dateness of Web documents beyond the simple "last-modifying" time stamp, which has proven insufficient for many purposes.

References

[1] T. Berners-Lee. Information management: A proposal, 1989. www.w3.org/History/1989/proposal.html.

[2] B. E. Brewington and G. Cybenko. How dynamic is the Web? In *Proc. of the 9th Int. World Wide Web Conf.*, Amsterdam, The Netherlands, 2000. W3C.

[3] A. Broder, S. Glassman, M. Manasse, and G. Zweig. Syntactic clustering of the Web. In *Proc. of the 6th Int. World Wide Web Conf.*, pages 391–404, Santa Clara, CA, 1997. W3C.

[4] C. Dyreson, H.-L. Lin, and Y. Wang. Managing versions of Web documents in a transaction-time Web server. In *Proc. of the 13th Int. World Wide Web Conf.*, pages 422–432, New York, NY, 2004. W3C.

[5] D. Fetterly, M. Manasse, M. Najork, and J. Wiener. A large-scale study of the evolution of Web pages. In *Proc. of the 12th Int. World Wide Web Conf.*, pages 669–678, Budapest, Hungary, 2003. W3C.

[6] A. France et al. RichTextEdit, 2003. www.richtext.org.uk/.

[7] J. T. Giles, L. Wo, and M. W. Berry. GTP (General Text Parser) software for text mining. In H. Bozdogan, editor, *Statistical Data Mining and Knowledge Discovery*, pages 455–471. CRC Press, 2003.

[8] S. C. Gupta, T. N. Nguyen, and E. V. Munson. The software concordance: Using a uniform document model to integrate program analysis and hypermedia. In *Proc. of 10th Asia-Pacific Software Engineering Conf.*, pages 164 – 173, Chiang Mai, Thailand, 2003. IEEE CS Press.

[9] J. Hand. *Feasibility of using citations as document summaries.* PhD thesis, Drexel University, 2003.

[10] B. Hu and F. Lauck. Prototype of a Web and XML based collaborative authoring system. In *Proc. of the Int. Conf. on Computing, Communications and Control Technologies*, volume IV, pages 79–84, Austin, TX, 2004. IIIS.

[11] A. Kirby, P. Rayson, T. Rodden, I. Sommerville, and A. Dix. Versioning the Web. In *7th Int. Wkshp. on Software Configuration Management*, pages 163–173, Boston, MA, 1997.

[12] M. D. Lee, B. M. Pincombe, and M. B. Welsh. An empirical evaluation of models of text document similarity. In *Proc. of the 27th Annual Conf. of the Cognitive Science Society*, to appear 2005.

[13] N. Littlestone. Learning quickly when irrelevant attributes are abound: A new linear threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[14] A. Macedo, M. Pimentel, and J. Guerrero. Latent semantic linking over homogeneous repositories. In *Proc. of the 2001 ACM Symp. on Document engineering*, pages 144–151, Atlanta, GA, 2001. ACM Press.

[15] Microsoft Corp. XMLDiff 1.0. apps.gotdotnet.com/xmltools/xmldiff/, 2002.

[16] C. Nentwich, L. Capra, W. Emmerich, and A. Finkelstein. xlinkit: a consistency checking and smart link generation service. *ACM Trans. Inter. Tech.*, 2(2):151–185, 2002.

[17] K. Nigam and M. Hurst. Towards a robust metric of opinion. In *AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*, Palo Alto, CA, 2004. Stanford University.

[18] A. Ntoulas, J. Cho, and C. Olston. What's new on the Web? the evolution of the Web from a search engine perspective. In *Proc. of the 13th Int. World-Wide Web Conf.*, pages 1–12, New York, NY, 2004. W3C.

[19] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.

[20] J. Scheffczyk, U. M. Borghoff, P. Rödig, and L. Schmitz. Managing inconsistent repositories via prioritized repairs. In *Proc. of the 2004 ACM Symp. on Document Engineering*, pages 137–146, Milwaukee, WI, 2004. ACM Press. see also www2-data.informatik.unibw-muenchen.de/cde.html.