CS 751: Assignment 2

Bharath Kongara

Spring 2015

Contents

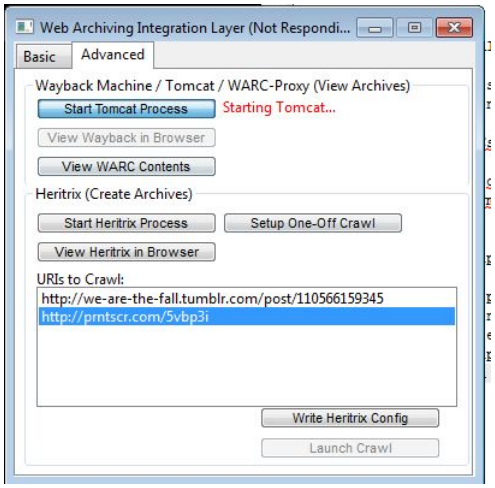

Figure 1: Generating warc files for multiple URIs using WAIL

1 Question 1

Choose 100 URIs from Assignment1 and generate WARC files of those URIs using:

- wget ,WARCreate ,Heritrix (stand-alone or via WAIL) and webrecorder.io
- Describe the resulting WARC files: quantitavely compare and contrast the results of the WARC files of the same URI as generated by different tools - choose interesting examples
- Demonstrate playback of 2-3 WARCs in the (Wayback Machine (via WAIL or stand-alone) or pywb) and (webrecorder.io) -"`https://github.com/iipc/openwayback`" -"`https://github.com/ikreymer/pywb`"

1.1 Solution

The following steps were taken to setup tools and generate WARC files:

- Installed wget using 'brew install wget'.
- WARC file for each URI is generated using the command 'wget –warc-file=outputfilename link'.
- Downloaded WARCreate chrome extension from chrome web store and added it to the chrome browser. Generated WARC for each URI manually by providing input to WARCreate.
- Installed WAIL.
- The figure below shows how to generate warc files for multiple URIs using single heritrix instance.
- When a URI had parameters then WAIL wasn't able to generate warc file.

