**Assignment Three**

Jacob Berlin

CS432 – Spring 2016

1. Download the 1000 URIs from assignment #2. "curl", "wget", or "lynx" are all good candidate programs to use. We want just the raw HTML, not the images, stylesheets, etc. from the command line:

% curl http://www.cnn.com/ > www.cnn.com

% wget -O www.cnn.com http://www.cnn.com/

% lynx -source http://www.cnn.com/ > www.cnn.com

"www.cnn.com" is just an example output file name, keep in mind that the shell will not like some of the characters that can occur in URIs (e.g., "?", "&"). You might want to hash the URIs, like:

% echo -n "http://www.cs.odu.edu/show\_features.shtml?72" | md5

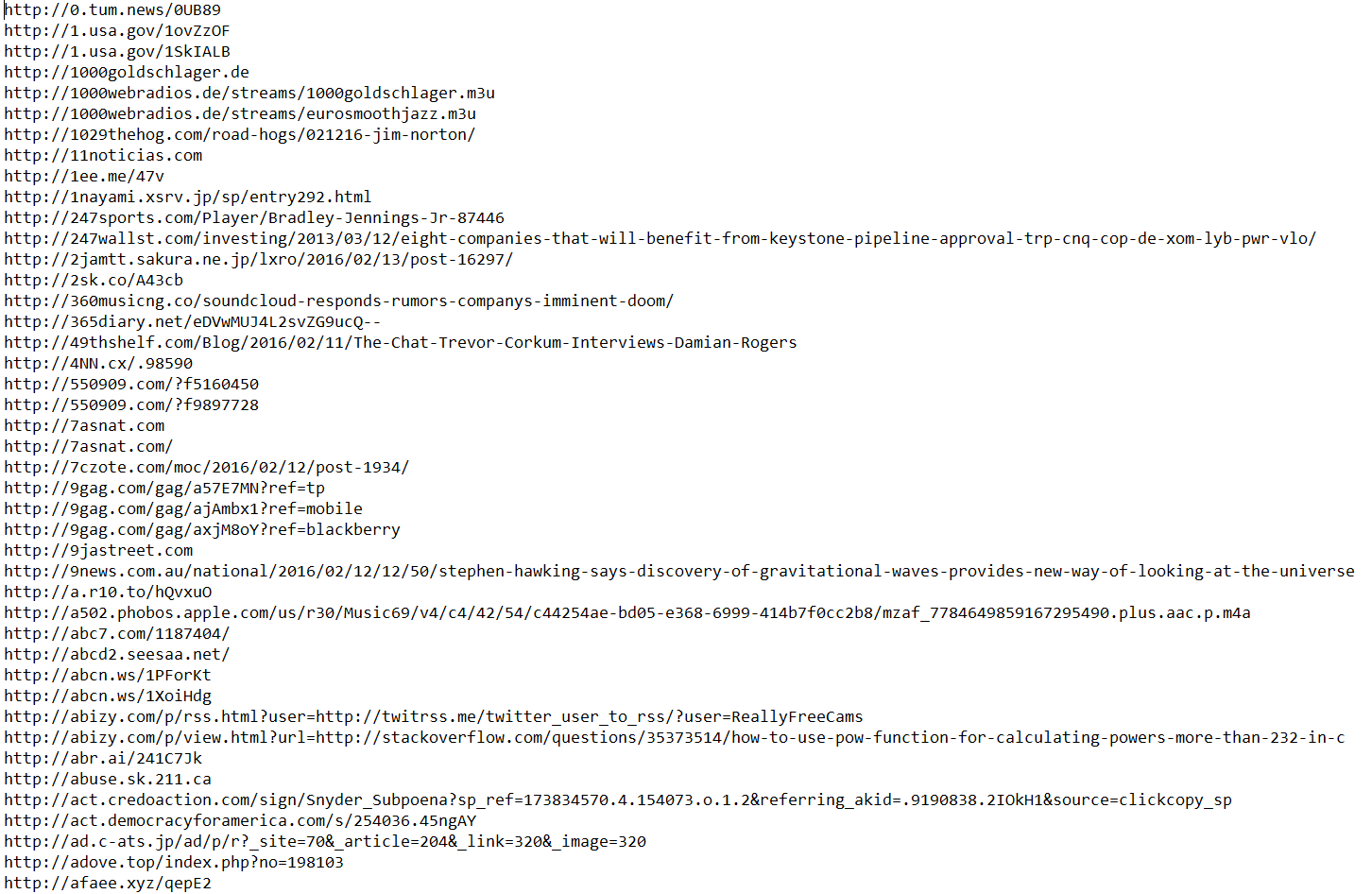
41d5f125d13b4bb554e6e31b6b591eeb

("md5sum" on some machines; note the "-n" in echo -- this removes the trailing newline.)

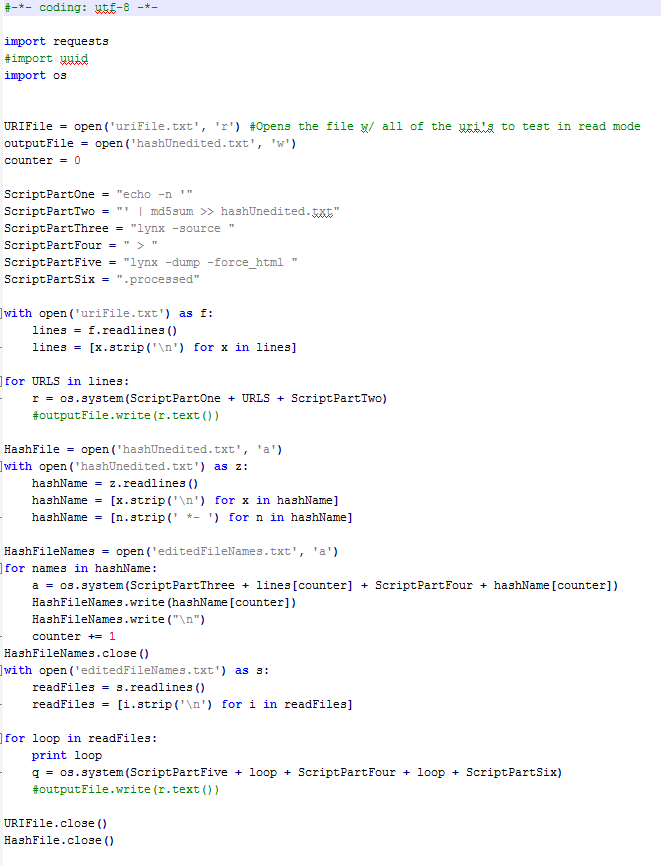
Now use a tool to remove (most) of the HTML markup. "lynx" will do a fair job:

% lynx -dump -force\_html www.cnn.com > www.cnn.com.processed

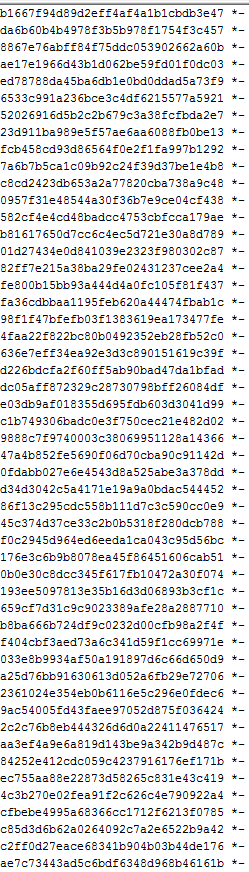
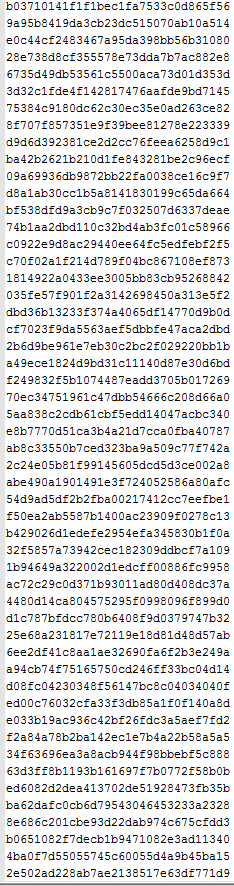
Use another (better) tool if you know of one. Keep both files for each URI (i.e., raw HTML and processed).



