European Commission Seventh Framework Programme
Call: FP7-ICT-2007-1, Activity: ICT-1-4.1
Contract No: 216267

Prototype and Applications Evaluation Report V1
D6.4
Version 1.0

Editor: BeG
Work Package: WP6
Status: Final Version
Date: M15
Dissemination Level: RE
Project Overview

Project Name: LiWA – Living Web Archives
Call Identifier: FP7-ICT-2007-1
Activity Code: ICT-1-4.1
Contract No: 216267

Partners:
1. Coordinator: Universität Hannover, Learning Lab Lower Saxony (L3S), Germany
2. European Archive Foundation (EA), Netherlands
3. Max-Planck-Institut für Informatik (MPG), Germany
4. Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SZTAKI), Hungary
5. Stichting Nederlands Instituut voor Beeld en Geluid (BeG), Netherlands
6. Hanzo Archives Limited (HANZO), United Kingdom
7. National Library of the Czech Republic (NLP), CZ
8. Moravian Library (MZK), CZ

Document Control

Title: D6.4 Prototype and Applications Evaluation Report V1
Author/Editor: Roeland Ordelman (BeG)
Radu Pop (EA)
Mark Williamson (HANZO)
Andras Benczur (MTA)
Marc Spaniol (MPG)
Wolf Siberski (L3S)
Libor Coufal (NLP)

Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author/Editor</th>
<th>Description/Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>13/03/2009</td>
<td>Roeland Ordelman</td>
<td>first draft</td>
</tr>
<tr>
<td>0.2</td>
<td>01/04/2009</td>
<td>all</td>
<td>modules and applications description</td>
</tr>
<tr>
<td>0.3</td>
<td>16/04/2009</td>
<td>Roeland Ordelman</td>
<td>input incorporated</td>
</tr>
<tr>
<td>1.0</td>
<td>20/04/2009</td>
<td>Roeland Ordelman</td>
<td>final version</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Radu Pop</td>
<td></td>
</tr>
</tbody>
</table>
Legal Notices
The information in this document is subject to change without notice.
The LiWA partners make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The LiWA Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.
# Table of Contents

INTRODUCTION ........................................................................................................... 5

PROTOTYPE EVALUATION ............................................................................................... 6

1 LiWA Modules ............................................................................................................. 7
   1.1 Rich Media Capturing ............................................................................................. 7
   1.2 Link Extractor ......................................................................................................... 10
   1.3 Spam filter .............................................................................................................. 11
   1.4 Temporal Coherence ............................................................................................. 14
   1.5 Terminology Evolution .......................................................................................... 16

2 Integration testing on the testbed ................................................................................ 19
   2.1 Integration of the Temporal Coherence module ..................................................... 20
   2.2 Integration of the Rich Media Capturing module .................................................... 22

LIWA APPLICATIONS ................................................................................................... 24

3 “Streaming” Application ............................................................................................ 24
   3.1 Feedback on mock-up ............................................................................................ 24
   3.2 Application evaluation plan ................................................................................... 27

4 “Social Web” Application .......................................................................................... 30
   4.1 Feedback on mock-up ............................................................................................ 30
   4.2 Application evaluation plan ................................................................................... 33

CONCLUSION .................................................................................................................. 35
Introduction

The purpose of this document is to capture the experiences gathered so far in order to adapt the further development in LiWA. The collected information stems from two different sources. First, the integration testing testbed (see D 6.2) provided valuable experience reports regarding the evaluation of individual LiWA modules as well as their integration. Second, testbed partners have been asked to give feedback to the application mock-ups for the “Streaming” application (see D7.1) as well as the “Social Web” application (see D8.1).

Section 2 describes the evaluation of LiWA modules and their integration in the testbed. For each individual module, the conducted tests are described, including evaluation methods, performance indicators, and the test collections on which the tests are performed (see Section 2.1). In Section 2.2, the integration test parameters and settings and the outcomes of the integration tests are reported.

The feedback on application mock-ups is described in Section 3. For both applications, “Streaming” and “Social Web”, a walkthrough is described to frame the scope of the evaluation. Feedback has been captured by collecting questionnaire responses. The results of this feedback process are summarized for both applications. Finally, plans for the further evaluation of the “Streaming” and “Social Web” applications are provided.
Prototype Evaluation

As we described in the system's architecture for LiWA (D1.1), we distinguish between two different types of modules, with regards to their place in the Web archiving workflow:

- *Internal modules* – directly integrated into the crawl engine, to improve the crawl management and quality: the Rich Media Capturing module and the Temporal Coherence analyser;
- *External modules* – loosely coupled with the crawler, used in the post-processing phase: the Spam Filter, the Off-line Temporal Coherence analyser and the Semantic analyser.

The goal of the integration testing activity is to create a software framework on LiWA’s testbed. This framework is necessary for the complete workflow in Web archiving and for the seamless integration of the new modules.

Each module developed in LiWA has been locally tested for the first version of the technologies and their functional description is briefly presented in the next sections. Furthermore, the preliminary steps in assembling all the software pieces was achieved by building the four LiWA modules on the same environment, using the Cruise Control system deployed on the testbed (as described in D6.3).

In Section 1.1 we present a summary description of the LiWA modules, including an overview of the performance indicators and their test collections. Section 1.2 presents the experience gathered so far on integrating the internal modules into the testbed system.
1 LiWA Modules

1.1 Rich Media Capturing

The Rich Media Capturing module is designed to enhance the capturing capabilities of the crawler, with regards to different multimedia content types. The main performance indicator for this module is therefore related to the number of additionally archived multimedia types.

The current version of Heritrix is mainly based on the HTTP/HTTPS protocol and it cannot treat other content transfer protocols widely used for the multimedia content (such as streaming). The Rich Media Capturing module delegates the multimedia content retrieval to an external application (such as MPlayer) that is able to handle a larger spectrum of transfer protocols.

The Real Time Streaming Protocol (RTSP) is a network control protocol for use in entertainment and communications systems to control streaming media servers. The protocol is used to establish and control media sessions between end points. Clients of media servers issue VCR-like commands, such as play and pause, to facilitate real-time control of playback of media files from the server.

