Digital Library Assignment 2

Q1)

WARC can be created using following four tools wget, WARCreate, Heritrix and webrecorder.io.

wget was the simplest to implement. It was just one line of code as below :-

```
os.system("wget '%s' --warc-file='%s' --no-warc-compression" % (line, name))
```

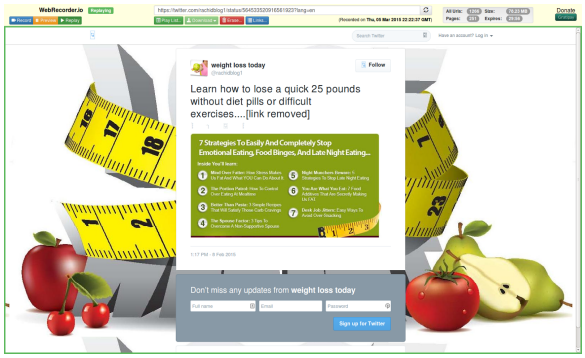
WARCreate is a chrome plugin to create a warc of current file. It was not creating WARC's for certain urls and from what I observed it was not doing it for pages that have deep links like imdb.

Heritrix warc was done using WAIL. Setting up wail took a lot of time. Wail configuration was also not straightforward. It was taking time to create WARC for certain urls as they had many sub pages. So later on a configuration was added to heritrix to restrict by time size and documents in heritrix job config file.

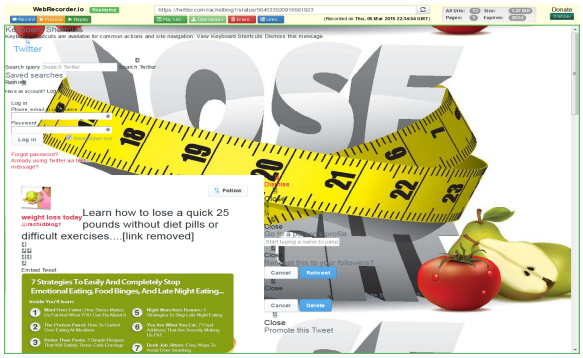
Webrecorder.io is an online tool for creating and replaying WARC file. It was simple to use. In Webrecorder.io was pretty useful in replaying WARC files created from all four methods.

Below are the screenshot for replay of warc generated by using Heritrix, WARCreate, webrecorder.io and wget in the same order. Also after every example I am adding description and comparison for warc files. All screenshot can be found in the following path (https://github.com/aag1091/cs851-s15/tree/master/assignment2/outputs)

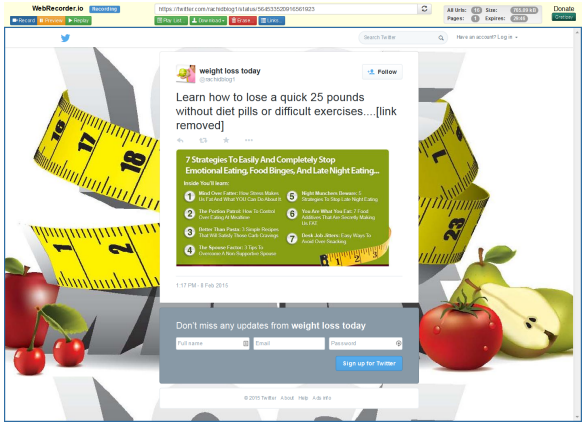
1) https://twitter.com/rachidblog1/status/564533520916561923