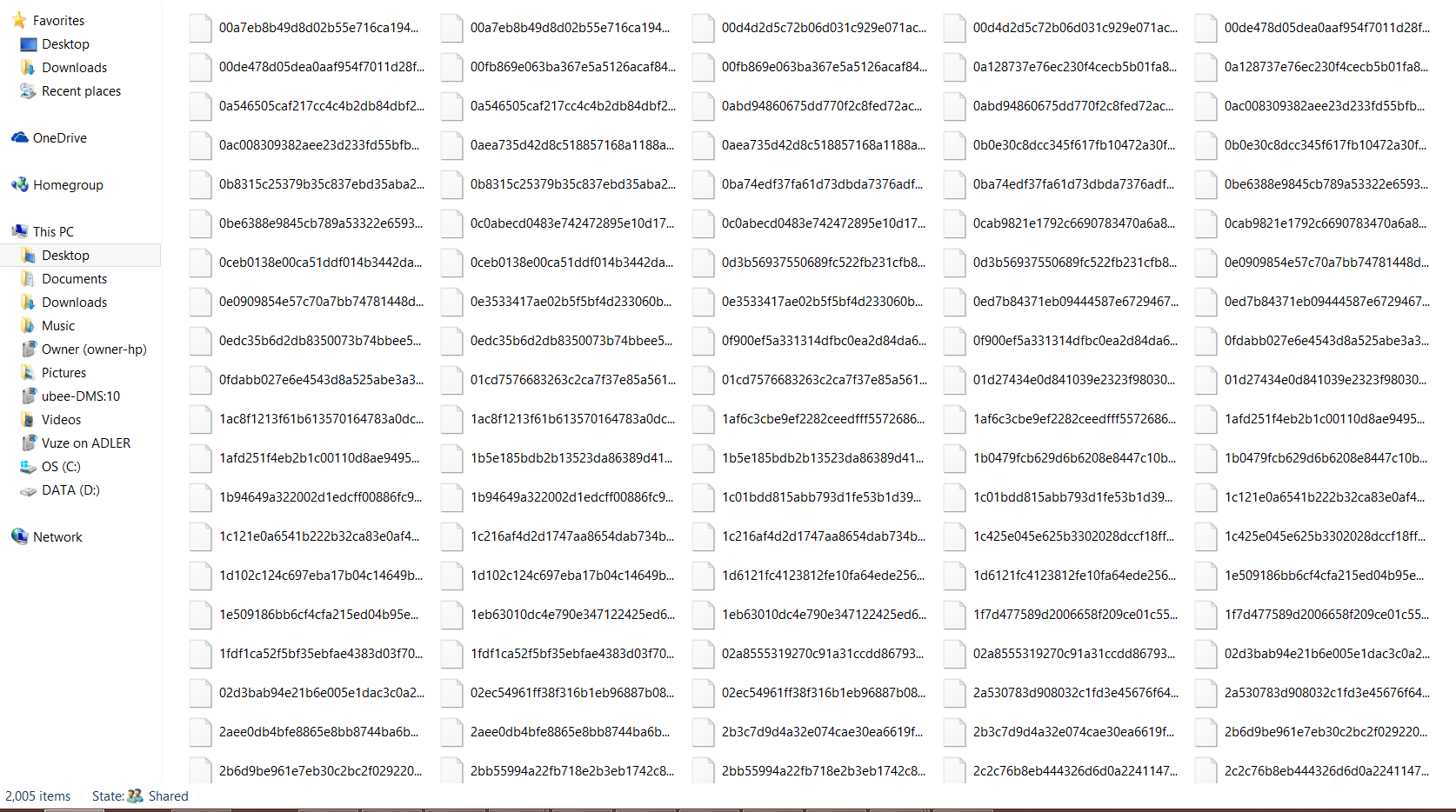
For the first question I took the 1000 different URI’s that I had acquired from question one and put them in a new file named ‘uriFile.txt’ With that information I created a new python program called ‘htmlExtractor.py’ which covered all of the steps to extract the processed files.