The RTSP protocol has similarities to HTTP, but RTSP adds new requests. While HTTP is stateless, RTSP is a stateful protocol. A session identifier is used to keep track of sessions when needed. Thus, no permanent TCP connection is needed. RTSP messages are sent from client to server, although some exceptions exist where the server will send to the client. The default transport layer port number is 554.

Multimedia Messaging Service (MMS) is a telecommunications standard for sending messages with multimedia objects (images, audio, video, rich text). MMS is an extension of the SMS standard, allowing longer message lengths and using WAP to display the content. MMS messages are delivered in a fashion almost identical to SMS, but any multimedia content is first encoded and inserted into a text message in a fashion similar to sending a MIME e-mail.

1.1.1 Performance indicators, metrics and methods

The module is constructed as an external plugin for Heritrix. Using this approach, the identification and retrieval of streams is completely decoupled, allowing to use more efficient tools to analyze video and audio content. At the same time, using the external tools helps in reducing the burden on the crawling process.

The module is composed of several subcomponents that communicate through messages. We use an open standard communication protocol called Advanced Message Queuing Protocol (AMQP). The workflow of the messages can be summarised as follows.
The plugin connected to Heritrix detects the URLs referencing streaming resources and it constructs for each one of them an AMQP message. This message is passed to a central Messaging Server. The role of the Messaging Server is to decouple the Heritrix crawler from the clustered streaming crawlers (i.e. the external capturing tools). The Messaging Server stores the URLs in queues and when one of the streaming crawlers is available, it sends the next URL for processing.

In the software architecture of the module we identify three distinct submodules:

- a first control module responsible for accessing the Messaging Server, starting new jobs, stopping them and sending alerts;
- a second module used for stream identification and download (here an external tool is used, such as the MPlayer);
- a third module which repacks the downloaded stream into a format recognized by the access tools.

When available, a streaming crawler connects to the Messaging Server to request a new streaming URL to capture. Upon receiving the new URL, an initial analysis is done in order to detect some parameters, among others the type and the duration of the stream. Of course, if the stream is live, a fixed configurable duration may be chosen.

After a successful identification the actual download starts. The control module generates a job which is passed to the MPlayer along with safeguards to ensure that the download will not take longer than the initial estimation.

After a successful capture, the last step consists in wrapping the captured stream into ARC/WARC format and in moving it to the final storage.

### 1.1.2 Test collections and preliminary results

We conducted several test crawls using the new capturing module on the GOV.UK collection. This UK governmental Web site collection is regularly crawled and enriched monthly by the European Archive. During the last 2 monthly crawls, the capturing module was successfully used to retrieve the multimedia content found on these Web sites and which it would not have been possible to capture previously. The table below gives some examples of the discovered URIs.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Web site</th>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>rtsp</td>
<td><a href="http://www.epsrc.ac.uk">http://www.epsrc.ac.uk</a></td>
<td>rtsp://m.groovygecko.net/groovy/epsrc/EPSRC_Aroll_041208_hb.rv</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rtsp://m.groovygecko.net/groovy/epsrc/Pioneers09_hb.rv</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>mms</td>
<td><a href="http://www2.cimaglobal.com">http://www2.cimaglobal.com</a></td>
<td>mms://groovyg.edgestream.net/groovyg/clients/Markettiers4dc/Video%20Features/11725/11725_cima_employers2_HowToTV.wmv</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
In the next test round we plan to extend the Web sites collection to some particular video broadcast sites, such as [http://www.dw-world.de](http://www.dw-world.de), where the number of video streams is considerably larger.
1.2 Link Extractor

The link extractor module is designed to 'extract' links that the standard link extractors cannot find. Link extraction is the process of discovering links in a web page. Web pages can contain dynamic content generated by javascript or plugins, for example Flash. Standard link extraction techniques, all boil down to looking in the source code of the page for things that look like URL’s.

The link extractor runs as a Web service: http://www.example.com/extract/<URL>

The return value is a JSON encoded list of discovered URL's. Hanzo will operate the Web service.

For testing purposes the web-service will be called by a plug-in to Heritrix that acts as an extractor. The extractor will call the Web service and add the returned URLs to the discovered link list.

1.2.1 Performance indicators, metrics and methods

Baseline performance is a moving target but should be considered the current capabilities of the two crawlers we use - Heritrix and the Hanzo crawler. Both crawlers continue to improve their ability to detect URL’s but the link extractor should always be able to find more URLs on any page that uses dynamic link generation.

Evaluation will be on a small set of carefully chosen Web pages. The Web pages should be measured as to how much dynamic link generation they do as a rough estimate of their total number of links. This value will range from 0% - totally flat HTML to close to 100% where almost all the links on the page are generated by javascript.

For each page in the test collection there should be two human generated measures:

- % page dynamic
- Estimated number of links

The testing then should generate two other figures:

- Standard number of links discovered by crawler
- Enhanced number of links discovered by crawler using WP2 link extractor
1.3 Spam filter

The Spam Filter module described in D3.1 Archive Filtering Technology V1 takes WARC format crawls as input and outputs a list of the sites with a predicted spamicity (strength of similarity in content or behaviour to spam sites) as a value between 0 and 1.

For testing purposes, the output of the evaluation script is a Weka classifier output that contains a summary of the relevant performance measures over a predefined labelled test set.

Since part of the testing is performed over the Web Spam Challenge 2007 and 2008 data sets, the baseline performance is published in the proceedings of the Adversarial Information Retrieval Workshops AIRWeb 2007 and 2008 available at http://airweb.cse.lehigh.edu/2007/proceedings.html
http://airweb.cse.lehigh.edu/2008/proceedings.html

Since part of the LiWA spam filtering features are omitted from the testing script due to the hardware limitations of the test server and since the LiWA system is not particularly tuned for the Web Spam Challenge data sets, performance slightly below the best Challenge participants is acceptable.