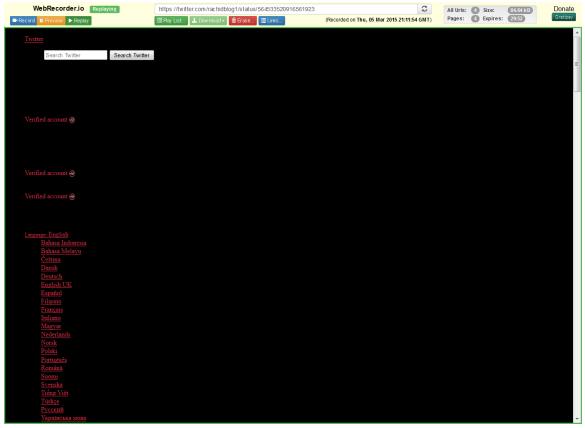
Heritrix



WARCreate

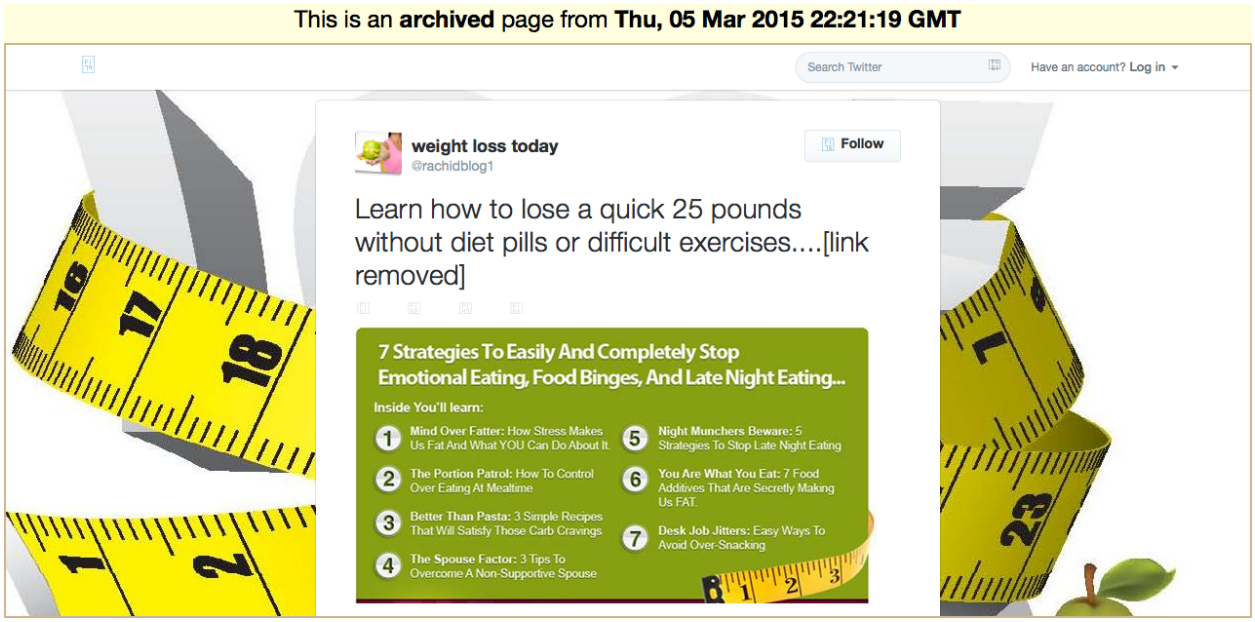


Webrecorder.io



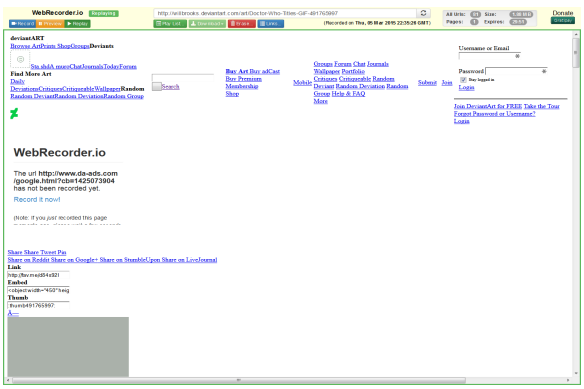
wget

All above screenshots are from webrecorder.io and below is one screenshot using pywb

