netarchive.dk

Integration of non-harvested web data into an existing web archive

Bjarne Andersen
Daily manager
netarchive.dk
bja@netarkivet.dk

Abstract

This paper describes a software prototype developed for transforming non-harvested web data into ARC-files[1]. The problems related to this will be analysed as will the problems in connection with a specific test done on a real delivery of web data delivered from the owner of a large Danish web site.

Furthermore the paper discusses how such data, once transformed into ARC-files, could be indexed with the open source WayBack[2] tool and what problems might arise when doing so. Indexing the data might reveal problems that potentially can affect the transformation process.

Keywords

Web data, ARC files, WayBack, transformation, indexing, mime-type, JMimeMagic.

Introduction

The scope of the prototype has been the transformation from a file system based collection of web data into a collection of ARC files. Once this process is finished the data is indexable with standard tools operating on ARC files, like the open source WayBack.

Transforming file system structures into ARC files is an important task for several reasons. Older web material may exist in smaller collections held by the website producers themselves (e.g. backup of web servers), within internet researchers' personal collections (maintained with various mirroring tools), or even within national collections as material collected by using older software like HtTrack or wget. Transforming such data into a more standardised format seems to be the only reasonable way of handling this in respect to both the access to and preservation of data for the future. Large scale archiving projects have proven the ARC format to be reliable for this purpose. Furthermore, tools for handling ARC files exist and probably will exist in the future as well since very large collections of web material stored in that specific format already exist.

---

1 Container file format developed especially for web data by Internet Archive: http://www.archive.org/web/researcher/ArcFileFormat.php

2 Open Source end user access tool with possibilities to browse historical web data including the time dimension http://archive-access.sourceforge.net/projects/wayback

Integrating non-harvested data into existing archives (of e.g. harvested material) would probably also require cataloguing the delivered and transformed data but that task is very dependant on requirements of the archive in which the new data should be part and thus, it is not the focus of this paper.

The prototype has been developed as part of a project funded by the Danish Ministry of Culture. The project is looking at old web material from the Danish national radio and television company (Danmarks Radio – DR[3]) as a pre-project for another project which focuses on documenting the story of the DR website from 1996 to 2006[4].

This paper describes the technical part of the pre-project. The content oriented part of the pre-project which concerns the contents of the test material and the browsing quality of the transformed data has been described in another paper[5].

The basics of an ARC file

To understand the problem of transforming directory structures and files into ARC files it is necessary to understand the very basics of the ARC file format. An ARC file is a collection of web objects bundled together to avoid millions of small files in a file system. Each entry in an ARC file is called an ARC record. The first ARC record is a file description record describing the ARC file itself and the following records are all web objects. The file description record allows the creator of the ARC file to add some metadata (in e.g. XML) to the file itself.

Each ARC record has the following basic structure:

- ARC header
- Content

An ARC file is nothing but a collection of ARC records. Thus an ARC file has the structure:

- ARC header
- Content
- ARC header
- Content
- Etc.

The header is a single line providing the following information about the content:

<URL> <IP-number> <date stamp> <mime-type> <size>

---

[3] http://dr.dk

[4] The project has its own website: http://drdk.dk/index_eng.htm

[5] The Archived Website as Historical Document: Principles, Rules and Recommendations. The 18th Nordic Conference for Media and Communication Research, Media History, Session: Technology, Media, History: Old and New Media. Helsinki, 2007. 23 pages http://www.uta.fi/laitokset/tiedotus/nordmedia2007/arbetsgrupper/mediehistoria.php

The <size> value is the size of the content – until the next ARC header begins. The other values should be selv-explanatory.

The content part of an ARC record usually has the HTTP-response headers followed by a blank line included at the top of the content itself – thus the headers have to be removed if you want to read the original content only. However, the headers do contain important information which is used by indexing and replay tools like the open source WayBack.

Delivery of sample material

The Danish national radio and television company was contacted to identify sample material for analysing and potentially use with the prototype. This led a delivery of five Gbytes consisting of historical web data on nine CD-ROMs and two DVDs. The analyses in this paper mainly present sample material from a technical point of view in order to reveal technical problems within the material and its transformation into ARC files.

The material can roughly be divided into the five groups presented below. Please notice that the groups have been derived from this specific sample material and that other samples could possibly reveal other categories.

The five groups are:

(1) **Static web server content**

The static web server content consists of ordinary files as they would have appeared on a web server. Content is defined as static if it only requires an ordinary web server and could most likely be delivered from any kind of web server (with small differences in file naming conventions between e.g. Windows (IIS) and Linux (Apache) and minor differences in the way the default index pages are named).