The software relies on the open source machine learning toolkit Weka available at http://www.cs.waikato.ac.nz/ml/weka/. Classifier accuracy depends on the implementation of several methods in Weka, hence the version of Weka as described in the testing script is required. The open source Warctools http://code.google.com/p/warc-tools/ matching the input WARC version is also required.

Performance evaluation is tested in the following steps. Given the correct performance of the unit tests, the entire system test produces the Weka output containing the required performance measures. The V1 testing script is adapted to the hardware limitations of the test server and hence features with large internal memory requirements to compute are omitted from the test.

1.3.1 Performance indicators, metrics and methods

The main performance indicators for the spam filters will measure the correctness of the predicted spamicity.

The Weka output contains in part self-explanatory figures such as “Correctly Classified Instances”. Other measures are described next.

Accuracy

Gives a measure for the overall accuracy of the classifier:

\[
\text{accuracy} = \frac{\text{number of correctly classified instances}}{\text{number of instances}}
\]
**Precision and recall**

For X being spam or honest,

\[
\text{precision}(X) = \frac{\text{number of correctly classified instances of class } X}{\text{number of instances classified as belonging to class } X}
\]

\[
\text{recall}(X) = \frac{\text{number of correctly classified instances of class } X}{\text{number of instances in class } X}
\]

**F-measure**

A popular measure that combines Precision and Recall is the weighted harmonic mean of precision and recall, the traditional F-measure or balanced F-score:

\[
F = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\]

This is also known as the F1 measure, because recall and precision are evenly weighted.

**Confusion matrix**

Confusion matrices display the distribution of correct and incorrect instances. Typical Weka output contains the following:

```text
=== Confusion Matrix ===

a b <-- classified as

7 2 | a = spam
3 2 | b = honest
```

Weka classifies instances into two possible classes: spam and honest. The columns represent the instances that were classified as that class. So, the first column shows that in total 10 instances were classified spam by Weka, and 4 were classified as honest. The rows represent the actual instances that belong to that class.

**AUC (Area Under receiver operator characteristic Curve)**

The AUC, a value between 0.5 (random) and 1 (perfect) is the area under the receiver operating characteristic (ROC) curve. The ROC is a plot of the fraction of true positives (TPR = true positive rate) vs. the fraction of false positives (FPR = false positive rate).

The Web Spam Challenge 2007 benchmarks involve the F and AUC measures while the Web Spam Challenge 2008 uses AUC only. Due to its stability we also consider AUC as the main performance indicator that we will base our reports on.

**1.3.2 Test collections**

For evaluating the performance of this module, a test collection will be used that consists of a sequence of periodic recrawls made available for the purposes of spam
filtering development. We preprocessed the data set of 13 UK snapshots (UK-2006-05 … UK-2007-05 where the first snapshot is WEBSPAM-UK2006 and the last is WEBSPAM-UK2007) provided by the Laboratory for Web Algorithmics of the Università degli studi di Milano [6] supported from DSI-DELIS project. We selected a maximum of 400 pages per site to obtain approximately 40GB WARC files for each snapshot. The LiWA test bed consists of more than 10,000 manual labels that proved to be useful over this data.

1.3.3 Test results

We conducted the test on 16-April-2009 over the WEBSPAM-UK2007 data set converted into WARC 0.19 by SZTAKI as part of the LiWA test bed. For testing and training we used the predefined labeled subsets of WEBSPAM-UK2007. The results of the test are as follows.

<table>
<thead>
<tr>
<th></th>
<th>Training set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>4000</td>
<td>2053</td>
</tr>
<tr>
<td>True positive</td>
<td>236</td>
<td>72</td>
</tr>
<tr>
<td>True negative</td>
<td>2461</td>
<td>1242</td>
</tr>
<tr>
<td>False positive</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td>False negative</td>
<td>1301</td>
<td>715</td>
</tr>
<tr>
<td>Correctly classified</td>
<td>2697</td>
<td>1314</td>
</tr>
<tr>
<td>Incorrectly classified</td>
<td>1303</td>
<td>739</td>
</tr>
<tr>
<td>Precision</td>
<td>0.154</td>
<td>0.091</td>
</tr>
<tr>
<td>Recall</td>
<td>0.992</td>
<td>0.75</td>
</tr>
<tr>
<td>$F_1$</td>
<td>0.266</td>
<td>0.163</td>
</tr>
<tr>
<td>AUC (ROC area)</td>
<td>0.895</td>
<td>0.756</td>
</tr>
</tbody>
</table>

When comparing to the baseline, we use the AUC measure since all other measures are sensitive to changing the threshold used to separate the spam and non-spam classes in the prediction. The best performing Web Spam Challenge 2008 participant reached an AUC of 0.85 while our result reached 0.80. Some of the research codes still require industry level tested implementations and will gradually be added to the LiWA code base. Also some of the most memory intensive features were not generated since the 3GB memory available on the test server was insufficient for the current implementation. We are also expecting progress in reducing the resource needs for the feature generation code.
1.4 Temporal Coherence

The temporal coherence framework needs to be capable of dealing with properly as well as improperly dated contents. Depending on a Web server’s accuracy, different coherence-optimizing crawling strategies need to be evaluated. To this end, our tests take place in a simulation environment that employs the same algorithms we have developed for the temporal coherence processor of Heritrix, but gives us full control over the content (changes) and allows us to perform extreme tests (in terms of change frequency, crawling speed and/or crawling strategy). Thus, experiments employing our coherence-ensuring crawling algorithms can be carried out with different expectations about the status of Web contents and can be compared against well-defined ground truth.

With respect to comparing our coherence-improving crawling strategy there is no pre-existing competitor and, thus, no appropriate benchmark. Conventional implementations of archiving crawlers are based on a priority driven variant of the breadth-first-search (BFS) crawling strategy and do not incorporate revisits. However, “virtual time stamping” is unavoidable in order to determine coherence under real-life crawling conditions. Therefore, standard tests of our crawling algorithms are comparisons with crawl-revisit pairs either implemented as BFS-LIFO (breadth-first-search combined with last-in-first-out) or as BFS-FIFO (breadth-first-search combined with first-in-first-out). In addition, we are able to identify baselines for optimal and worst-case crawling strategies, which are obtained from full knowledge about changes within all pages during the entire crawl-revisit intervals. These baselines are only considered as theoretically achievable limits.

