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
```

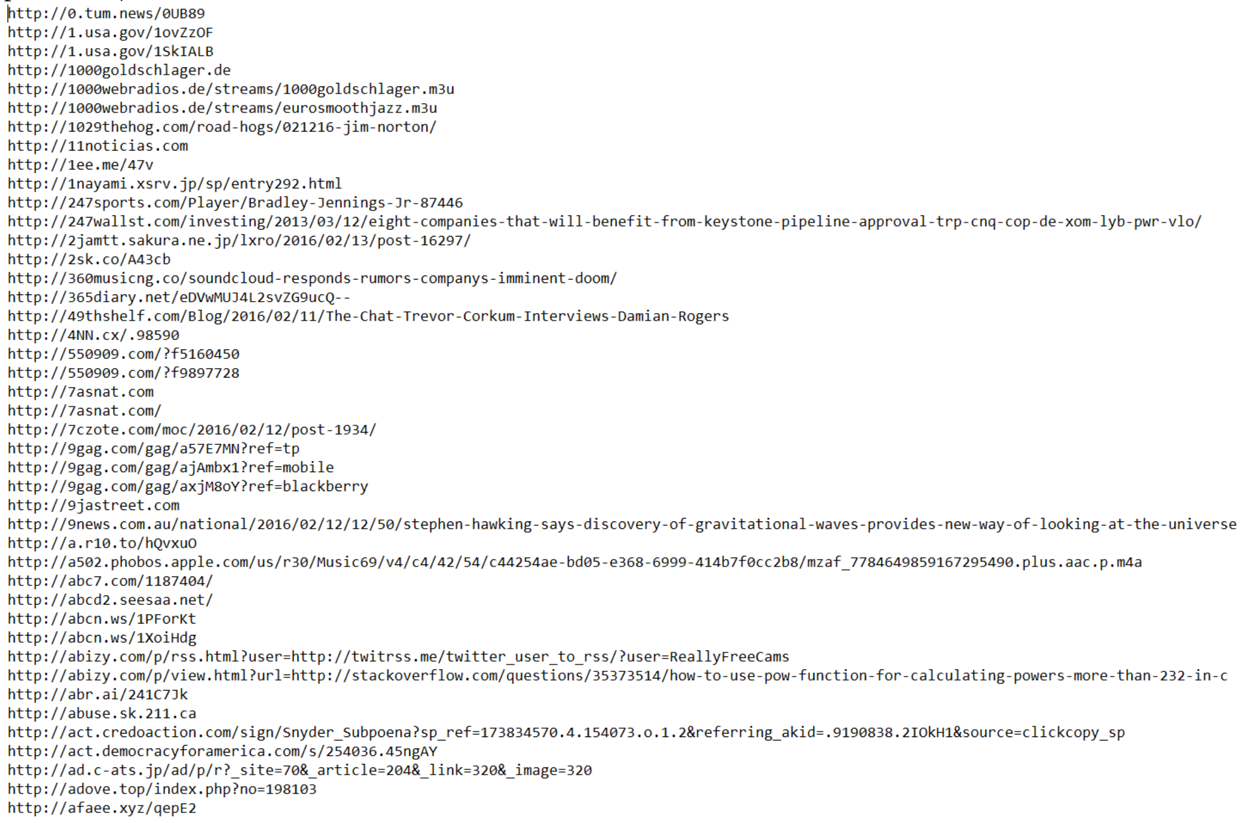
```
41d5f125d13b4bb554e6e31b6b591eeb
```

("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:

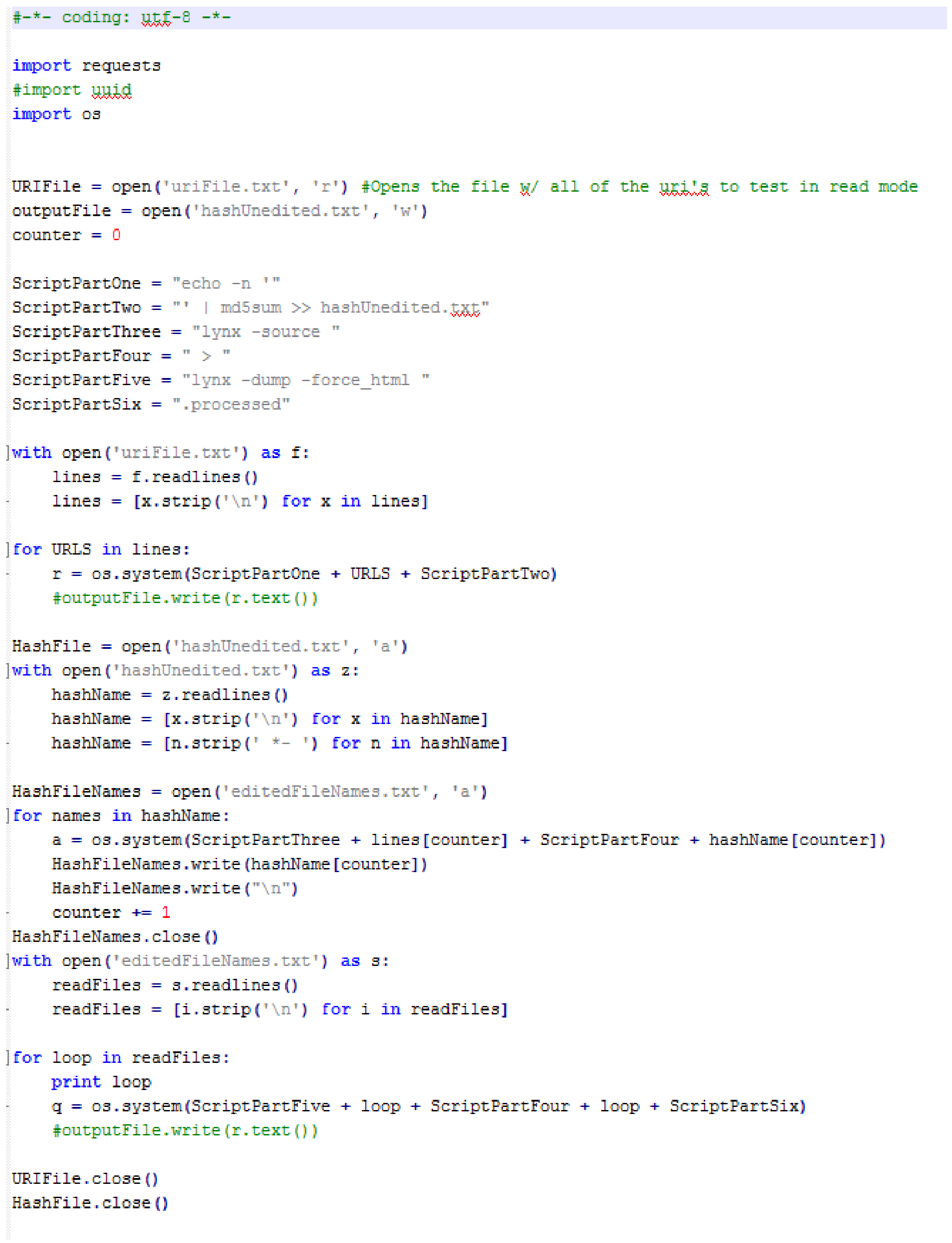
```
% lynx -dump -force_html www.cnn.com > www.cnn.com.processed
```

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



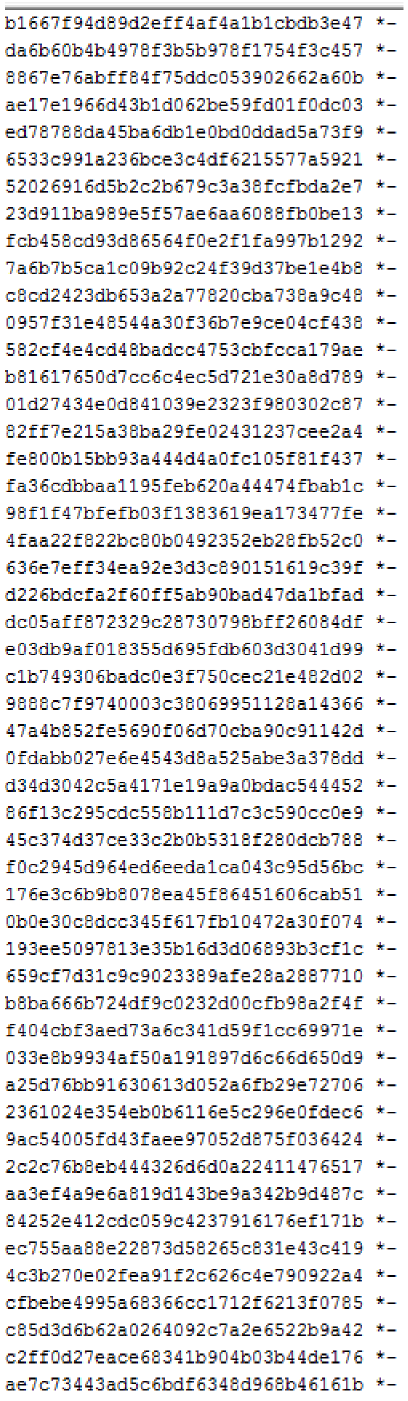
```
http://0.tum.news/0UB89
http://1.usa.gov/1ovZzOF
http://1.usa.gov/1SkIALB
http://1000goldschlager.de
http://1000webradios.de/streams/1000goldschlager.m3u
http://1000webradios.de/streams/eurosmoothjazz.m3u
http://1029thehog.com/road-hogs/021216-jim-norton/
http://11noticias.com
http://1ee.me/47v
http://1nayami.xsrv.jp/sp/entry292.html
http://247sports.com/Player/Bradley-Jennings-Jr-87446