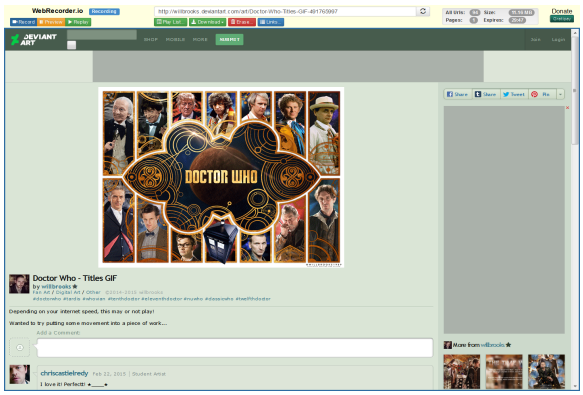


2) http://willbrooks.deviantart.com/art/Doctor-Who-Titles-GIF-491765997

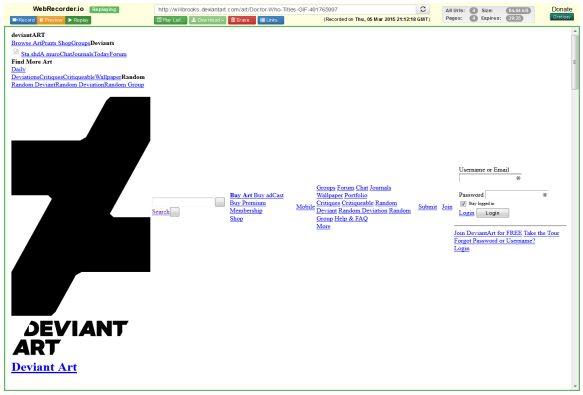
Heritrix



WARCreate

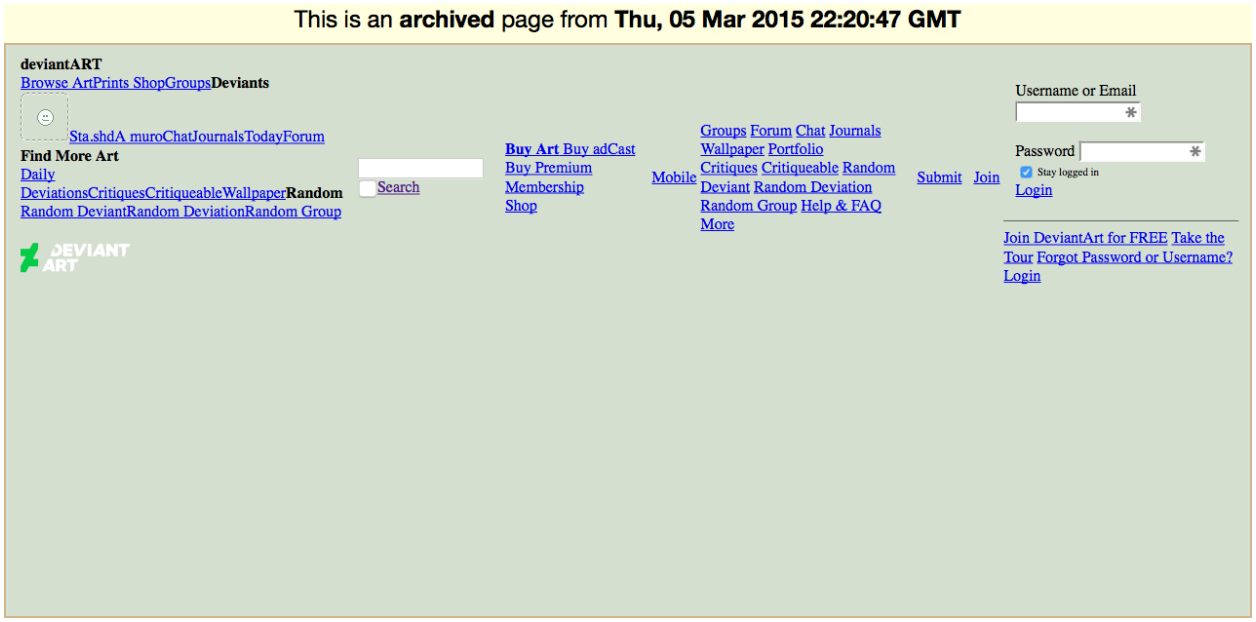


Webrecorder.io



wget

All above screenshots are from webrecorder.io and below is one screenshot using pywb