Experiments take place within our simulation environment based on synthetically generated Web sites, varying in size, topology and change behavior. Thus, the improvement of the coherence obtained by using our crawling strategies can be tested on virtual Web sites from small to huge and from highly dynamic to almost static. Consequently, the simulation environment is able to mimic real-life crawling lasting from a few hours to several days or weeks.

1.4.1 Performance indicators, metrics and methods

The main performance indicator for the temporal coherence module will be the fraction of accurately dated content and the crawl cost measures. The cost of the crawl can be measured by parameters such as the number of downloads, bandwidth consumed or crawl duration. As we have shown in the temporal coherence section of deliverable D9.1, full guarantees on properly dated content require a more sophisticated “virtual time stamping mechanism”. This implies that temporal coherence and crawl cost are contradictory objectives. However, it is possible to ensure and evaluate proper dating of contents and reduce the crawl cost in a subsequent step. Therefore, in our simulation environment we will measure the achievable coherence, performance and efficiency of our algorithms developed.
In terms of coherence, we measure the percentage of content in a Web site that is coherently crawled (that means “as of the same time point or time interval”). The performance of our algorithms is evaluated against comparable modifications of conventional crawling strategies such as BFS-LIFO (breadth-first-search combined with last-in-first-out) or BFS-FIFO (breadth-first-search combined with first-in-first-out). In addition, we provide theoretically achievable baselines for optimal and worst-case crawling strategies, which are obtained from full knowledge about all changes within a Web site. With respect to efficiency, we quantify the reduction of crawl cost based on partial re-crawling strategies with respect to a given (but freely adjustable) achievable coherence threshold.

All the aforementioned metrics are evaluated against different Web sites varying in size, topology as well as change behavior (e.g. medium sized crawls of Web sites consist of 10,000 - 50,000 contents). All parameters are freely adjustable in order to resemble real life Web site behavior. Each experiment follows the same procedure, but varies in size of Web contents and change rate. We model site changes by Poisson processes with page-specific change rates. These rates can be statistically predicted based on page type (e.g., MIME types), depth within the site (e.g., distance to site entry points), and URL name (e.g., manually edited user homepages vs. pages generated by content management systems).

Each page of the data set has a change rate $\lambda_i$. Based on a Poisson process model, the time between two successive changes of page $p_i$ is then exponentially distributed with parameter $\lambda_i$:

$$P[\text{time between changes of } p_i \text{ is less than time unit } \Delta] = 1 - e^{-\lambda_i \Delta}$$

Equivalently, the probability that $p_i$ changes $k$ times in a time interval of length $n$ follows a Poisson distribution:

$$P[k \text{ changes of } p_i \text{ in one time unit}] = \frac{\lambda_i^k e^{-\lambda_i}}{k!}$$

Within the simulation environment a change history is generated, which registers every change per time unit. The probability that page $p_i$ changes at $t$, then is:

$$P[p_i \text{ has at least one change}] = 1 - e^{-\lambda_i}$$

1.4.2 Test collections

For evaluating the performance of this module, we do not require a specific test collection, since data sets for testing our algorithms are automatically generated within the simulation environment. The properties of the data sets are specified via freely adjustable Web site parameters (e.g. size, change behavior, out-degree, etc.). However, test collections of other work packages may be applied for functional tests of the LiWA temporal coherence processor in Heritrix, as long as it is possible to crawl their content with a real-life crawler.
1.5 Terminology Evolution

The aim of the terminology evolution module is the automatic detection of the evolution of terminology over time. The problem can be split into three different sub-problems: terminology snapshot creation, merging of terminology snapshots and mapping concepts to graphs. These major steps can further be divided into smaller processing units as follows:

- Terminology snapshot creation
  - Document pre-processing (HTML, PDF, ..)
  - Part of Speech Tagging & Annotations
  - Terminology Extraction
  - Co-occurrence Analysis
  - Term Clustering
  - Creation of Term-Concept-Graphs (TCG)
- Merging Term-Concept-Graphs
  - *Detailed processing steps will be developed in the 2nd project year*
- Detecting Terminology evolution and mappings
  - *Detailed processing steps will be developed in the 2nd project year*

As input, our terminology extraction tool receives a directory containing the part of archive to be analyzed. The documents in the archive are parsed sequentially and cleaned of stopwords. In this step we also perform part-of-speech (POS) tagging and lemma annotation using the open-source TreeTagger tool (http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/). At the end of this step we obtain an index of the terms found in the collection. For each term we index its position information in the containing documents, and also its stemmed form, its part of speech and its lemma.

From the index we build co-occurrence matrices for lemmas assigned to the terms in the collection and extract a term frequency dictionary.

In the next stage we apply a curvature clustering algorithm on the co-occurrence matrix and the dictionary. By extracting and comparing clusters for different time intervals we aim at identifying terminology evolution.

The main performance indicators for the terminology evolution will be:

1. the completeness (coverage) of extracted terminology from different time periods
2. the usability of the semantic terminology browser.

The performance measurement depends on the quality and coverage of the selected test collection. For having a long-term archive, the London Times archive from 1785 – 1985 will be used. However, as these are uncorrected OCR scans, the OCR errors can influence the results.

The first performance measurements will be conducted for the deliverable D6.5.
1.5.1 Performance indicators, metrics and methods

The definition of performance indicators for terminology evolution is a challenging task. The reason is that no ground truth for terminologies exists. Most available dictionaries, ontologies or thesauri just cover parts of today’s language. But as far as we know no electronic historic dictionaries or mapping between different periods of time are available. So there exists no reference point to compare the quality of our results. Therefore we will focus on quantitative measures and usability studies of the terminology evolution browser. The main performance indicator for the terminology evolution will be

1. the completeness (coverage) of extracted terminology from different time periods and
2. the usability of the semantic terminology browser.