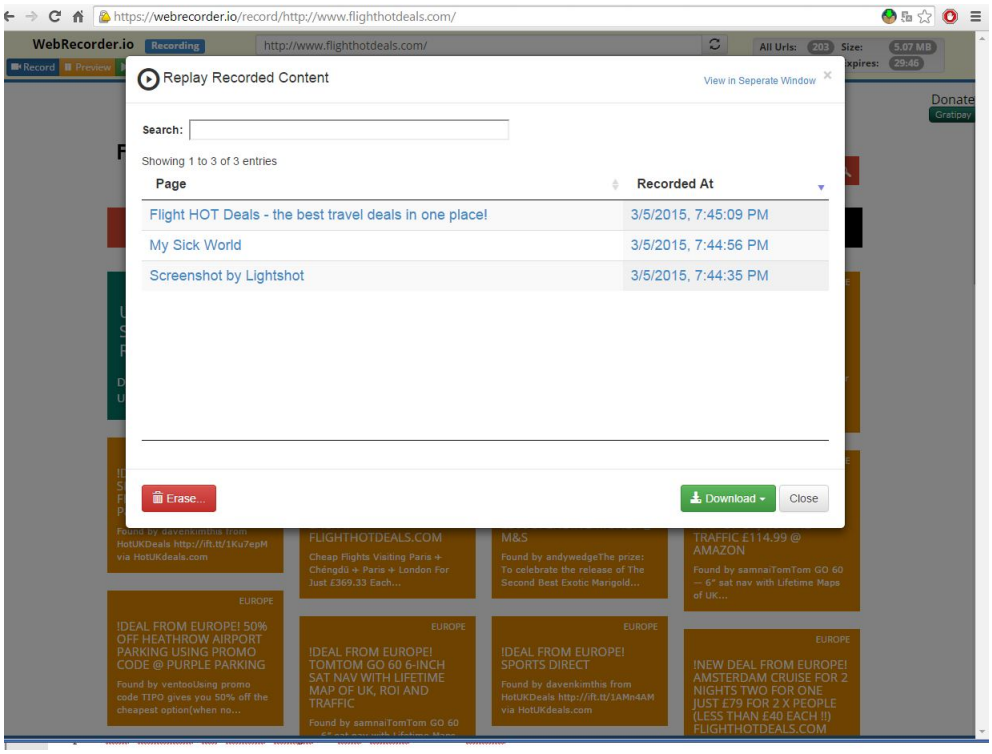

Figure 2: Generating warc files for multiple URIs using webrecorder

- Used "`https://webrecorder.io/`" . Generating warc files for multiple URIs is done as shown in the figure below.
- Quantitative comparison of WARC files generated by WAIL,webrecorder,WARCreater and wget is done on the basis of sizes.
- The comparison sizes are WAIL= 40 MB , webrecorder = 20.48 MB, WARCreate = 15.36 MB, wget = 4 MB.
- The WAIL size is more compared to others because WAIL software crawls data of the links in the website.
- This comparison is shown in the graph below.
- Installed pywb to playback warc files.
- pywb requires cdx for each warc to playback,so generated cdx file for each warc file.
- I have put all my warc files into a new folder and changed the archivepaths of config.xml to point my warc files.
- Play back for two warc files using pywb is shown in the below figure
- WebRecorder.io takes warc file to replay the archived file. Play back for two warc files using WebRecorder.io is shown in the below figure