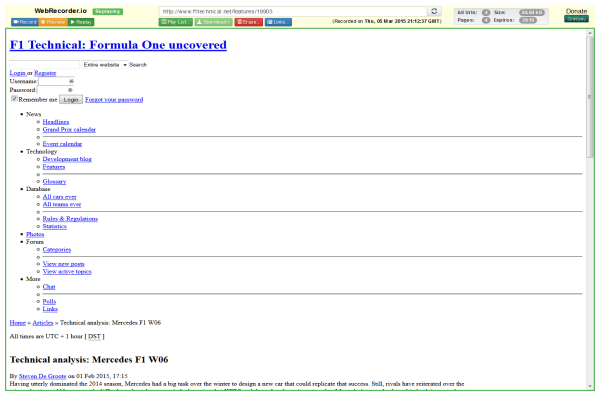


3) http://www.f1technical.net/features/19903

Heritrix

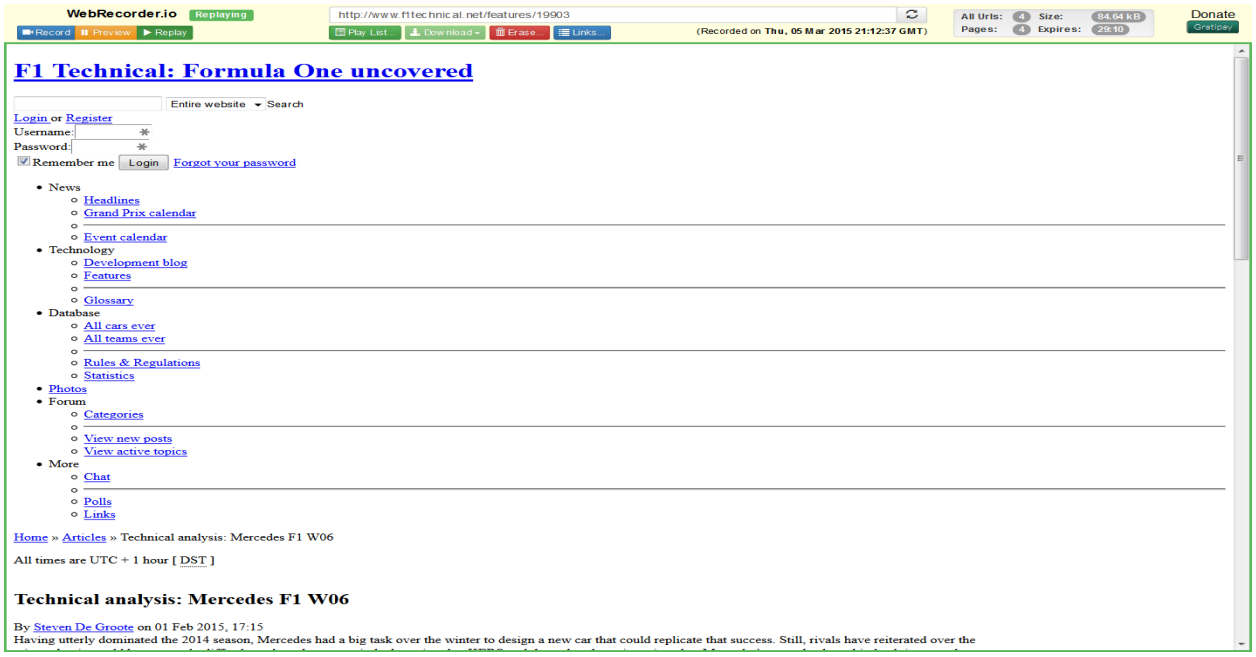
WARCreate

Webrecorder.io

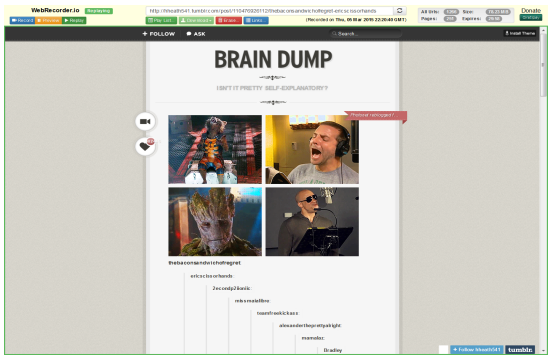


wget

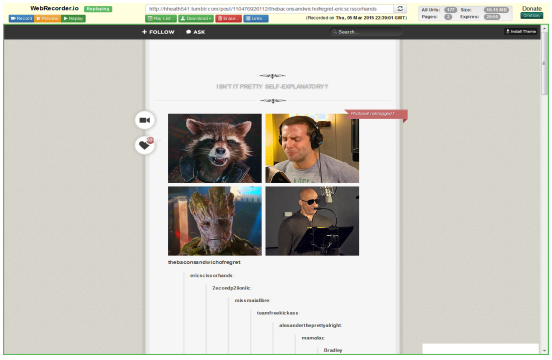
All above screenshots are from webrecorder.io and below is one screenshot using pywb