Completeness of extracted terminology per time period

The completeness of the extracted terminology will be measured by analyzing the number of extracted terms of a dictionary based POS tagger. The assumption behind it is that the number of identified and extracted terms from recent documents by using a recent dictionary will be higher than in old documents using the same dictionary. By using the terminology evolution detection technology developed in WP5 and deriving the mappings between terms, the dictionary can automatically be extended with useful similar terms. If these extensions are successful the number of identified terms in older documents will increase.

The measurement process is depicted in the picture below. In the first step a given document collection will be divided into time periods. Afterwards a sub-collection will be selected processed by the POS tagger. Finally, the number of identified nouns is counted. This will be repeated for every sub-collection.

![Diagram](image)

*Figure 1: Performance Measurement for Completeness of extracted terminology per time period*

This process will be repeated regularly to measure the performance improvements. The results will be summarized in a table as shown below.
<table>
<thead>
<tr>
<th>Time period</th>
<th># Nouns – 1st Run</th>
<th># Nouns – 1st Run</th>
<th># Nouns – 1st Run</th>
</tr>
</thead>
<tbody>
<tr>
<td>1800 – 1900</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1900 – 1920</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1920 – 1950</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1950 - 2009</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Usability of the semantic terminology browser**

The second performance evaluation will be an end user test about the usability and practicability of the semantic evolution browser. The semantic evolution browser (defined in Task 5.6) allows end users to visually understand the evolution of terminologies over time. In addition, it will allow search within an archive by using a prototype of terminology evolution aware retrieval engine (defined in Task 5.7). An initial version of the semantic evolution browser will be available after the 2nd year. The terminology evolution aware retrieval engine will be developed in the 3rd year of the project. Therefore the details for the performance measures will be defined after the architecture development of these components.

1.5.2 **Test collections**

A major requirement for the test collection is that it covers long enough time periods to ensure the availability of terminology evolution. Web archives cover at most 10 years. Therefore there is a certain risk that only limited terminology evolutions will be identified. However, once the technology reaches a higher level of maturity, measurements on Web archives will also be conducted.

For our first stage tests we have selected the Times Newspaper archive, consisting of all published articles spanning from 1785 to 1985. The articles are in XML format. We have noticed an increase of the size of published text from the 18th century to the 20th century. The size of the archive of articles published in a year from the 18th century was, on average, 163MB; in the 19th century the average size went up to 725MB and in the 20th century to 936MB. The actual size is around three times larger than the archive size.

In the recent years there are around 72000 articles per year on various topics. Each article consists of metadata, text and a keyword index. The textual information amounts to around 10% of the article's size (which includes also xml tags).

For one month in 1985, with 6141 articles amounting to 153MB, after text extraction and stop word removal we ended up with 14MB of data which led to a dictionary of 6.3MB.
2 Integration testing on the testbed

Integrating the new modules in the archiving workflow naturally begins with testing the usage of the internal modules with the current version of the Heritrix crawler. The integration of the external modules will proceed after the accomplishment of several functional tests locally performed by each workpackage, since the relevant data sets for testing specific functionalities greatly differs at this stage.

Some external modules are also in an early stage of development, merely after the first version of the technologies, and the interaction with the other modules is not yet completely defined.

In the first integration phase we therefore focused mainly on integrating the internal modules developed in LiWA: the Temporal Coherence module and the Rich Media Capturing module.
2.1 Integration of the Temporal Coherence module

The first experiences on the testbed are related to the integration of the Temporal Coherence module into the original version of Heritrix crawler (version 2.0.2), installed on the testbed.

There are several actions that have been taken for adapting and integrating this module into the current version of the crawler. They are summarized in the following steps, along with some remarks on the improvements to be taken into account for the future development.

2.1.1 Step 1: Building the “Heritrix-LiWA.jar” in the Cruise Control system

The artifacts generated by CruiseControl when building the Temporal Coherence module are the archived Java classes (Heritrix-LiWA.jar) and the archived Web application (Heritrix.war).

2.1.2 Step 2: Setting the “liwa.parameters”

The essential section of this configuration file is dedicated to the parameters used by the database connections:

```bash
#db
liwa.db.user=liwa
liwa.db.password=liwa
liwa.db.protocol=jdbc:oracle:thin
liwa.db.host=debug.europarchive.org
liwa.db.port=1521
liwa.db.sid=XE
```

The name of the host machine is set to the name of the testbed and the connection port (1521 by default) has to be open for external connections. The default value set for the Session-ID parameter (in the open version of Oracle, installed on the testbed) is “XE”, as defined in the settings file “oracle_env.sh”:

```bash
ORACLE_HOME=/usr/lib/oracle/xe/app/oracle/product/10.2.0/server
export ORACLE_HOME
ORACLE_SID=XE
export ORACLE_SID
```

One possible issue related to the configuration file concerns its place in the Heritrix deployment, for instance inside the /conf subfolder or added to the classpath variable used for launching the crawler. This file should be externalized from the jar file generated by the Temporal Coherence module, in order to adjust the necessary settings for the local deployment.
2.1.3 **Step 3: Adding the LiWA Processor**

In order to use the Temporal Coherence module, the LiWA Processor must be added to the list of global processors in the Frontier. The “web.xml” descriptor file must be modified accordingly, by adding the corresponding listener:

```xml
<listener>
  <listener-class>
    org.liwa.coherence.events.HeritrixListener
  </listener-class>
</listener>
```

2.1.4 **Step 4: Updating the database schema and the SQL script**

Some minor updates had to be done to the previous database schema created on the testbed. T_PAGES and T_LINKS tables were updated.

2.1.5 **Step 5: Running the GraphML Exporter**

The GraphML Exporter is an auxiliary module used for generating the graph structure associated to the crawl of a Web site. For a given crawl_ID, the data is collected from the database and exported in a graph structure, in order to obtain a more comprehensive view on the structure of the Web site and also on the crawl process.