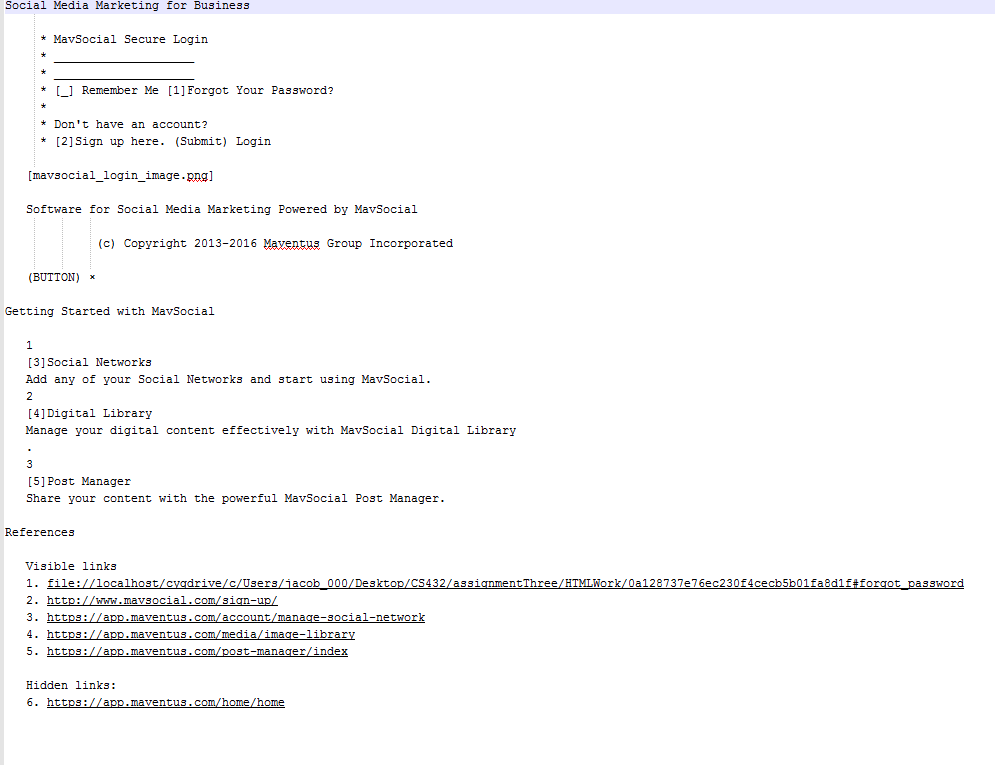
Using the md5sum commands I was able to traverse through my URI file and create a hash value for each to be used as a naming convention. This was all stored initially inside of a file named ‘hashUnedited’ due to md5sum creating a trailing ‘\*-’ after each has value.



After obtaining the initial unedited hash values, I put all of the original URIs through the ‘lynx –source foo.bar > hashValue’ meanwhile trimming off the ends of the hash values. This gave me both the list of edited hash values and the 1000 files of pure html named after the hash codes under the filename ‘editedFileNames.txt’.