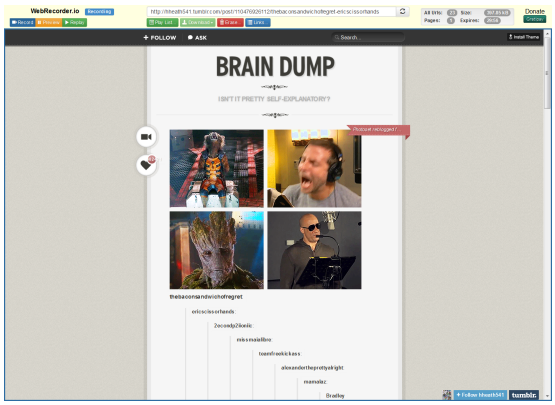
4)
http://hheath541.tumblr.com/post/110476926112/thebaconsandwichofregret-ericscissorhands



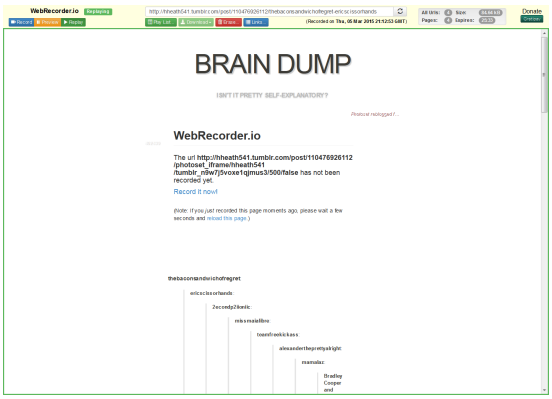
Heritrix



WARCreate

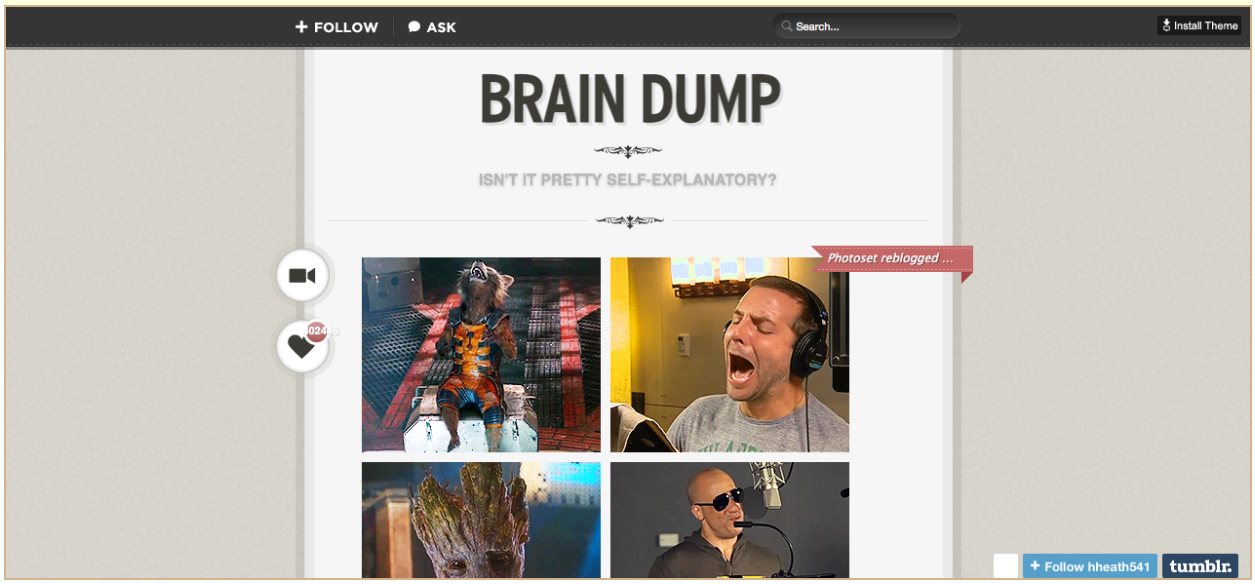


Webrecorder.io

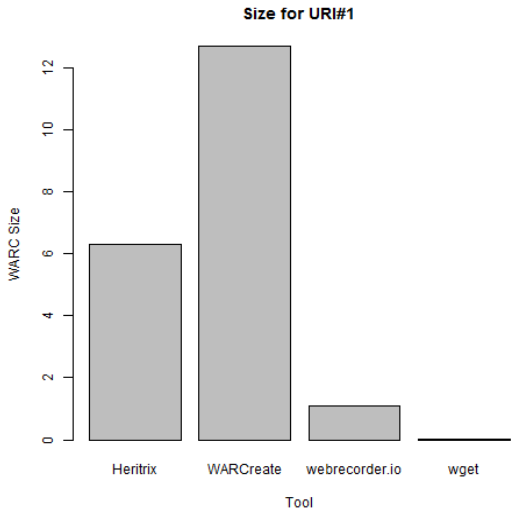