The module is developed in Java and it exports the crawl graphs in GraphML, an XML-based file format for graphs. Some minor issues related to different versions of the Oracle’s jdbc library have been fixed.

2.1.6 **Step 6: Installing the visualization tool yEd**

yEd is a very powerful graph editor that can be used to quickly and effectively generate drawings and to apply automatic layouts to a range of different diagrams and networks. yEd makes full use of the yFiles library. This is a Java class library enabling the viewing, editing, optimizing, drawing, and animating of a wide range of diagrams, networks, and other graph-like structures ([http://www.yworks.com/en/products_yed_about.html](http://www.yworks.com/en/products_yed_about.html)).

The use of the visualization tool gives a qualitative measure of the crawl, from the temporal coherence perspective. Based on the information stored in the database during the crawling process and exported afterwards in an XML format, the visualization tool is able to offer a graphical representation of the Web sites, pointing out different patterns of incoherencies.

These insights on the crawls are particularly valuable from the crawl engineer perspective, since they greatly help in identifying the Web site incoherencies.

From a practical point of view, EA is planning to use the visualization tool on a selection of Web sites to be crawled from the GOV.UK collection. The goal of this exercise will be to build a collection of “coherence patterns” frequently encountered in the structure of the Web sites. This information will help in further tuning the crawl parameters in Heritrix for a more coherent capture.
2.2 Integration of the Rich Media Capturing module

The second internal module to be integrated in the LiWA testbed is the Rich Media Capturing module. The following steps summarise the actions that have been taken for adapting and integrating this module into the current version of the crawler.

2.2.1 Step 1: Building the “HeritrixStreamingPlugin-LiWA.jar” in the Cruise Control system

The artifact generated by CruiseControl when building the Rich Media Capturing module is represented by the archived Java classes in HeritrixStreamingPlugin-LiWA.jar.

2.2.2 Step 2: Setting the “config.properties”

The next step in adapting the module to the framework consists of updating the main configuration parameters. They are set in the “config.properties” file, selecting the multimedia types that can be captured by the module, as well as the address (host + port) of the external applications in charge of downloading the content:

```java
# schemas watched by the module
active_schemas=rtsp,mms
amqp_host=HOST
amqp_port=PORT
amqp_queue=streaming_in
```

2.2.3 Step 3: Adding the StreamingURI Processor

The implementation of the module is represented by a new plugin for Heritrix called StreamingURI. The plugin intercepts all requests going to streaming resources (normally the crawler would reject them as it does not know how to process them) and passes them to a cluster of computers running the stream capturing tool (MPlayer). The capturing messages are generated using the following structure:

```java
String MESSAGE_SKELETON="<job>
  
  "
  <client>[$client]</client>
  
  "
  <command>[$command]</command>
  
  "
  <crawl-id>[$crawl-id]</crawl-id>
  
  "
  <patch-for>[$patch-for]</patch-for>
  
  "
  <seeds>
  
  "
  <seed>[$seed]</seed>
  
  "</seeds>
  
  "
  <scope>[$scope]</scope>
  
  "
  <arc-prefix>[$prefix]</arc-prefix>
  
  "
  <politeness>2</politeness>
  
  "
  <follow-robots>false</follow-robots>
  
</job>";
```
In order to activate the **StreamingURI** plugin, the configuration file of Heritrix (conf/heritrix.properties) also needs to be modified. Moreover, the property org.archive.net.UURIFactory.ignored-schemes in the crawling parameters should exclude the streaming schemes. The rtsp and mms schemes should therefore be removed from the list of ignored schemes and added to the list of accepted schemes, like for instance:

```
org.archive.net.UURIFactory.schemes = http,https,dns,invalid,rtsp,mms
```

The plugin should be connected to the list of pre-fetch processors. Moreover, a new `ExternalImplDecideRule` should be added to the crawl order with the name of the implementation class: `org.europarchive.StreamingURI`.

The decision setting for the new processor rule should be REJECT.
LiWA Applications

In order to show the advantages of the advanced Web archiving approach for libraries and archiving organizations, two applications, named "Streaming" and "Social Web", were designed based on the requirements identified in close collaboration with potential users within the consortium in WP1. On the basis of the design and requirements, mock-ups of these applications were constructed showing the planned functionality and the possible user interaction of the considered application. The mock-ups were demonstrated and discussed with potential users of the application and demonstrated at the first review.

In the next paragraphs, feedback received during the mock-up creation process and further evaluation steps will be described.

3 “Streaming” Application

The WP7 application “Streaming” shows how LiWA technology fits in the workflow of the Sound and Vision audiovisual archive. The focus in this application is on archiving Dutch (streaming) audiovisual web content including its context and to link both audiovisual data and context data to traditional archive collections. The goal of the application is two-fold: (i) archiving audiovisual content from the web and making this content accessible for end-users, and (ii) deploying web-content as context information to help documentarists or automated processes to assign relevant labels to assets already in the archive.

An application mock-up was created on the basis of the analysis of target group requirements for the LiWA applications (D1.1).

3.1 Feedback on mock-up

The mock-up was made available via an URL (http://www.slidingdoors.nl/LiWA) and a walkthrough with a questionnaire for feedback was send to users in the target groups and the LiWA consortium members. Below, the walkthrough and the questionnaire feedback are provided.
3.1.1 Walkthrough

[Beeld en Geluid-logo = back to home]
In the first screen, all results for a query 'skateboard' are shown as a list. Sorting is possible by relevance and date. In this example, the most relevant item is marked and is ready to play in the right part of the screen. Metadata of the stream is also available.

[click on 'audio']
2) This screen resembles the screen with video results, but this time it exclusively shows the audio. On the right, a player with a visual equaliser is included.

[click on 'audio']
3) This screen features the matching Web page as well as web video (archived from Youtube etc.) results. The right-part of the website changes depending on the thumbnail that is selected. [Click] on the thumbnail of the 'latest video' to see the mouse-over effect. [Click] on the thumbnail of the website to select it and see that the player changes. This thumbnail has a pop-up as well as [click] to see it. In the actual application, it will work by hovering over with your mouse.