Such content is the easiest kind to handle because the files are almost directly transferable into ARC files with a good result – which is also the reason why we chose this kind of material to test the prototype on.

A closer look at the files and structures of the static web server content revealed interesting files. The web server seemed to host a lot of material which did not look as if it was meant for publishing. This was older material (directories renamed with the addition of ”_old”) or temporary content, e.g. different versions of graphics and pictures. This kind of material revealed what we defined as ”hidden” content. Most likely it had once been available on the web but since the naming of such files and directories showed that the files were of a temporary kind, and since nothing linked to these folders and objects it was only available for users who could guess the name of such ”hidden” files or folders.

The static web server content revealed only minor problems which will be described later in the paper.

(2) **Dynamic web server content**

The dynamic web server content is probably the most difficult of all to handle seen from a technical point of view. The dynamic files from a web server require a special environment on the web server (e.g. scripting capabilities like java, php or asp) or interaction with other services (e.g. database servers).

Examples of ".jhtml" files requiring a java setup on the server was found in the test material. These files require access to a database server that was not part of the delivery. Neither the database server itself nor the needed data (whether in the original or some other form) was included.

To be able to transform this kind of data into static ARC files it is necessary to setup the exact same server environment as intended for the original files. This is an almost impossible task unless you have a very sound description of the environment (server type, version, modules, etc.) This kind of description was not part of the delivery of the sample material. One could try to guess the setup by looking at the delivered files but it would be a very hard task to complete without the precise information about the original setup. In cases where the environment interacts with other services (like database-servers) the restoration is even more complicated.

The dynamic web server content is frequently mixed with static content since images and other static files are often part of the dynamic server side setup. Consequently objects can be found in deliveries with dynamic web server content that could be transformed directly into ARC files – but they will stand alone and be out of context. If such content is named in a meaningful way (like names of persons or places) it would be at least slightly useful when e.g. free text search includes file names.

(3) Presentations

The third kind of material in the test delivery can be categorised as presentations. These include mainly PowerPoint presentations (and some PDF-documents) made to present a new website or a re-design of an existing website (or part of a website).

The main problem with this kind of content is that it has never been published on a web server and thus, is more a kind of metadata about the creation of a specific website. Since it has never been published on a web server objects of this kind have no URL which is normally the main way of identifying material on the web.

The files are static files so making these into ARC files is not a problem seen from a technical point of view, but the content is not "real" web content and thus ought to be handled in another way and maybe stored in another system handling metadata about websites.

(4) Design files

This category is much like the category above except that this one refers to screen dumps (e.g. jpeg pictures), graphics files and design templates (e.g. Adobe Photoshop files).

This kind of material often represents different designs for the same website and thus is interesting material seen from a content oriented point of view.

Some of the test material included ”real” screen dumps with browsers, but the browser addresses almost always pointed at local web server installations (http://localhost/...) so it did not reveal the intended place for a potential publishing of the designs.

The design files suffer from the same problem as the presentations; they cannot be connected to a specific URL and thus, do not fit into the ARC files.

Both the design files and presentations suffer from another general problem, namely that they can be difficult to relate to a specific point of time. They suffer even more since they have never been published. A more detailed discussion about the timestamp problem will be presented in the prototype section of this paper.

(5) Backup files

The last category is a meta category because it covers files wrapped in containers created by backup programs. The main technical problem related to this is that you need the original backup program to restore the files or at least expert knowledge about the format allowing you to get the content out of the backup files and into ordinary files in a file system.

Once the content has been unpacked it will most likely fit into one of the other four categories. Within the sample material the backup files were from NERO backup and had been packed into a proprietary format. It is unknown whether this is an openly described format which allows you to write a new restoration program in case the original program is no longer available.

Other obvious examples of backup files are material packed into well-known containers like .zip or . tar and these are of course no real challenge (so far) since tools for handling such files are still around.

Test material

For the prototype we chose to work on material from the ”static web server content” category. This choice was made because static web server content is the easiest kind of material to handle and because we hoped that the transformation of this kind of material would give us quite good indexing and browsing results once a few other technical challenges had been dealt with.

A subset of material from the static web server content category was chosen (35,000 files) and used in the development of the prototype. This subset seemed to reveal the most important issues when transforming such data into ARC files for subsequent indexing with the open source WayBack for validation of the transformation quality.

Transforming the test material into ARC-files

Since the Heritrix web crawler maintained by the Internet Archive is developed in Java and includes an ARC format reader and writer it seemed most suitable to build the transformation prototype in Java as well and then make use of the Heritrix ARCReader and ARCWriter. It was written as a command line program to make it easy to test.