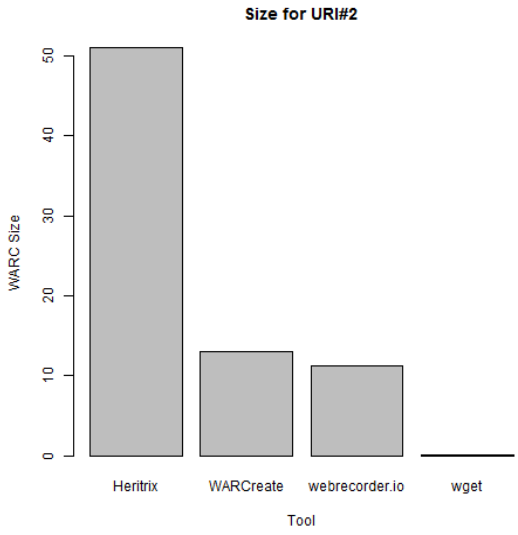
wget

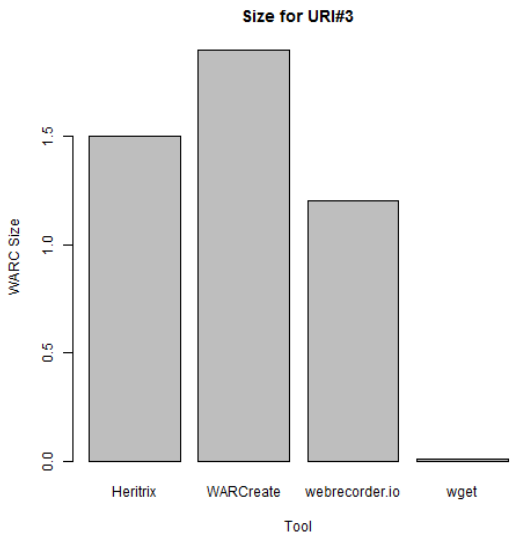
All above screenshots are from webrecorder.io and below is one screenshot using pywb

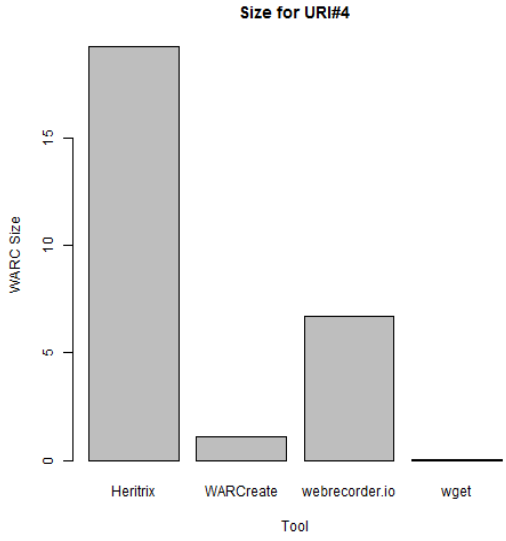


Here are the graphs comparing sizes for warc files created by different tools.