[click on the miniature] (see image) It shows the Web archive results.

4) This page lists all pages within the chosen domain featuring this search term. Under each of the screens, a text fragment is shown. The search term is highlighted. The red arrows on the right can be used to navigate to the next pages in the domain.
5) The Web page is loaded as stored in the Web archive. All “original” navigation is active. Search terms are highlighted. The upper part of this view still features the LiWA skin.

6) This is an alternative way of exploring the collection. It is inspired by the “video wall” from Blinkx (http://www.blinkx.com)

### 3.1.2 Questionnaire

The questionnaire below was used to gather feedback from target group users and consortium members. On the basis of the received feedback, the final version of the mock-up was created as consolidated via http://www.slidingdoors.nl/LiWA and shown during the first project review in February 2009.

<table>
<thead>
<tr>
<th>Area</th>
<th>Comment (usability, ideas how to present it, etc. etc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search and result pages</td>
<td>Where do the dependencies (contents – search terms) come from? Are they derived via so called tags? What happens if such tags don’t exist? Is search of a dedicated URL possible? Is there any temporal dimension?</td>
</tr>
<tr>
<td>Web archive results page</td>
<td>How are results ordered? How can the temporal dimension be taken into consideration?</td>
</tr>
<tr>
<td>Web page view (plus metadata view)</td>
<td>I’ve no idea why and how the web page shown is directly associated with the medium. The medium appears to be linked only, but not embedded. There might be many more media associated with it, which makes the search result (almost) arbitrary.</td>
</tr>
<tr>
<td>Navigation between pages</td>
<td>This is hard to guess from the mock-up. Temporal coherence might be a serious issue as there might exist many outdated or broken links. It is absolutely unclear to me how and to what extent you intend to overcome this problem.</td>
</tr>
<tr>
<td>Design features like mouse-over, highlighting. Zooming, video wall</td>
<td>Looks nice, a little too few LiWA look &amp; feel, temporal coherence appears to be completed disregarded.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Screen layout (which components should be located where? Use of colours)</td>
<td>---</td>
</tr>
<tr>
<td>General comments on usability</td>
<td>This is hard to guess from the mock-up. In general, I do see problems in (arbitrary) mixing up searches for tags (of streams) with those dealing with URLs (of web pages), which is the case up to now.</td>
</tr>
<tr>
<td>Comments on technical feasibility</td>
<td>In reality I see problems in keyword based retrieval of web pages as they would have to be indexed in before. With tagged contents this problem is less dominant, but still an issue. Temporal coherence might be a serious issue as there might exist many outdated or broken links.</td>
</tr>
<tr>
<td>Comments on link with research LiWA (notably, can we visualise semantic evolution?)</td>
<td>Whom to analyze and index the media according to the content they are dealing with? What’s the level of granularity: Tags, Metadata-tags or the plain content? Temporal coherence might be a serious issue as there might exist many outdated or broken links. It is absolutely unclear to me how and to what extent you intend to overcome this problem. How do you plan to realize the temporal dimension of web contents?</td>
</tr>
<tr>
<td>Additional functionalities that would be good to include.</td>
<td>A temporal way-back machine in order to navigate along a time line.</td>
</tr>
<tr>
<td>Other Comments</td>
<td>Design looks nice, but too little LiWA look &amp; feel. Problems might arise when streaming media and the web-sites they are linked from are considered without their temporal dimension.</td>
</tr>
</tbody>
</table>

### 3.2 Application evaluation plan

On the basis of the mock-up design, the WP7 application “Streaming” will be implemented (T7.4, T7.6) and evaluated (T7.5, T7.7). Evaluation results will be reported in D7.2 and D7.3 for the first and second version of the application, respectively. Below the evaluation plan for the application versions is described.
3.2.1 Evaluation version 1

The first version of the application targets the crawling of the envisaged broadcast domains, the capturing of audiovisual content from the crawled websites, the storage of the various content types (video, audio, websites) and a provisional web-based browsing and search functionality.

Evaluation for the first version will address:

(i) non-functional requirements:
   • the volume and quality of the crawls for the broadcast domains (e.g., is spam filtering a requirement for the application domain, how is temporal coherence for the application domain)
   • the quality of AV capturing for the domain

(ii) functional requirements:
   • storage and indexing of crawls and AV content:
     o can web context be extracted to “tag” the AV content for indexing
     o storage and indexing of multiple content types
   • browsing and searching of crawls and AV content
     o content type specific search
     o (re-) playability of AV content

3.2.2 Evaluation version 2

In the second version of the application, the focus will be on (i) adapting the application if necessary on the basis of the evaluation results of the first version and (ii) finalizing the (Web) interface that allows the browsing and searching of all content types according to the design as formulated in the mock-up.

Evaluation of the second version will update the evaluation of the first version where necessary but it will especially focus on the user evaluation for the target groups. This evaluation should provide insight on:

(i) to what extent the application fits with the requirements of a “general public” user group, representing public that is interested in AV related Web-archiving service provided by Sound and Vision

(ii) to what extent the application and underlying technology (tools and interfaces) fit with the requirements of a professional user group, both within Sound and Vision and at other institutes in the user group such as the Dutch Documentation Centre for Dutch Political Parties (crawl engineers, but also archivists that want to use the application as a tool for providing context
during annotation).

(iii) to what extent the Web content provided by the LiWA technology can be used effectively for the automatic linking of web-content to archival content.

Depending on the user group, users will be presented with the application, LiWA tools and a group specific questionnaire. Results will be reported in D6.8.
4 “Social Web” Application

The WP8 application “Social Web” shows how LiWA technology fits in the workflow of an active Web archiving institution, by considering a real-life scenario of the National Library of the Czech Republic. The application is designed as a set of independent modules developed in LiWA work packages 2 (capture of rich content), 3 (data cleansing and noise filtering), 4 (archive coherence) and 5 (semantic evolution). The modules can be readily integrated with existing Web archiving workflow management tools. A Web archiving institution can choose to deploy all of the modules or just some of them, depending on its needs and particular workflow. The application is designed as generic and can be used for archiving any type of web content. However, the concept will be demonstrated on a case of the social Web, as this part of the Web poses the real challenge for Web archives.