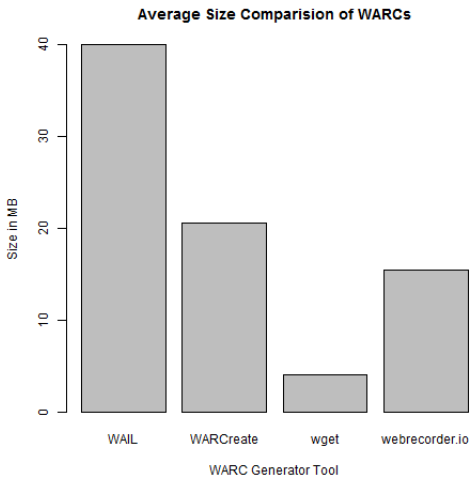

Figure 3: Average Size Comparision of WARC

Figure 4: Play back warc file using pywb

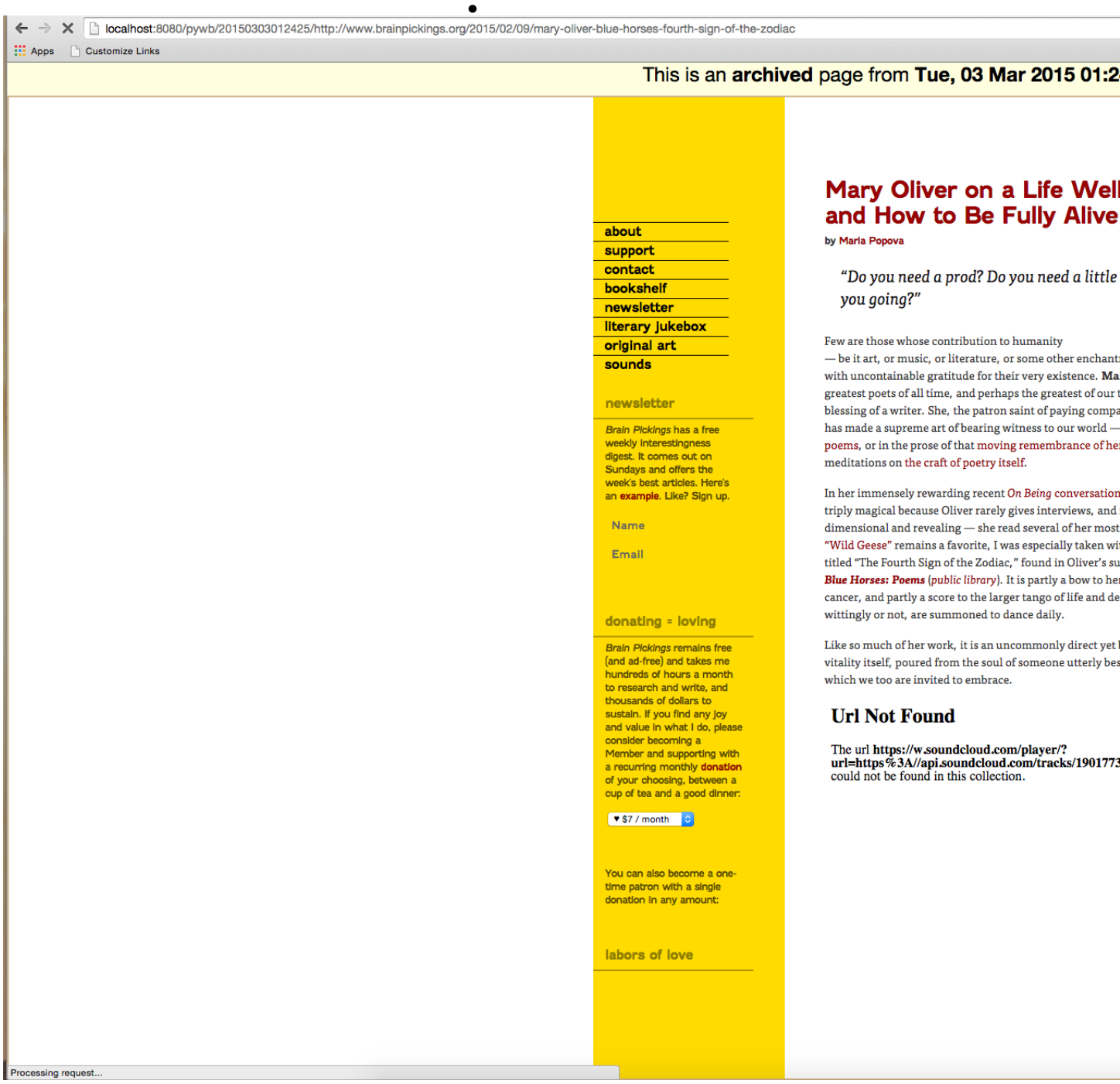

Figure 5: Play back warc file using pywb

Figure 6: Play back warc file using webrecorder

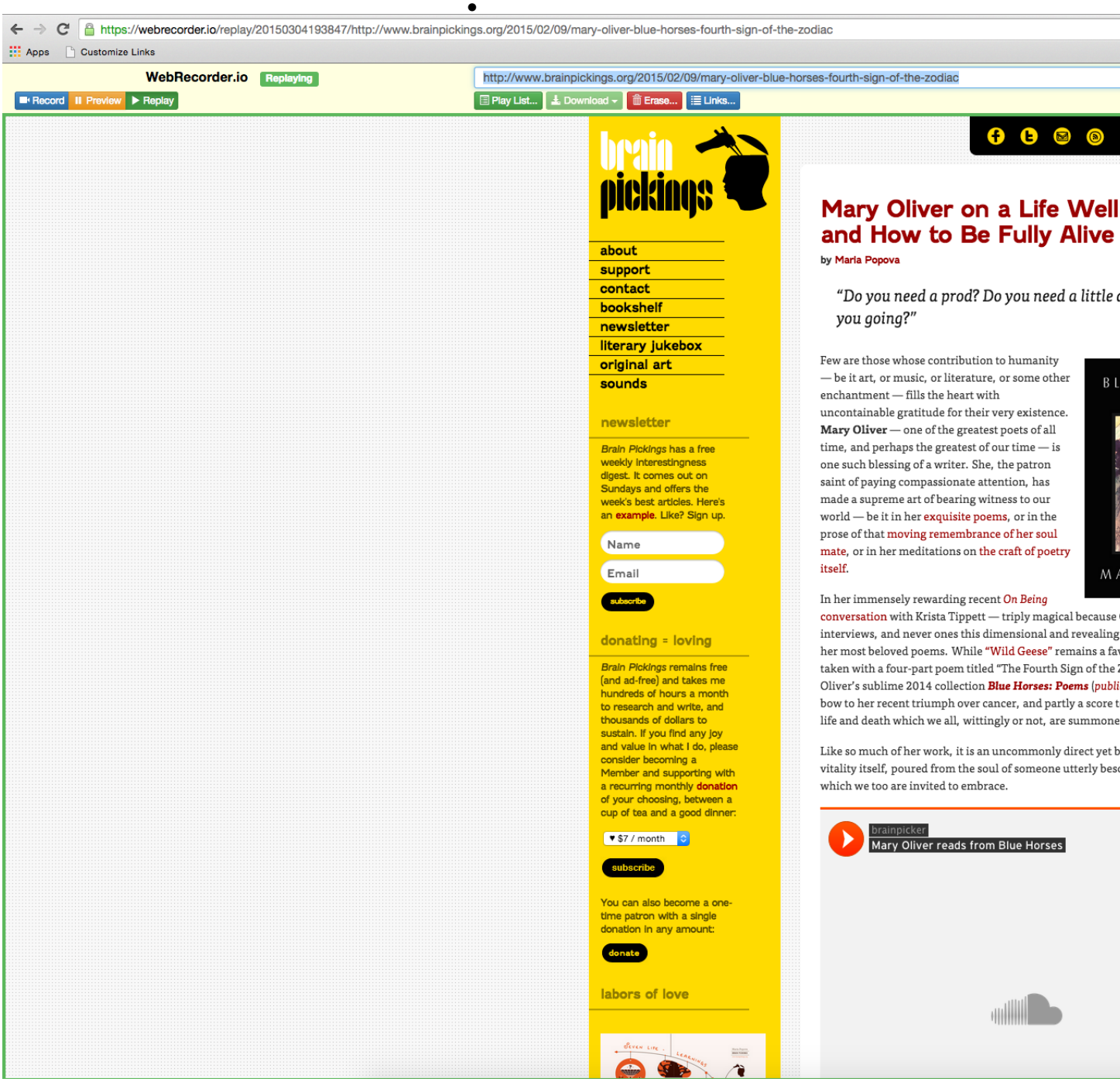

Figure 7: Play back warc file using webrecorder

2 Question 2

- Ingest the 100 URIs from their resulting WARC files into a SOLR instance - see the code + tutorial at: "`https://github.com/ukwa/webarchive-discovery`"
- Demonstrate several functioning queries on the files(a full front-end is not required) - describe the configuration choices you made in setting up SOLR and processing the documents

2.1 Solution

The following steps were taken to configure SOLR and process documents :

- The pre-requisites for SOLR are Maven 3 , Java 7 so I installed them.
- I faced an issue while installing SOLR which is jetty dependency not found.
- I got it working by adding the dependency to the pom.xml
- mvn jetty:run-exploded this command starts the SOLR instance.
- Indexing WARC file by using the command java -jar path of jar -s "`http://localhost:8080/discovery`" -t path of WARC
- Here's demonstarting of how to retrive the names and ids of all documents with "`http://localhost:8080/solr/select?q=inStock:false&wt=json&fl=id,name`"
- Here we are using the functional query idf(field,term).This function returns the inverse document frequency for the given term, using the similarity for the field. "`http://localhost:8080/solr/select/?fl=score,id&defType=func&q=mul(tf(text,memory),idf(text,memory))`"
- The functional query being used here is tf(field,term) and it returns the inverse document frequency factor for the given term using the similarity for the field. "`http://localhost:8080/solr/select/?fl=score,id&defType=func&q=mul(tf($f,$t),idf($f,$t))&f=text&t=memory`"
- termfreq(field,term) is the functional query that is being used and it returns the number of times the term term appers for that field in the document. "`http://localhost:8080/solr/select/?fl=score,id&q=DDR&sort=termfreq(text,memory)desc`"
- here norm(field) is the functional query that is being used. It returns the norm stored in the index, the product of the index time boost and then length normalization factor. "`http://localhost:8983/solr/select/?fl=score,id&q=DDR&sort=norm(text)asc`"