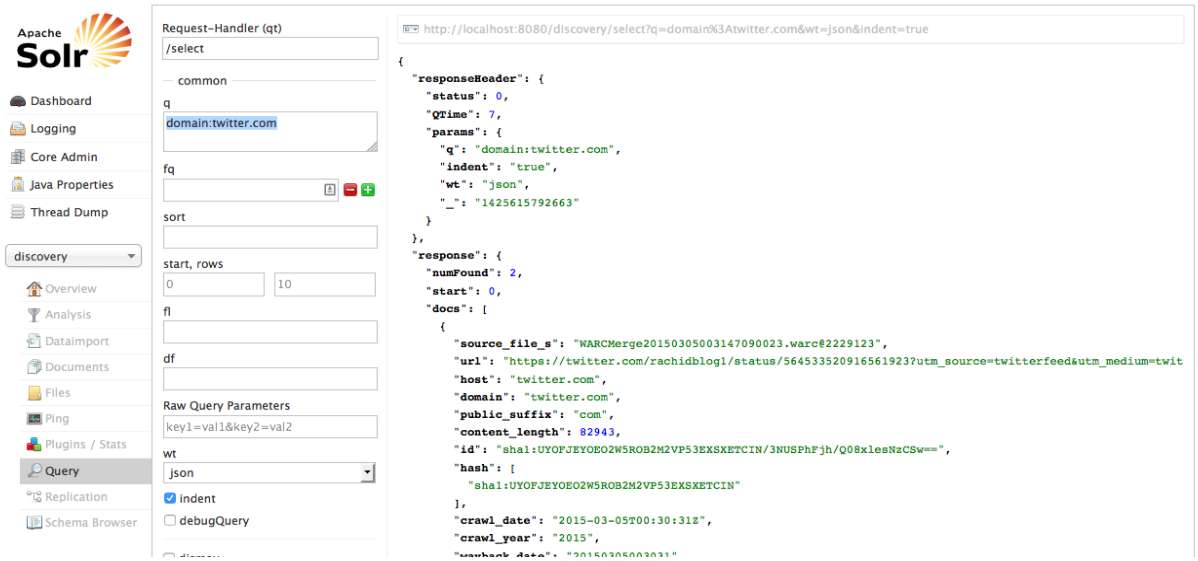
The sizes of warc files do depend on the method that we use to generate the warc file. Wget has the least size for all four URI. Heritrix would always have bigger size because it always crawls into sub-pages for a URI. The size of WARC also depends on url we are trying to call. If it has more media and deep pages it will be of bigger size.

Q2)

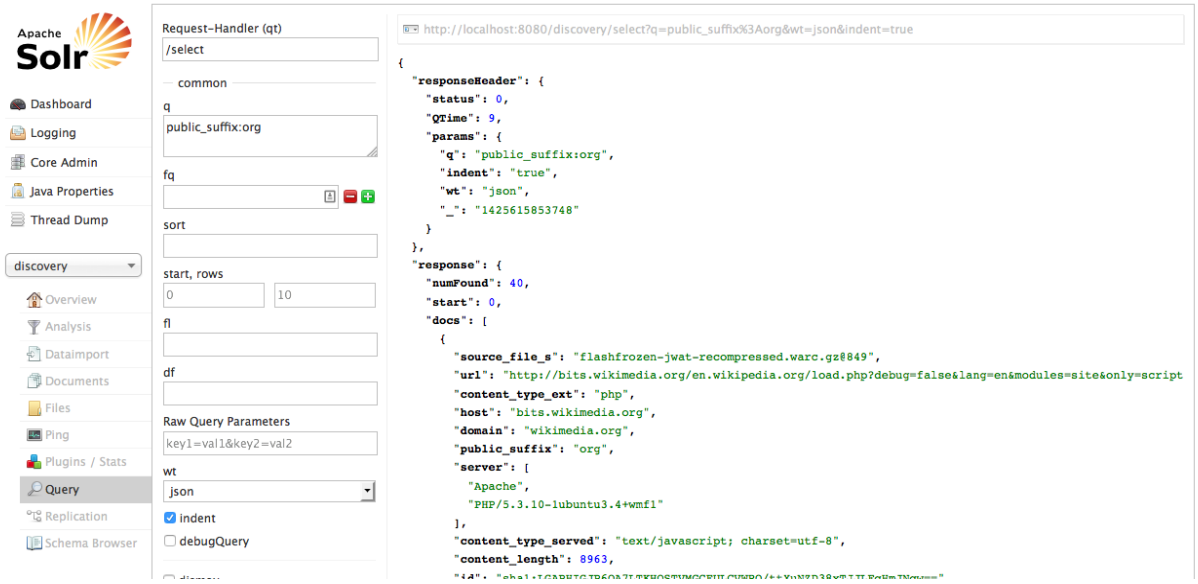
Solr has configurations to specify things like what we can index from warc files and what shouldn't be indexed. We can specify if images are to be extracted or not.

For running queries I kept the default configuration given in webarchive-discovery.

The below screenshot shows a query to return urls for domain twitter. This would give links from twitter.



The below screenshot shows a query to return urls that org as public suffix. This would give me all org domain urls.



For processing documents in solr I executed following command.

```
java -jar warc-indexer-2.0.1-20150116.110435-2-jar-with-dependencies.jar  -s http://localhost:8080/discovery -t /Users/avinashgosavi/projects/cs851-s15/assignment2/wget-combined-warc
```