An application mock-up was created on the basis of the analysis of target group requirements for the LiWA applications (D1.1).

4.1 Feedback on mock-up
A walkthrough of the mock-up with a questionnaire for feedback was sent to users in the target group and the LiWA consortium members. The walkthrough and feedback from the questionnaire are provided below.

4.1.1 Walkthrough

The application will be integrated into WA Admin, which is an in-house built workflow management tool developed by the National Library of the Czech Republic. WA Admin keeps track of all Web sites that the library considers for inclusion in its Web archive. It is organized around the fundamental stages of the workflow: nominating websites, appraisal, selection, negotiation with publishers, cataloguing and quality assurance.

The application modules will be utilized in the quality assurance stage of the process.
Checking previous instances with the LiWA temporal difference analyzer. Breach in continuity can be due to the lack of completeness but also to radical change in the site technology, which are also good indicators of potential crawling problems. This helps focus QA for repetitive crawls, for instance limiting QA time in case no significant change is identified here.

Clicking in the graph directly opens the archived version of the website.

Finding temporal incoherencies with the coherence graph. This graph indicates where in the site some temporal incoherence patterns can be found, and allows the QA operator to click on nodes to see the problematic sections of the site.

Clicking in the graph directly opens the archived version of the website.

WP2 advanced link extraction:

For sections which were not harvested due to Javascript etc., the curator clicks a button that runs the virtual browser. The virtual browser generates a list of links on the page. Missing links are added to the seed list for a re-crawl or next harvest.

The top half of the screen shows a challenging site to archive, while the bottom half displays a list of discovered links.
Quality assurance relies on the spam detection engine. Whenever a curator finds spam or other irrelevant content during this stage, he can use the spam module to mark it as such and document the relevant feature set for enhancing detection accuracy. The curator may also manually accept or reject the decision of the automatic filter.

### 4.1.2 Questionnaire

<table>
<thead>
<tr>
<th>Area</th>
<th>Comment (usability, ideas how to present it, etc. etc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temporal coherence module</td>
<td>Timeline mode: What is the granularity of the data? How fine can it be made? Will it be possible to see how much of the content has changed or disappeared? This would make it really useful for deduplication purposes…</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Link extraction module</td>
<td>The module needs to be plugg-able into 3rd party applications including open-source applications.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| |                        | We may not be interested in crawling all of the
extracted links. Some of them can be insignificant. It would be great to have an option allowing to decide whether the extracted links are added to the seed list or ignored.

<table>
<thead>
<tr>
<th>Module/Feature</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spam and noise-filtering module</td>
<td>Will the module allow marking spam in the page and choosing whether to delete it or leave it in?</td>
</tr>
<tr>
<td>Design features like mouse-over, highlighting.</td>
<td>In temporal coherence analyzer, timeline mode – on mouse-over display date of the harvested instance</td>
</tr>
<tr>
<td>Screen layout (which components should be located where? Use of colours)</td>
<td>---</td>
</tr>
<tr>
<td>General comments on usability</td>
<td>All modules should allow easy follow-up actions, e.g. from the link extraction module, add URLs to the seed list</td>
</tr>
<tr>
<td>Comments on technical feasibility</td>
<td>Needs to be easily pluggable into platform-independent environment (usable API, programming language independent)</td>
</tr>
<tr>
<td>Additional functionalities that would be good to include.</td>
<td>See feedback for individual modules above</td>
</tr>
<tr>
<td>Other Comments</td>
<td>---</td>
</tr>
</tbody>
</table>

### 4.2 Application evaluation plan

Based on the mock-up design, the WP8 application 'Social Web Archiving' will be implemented (T8.4, T8.6) and evaluated (T8.5, T8.7). Evaluation results will be reported in D8.2 and D8.3 for the first and second version of the application, respectively. Below the evaluation plan for the application versions is described.

#### 4.2.1 Evaluation version 1

The first step of the implementation of the Social Web Archiving application is to integrate LiWA services, i.e. advanced link extractor, spam filter and archive coherence, into a production system. Subsequent to the integration, the implemented services will be tested to ensure proper functioning and documented. A set of problematic web sites containing Javascript, Flash, streamed video or audio content, mashups, polls etc. gathered in T8.3 will be used for testing.
Evaluation for the first version will address:

(i) non-functional requirements:
   • the performance (especially the performance of advanced link extractor)

(ii) functional requirements:
   • capturing of rich media content
   • spam filtering
   • providing coherent crawls

4.2.2 Evaluation version 2

The second version of the application aims to revise, refine and extend the Social Web Archive application in accordance with the test reports from the previous evaluation. The evaluation of the second version is also aimed at advancing user interface in order to meet user requirements. Therefore the application and a questionnaire will be submitted to potential users.
The second evaluation will update the evaluation of the first version where necessary.
Conclusion

This document presented the experiences gathered so far with the new technologies developed in LiWA. It combines the information concerning the evaluation of the individual LiWA modules and their integration on the testbed with the feedback received for the two application mock-ups.

At the end of the first year of the project, the version V1 of LiWA technologies was released. This first evaluation report summarises the functionalities of each module developed in LiWA, together with their specific performance indicators and evaluation methods. Where available, some preliminary test results are mentioned. A more comprehensive set of results will be presented in the next deliverable (D6.5), when the test cases and the test frameworks will be more elaborated for each LiWA module. The integration of the new modules on the testbed has been done incrementally, this document focusing on the adaptations brought to the internal modules in LiWA, in order to work with the current version of Heritrix.

The second part of the document presented the feedback received on the application mock-ups. A brief walkthrough is described for both mock-ups and the results of the feedback process is summarized by the responses of the questionnaires. Finally, plans for the further evaluation of the “Streaming” and “Social Web” applications are provided.

The second evaluation report (D6.8, scheduled for the next year) will emphasize the improvements brought by version V2 of the technologies and the evaluation of the implemented applications.