The prototype is basically a program which, when given a set of parameters, will recursively loop in a directory structure and thereby will write all discovered files into an ARC file. For larger collections a pool of ARC writers writing into multiple files would be helpful to have but in this prototype setup we used only one ARC file. The name of the generated ARC file is the third parameter for the prototype (all parameters will be described later).

For each file found in the loop the following ARC header fields must be determined:

<URL> <IP-number> <date stamp> <mime-type> <size>

The **<URL>** is partly given by the filename and the directory structure in which the file is located. This can be transformed into: /directory_X/directory_Y/..../filename. The missing part is the part in front of the top-level directory at which the loop starts. This part could be merely a hostname (e.g. www.myhost.com) or it could be a hostname and a number of directories (e.g. www.myhost.com/news/...../).

So, an essential thing to do before looping a tree of directories is to find the URL-prefix to add to the directory names and file names. If this metadata has not been delivered by the provider of the files it might be difficult to find. In the test data we found that material for some links had used absolute links (complete URLs) pointing to some of the objects in the test material and in this way we could determine the correct URL-prefix. For the prototype we chose that the material which should be transformed had to be in a directory structure holding the whole URL-prefix – e.g.:

/home/development/test/www.dr.dk/news/..... (For material originating from www.dr.dk/news/...)

The first parameter for the prototype was a pointer for the directory to start at – in the example:

/home/development/test/

It could be argued that a separate argument for the URL-prefix would be useful – and this would be very simple to implement. In that case the material would not have to be moved into a directory structure holding hostname plus possible directory names but the original structure (e.g. from the delivered CD-ROM) could be maintained and just call the program with the URL-prefix.

When converting directory names and file names into URLs, strings have to be URL-encoded (except the slashes between directory names). A file name can (in most file systems) contain special characters, like whitespace which is not a valid character in an URL. Situations could be imagined in which an URL could give misleading / non-working results due to encoding problems but for the test material a simple URL-encoding seemed to work for the majority of the cases.

When crawling live the **<IP-number>** is of course taken from the host answering the request from the crawler. When working with delivered material you might not know what IP-number originally hosted the material (this could potentially be multiple IP-numbers for very big websites with load balancing mechanisms).

We did not have the original IP-numbers for the test material. So, we had to choose one of the following strategies:

- either use the current IP-number for the same hostname (which did not seem historically correct),

- or use a dummy IP-number to let the transformed data ”know” that it was not collected from the live Internet.

We chose the last strategy and decided to use 0.0.0.0 (localhost) as the IP-number since this would indicate, also in the future, that the material was not archived from a live host on the Internet. The IP-number is the second parameter for the prototype.

The **<date stamp>** is the third metadata field in an ARC header. Determining the date stamp raises a number of interesting questions including where to get the timestamp from – there are several possible solutions to this:

- from the file timestamp on the delivered file, or
- from metadata delivered with the whole collection of files (e.g. written date on the CD-ROM).

Choosing one or the other solution will affect the future indexing and browsing since you would prefer the registration of the objects to be as accurate as possible. When using the timestamp of the original file different timestamps for different files are assured – in accordance to the real world in which files are most likely not edited at the same time. This is probably the most accurate timestamp to use – but on the other hand there is no guarantee that it is actually the right timestamp, and that the file was last edited at exactly this time. The file could have been copied several times from one medium to another adding new timestamps to the files.

If the producer of the delivery of web material forgot to preserve the original timestamps of the files, all files in the delivered copy will probably have almost the same timestamp – in which case it might be advantageous to choose the other strategy and pick a point of time and then connect all objects in the delivery to this specific moment.

As for the test material it seemed that files and directories had very different timestamps which indicated that these were most likely the originally preserved ones. So we decided to use these for registrating the objects in the ARC files. For other samples a parameter for the transformation tool could probably be useful to allow for specification of whether to use the original timestamp of the files or to use a generic global timestamp specified as yet another argument to the program.

Like the timestamp the **<mimetype>** for harvested material is taken directly from the web server response. For delivered material there is no web server response so the mimetype has to be decided upon. In order to do this a library suitable for determining a mimetype for a certain file is needed.