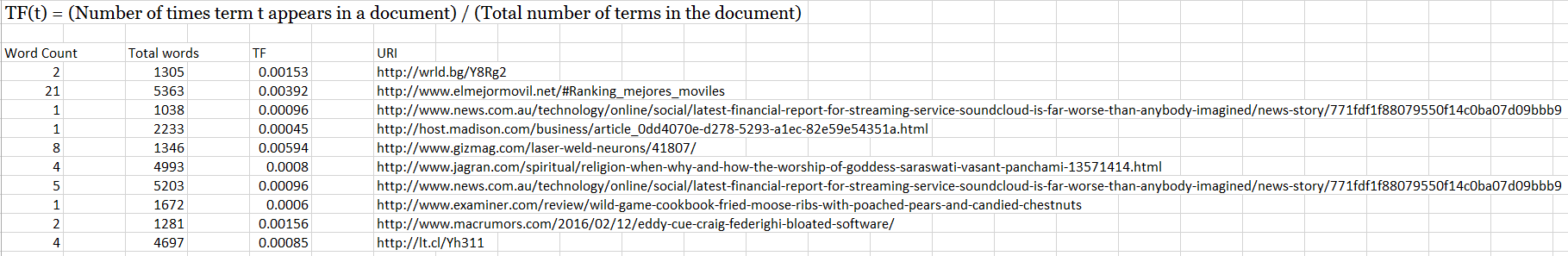
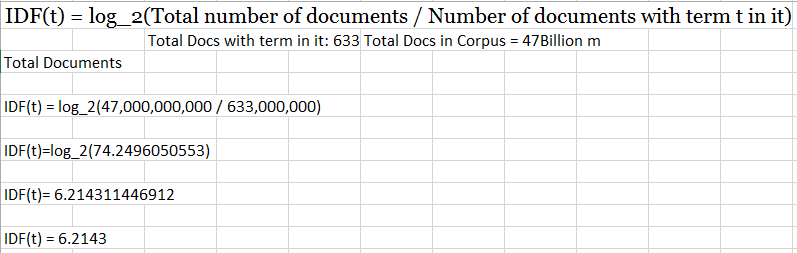
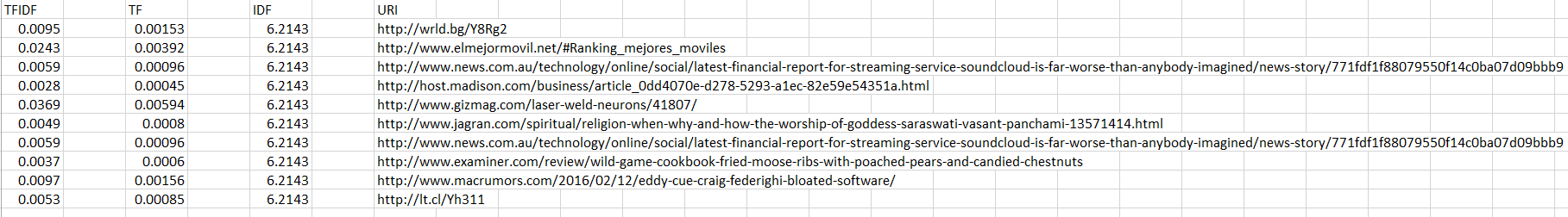
After that, I ran each generated file through the ‘lynx –dump’ command to get the 1000 processed files without the HTML structure thus completing question 1.



2. Choose a query term (e.g., "shadow") that is not a stop word (see week 5 slides) and not HTML markup from step 1 (e.g., "http") that matches at least 10 documents (hint: use "grep" on the processed files). If the term is present in more than 10 documents, choose any 10 from your list. (If you do not end up with a list of 10URIs, you've done something wrong).

As per the example in the week 5 slides, compute TFIDF values for the term in each of the 10 documents and create a table with the TF, IDF, and TFIDF values, as well as the corresponding URIs. The URIs will be ranked in decreasing order by TFIDF values.

All of my work can be shown in the ‘calculations.xlsx’ file. My keyword was ‘smartphone’.



3. Now rank the same 10 URIs from question #2, but this time by their PageRank. Use any of the free PR estimators on the web.