For the prototype we decided to test JMimeMagic[6] which is a Java open source library. The outcome was quite good and JMimeMagic only had to give up on around 12% of the files in the test. For these files we decided to implement a simple fall back mechanism that provided a mimetype based on the file extension alone because a significant amount of material unidentifiable by JMimeMagic was actually simple file types like XML, Adobe Photoshop and Macromedia Flash. With JMimeMagic and the fallback mechanism implemented for six different file extensions the number of files that was not identified was narrowed down to 4%. Further investigation of other tools and combination mechanisms should enable you to lower the number even more. For the objects given a mimetype we

[6] http://sourceforge.net/projects/jmimemagic/ - released under the LGPL open source license.

chose the mimetype ”text/plain” to be able to view the object character by character in a browser when testing the transformation quality afterwards.

The last ARC header field is the **<size>**. This is the only variable which can be determined with a 100% accuracy as this regards the size of the original object.

After the first test runs of the prototype we gave the resulting ARC file to the open source WayBack machine for indexing and browsing. The indexing went fine but we were not able to browse anything. A quick investigation told us, that the ARC file lacked some important information; the http response headers. In order to replay the open source tool WayBack needs at least two important header fields:

- http status code
- mimetype

The http status code[7] is a code telling the client (normally a web browser) the overall status of a request: OK (code 200), NOT Found (code 404), Not Allowed (code 401) etc.

Since we did not have a real http status code we decided to add ”HTTP/1.0 200 OK” to all objects based on the assumption that all files had once been available on a web server. This is of course not always true, but as long as there was no evidence or metadata from the web server telling us differently this is how we proceeded. We also added the mimetype earlier determined in a http header field ”Content-Type:” (e.g. ”Content-Type: text/html”).

These two lines were added at the top of all objects together with a subsequent blank line according to the ARC format standard. This did of course affect the **<size>** field of the ARC header, which consequently had to include the size of the extra added data as well.

After this improvement the content was ”perfectly” replayable in the open source WayBack machine. Browsing the material revealed some other interesting problems which will be discussed in the next section.

Indexing with WayBack

Browsing the transformed material with the WayBack engine seemed to work just fine in the beginning. Of course every file added to the ARC file was available on exactly the same URL as written in the ARC header and on some simple variants of URL canonicalization per default used by the WayBack indexer (e.g. same URL with and without 'www.').

Nevertheless some obvious problems were discovered which revealed important behaviour of a modern webcrawler.

The first thing missing in the transformation from file system structure into ARC files were default redirects. Standard web servers tend to behave with the following two default redirects.

1. URLs ending with a directory name redirects to the same URL + '/'
   e.g. http://dr.dk/news -> http://dr.dk/news/

[7] As defined in RFC2616 section 10: http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

2. URLs ending with a '/' redirects to the so-called default page
   e.g. http://dr.dk/news/ -> http://dr.dk/news/index.php

Modern webcrawlers (like Heritrix) write these redirects into the ARC files as objects with only header information. However, this means that the content is much more perfectly replayable since systems like WayBack can replay the redirects just as the live web server would do it.

A lot of links in the transformed material were pointing at URLs requiring these redirects and thus did not work since links often point to directory names (with or without tailing slash). Consequently all pages were – in theory and in practice – available from the full URL but browsing often went wrong due to the missing redirects.

A possible solution to this could re-index all default pages (which are the only real pages in the file system) ”back” to their ”original” redirects as they were on the web server. This could be done either while indexing the original files or in a separate process.

This was done in the test material by defining the name of default files, and whenever the prototype visited such a default file it would, together with the default file, write two extra dummy ARC objects into the ARC file: one which redirected from the directory name and another one which redirected from the directory name + '/'.

In order to use this method it is necessary to know the setup of the web server from which the material originates or you will have to try to guess the setup based on the files in the material to be transformed. This is often a very easy task since the same default filename will appear in many folders.

The method described dramatically improved the browsing quality since all links pointing to places inside the test material now seemed to work. It also proved why it is important that webcrawlers record everything – not only the real files.

Conclusion and future work

Building the prototype described in this paper showed us that it is certainly possible to transform simple static content into ARC files for long term storage and access. Converting file system based structures into ARC files reveals a number of problems with determining important metadata information about the original material: URL, IP-number, timestamp and mimetype.

The delivered dataset described in the beginning of the paper revealed serious problems with dynamic material requiring the original server environment to be setup to transform the material in a meaningful way. Once such a setup has been completed one could install a standard webcrawler on the web server including a simple module rewriting all URLs from the test-web server-URL-base to the original one.

The prototype has helped us realise that if at some point we are to start accepting material delivered in the same form as the sample material, we will have to decide on what standards to use and define the metadata for transforming the material in a sensible way or perhaps even define a set of metadata the should be included within the delivery since the producer is always the best source for providing more advanced information about the original server environment.