http://247wallst.com/investing/2013/03/12/eight-companies-that-will-benefit-from-keystone-pipeline-approval-trp-cnq-cop-de-xom-lyb-pwr-vlo/
http://2jamtt.sakura.ne.jp/lxro/2016/02/13/post-16297/
http://2sk.co/A43cb
http://360musicng.co/soundcloud-responds-rumors-companys-imminent-doom/
http://365diary.net/eDVwMUJ4L2svZG9ucQ--
http://49thshelf.com/Blog/2016/02/11/The-Chat-Trevor-Corkum-Interviews-Damian-Rogers
http://4NN.cx/.98590
http://550909.com/?f5160450
http://550909.com/?f9897728
http://7asnat.com
http://7asnat.com/
http://7czote.com/moc/2016/02/12/post-1934/
http://9gag.com/gag/a57E7MN?ref=tp
http://9gag.com/gag/ajAmbx1?ref=mobile
http://9gag.com/gag/axjM8oY?ref=blackberry
http://9jastreet.com
http://9news.com.au/national/2016/02/12/12/50/stephen-hawking-says-discovery-of-gravitational-waves-provides-new-way-of-looking-at-the-universe
http://a.r10.to/hQvxuO
http://a502.phobos.apple.com/us/r30/Music69/v4/c4/42/54/c44254ae-bd05-e368-6999-414b7f0cc2b8/mzaf_7784649859167295490.plus.aac.p.m4a
http://abc7.com/1187404/
http://abcd2.seesaa.net/
http://abcn.ws/1PForKt
http://abcn.ws/1XoiHdg
http://abizy.com/p/rss.html?user=http://twitrss.me/twitter_user_to_rss/?user=ReallyFreeCams
http://abizy.com/p/view.html?url=http://stackoverflow.com/questions/35373514/how-to-use-pow-function-for-calculating-powers-more-than-232-in-c
http://abr.ai/241C7Jk
http://abuse.sk.211.ca
http://act.credoaction.com/sign/Snyder_Subpoena?sp_ref=173834570.4.154073.o.1.2&referring_akid=.9190838.2IOkH1&source=clickcopy_sp
http://act.democracyforamerica.com/s/254036.45ngAY
http://ad.c-ats.jp/ad/p/r?_site=70&_article=204&_link=320&_image=320
http://adove.top/index.php?no=198103
http://afaee.xyz/qepE2
```

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.

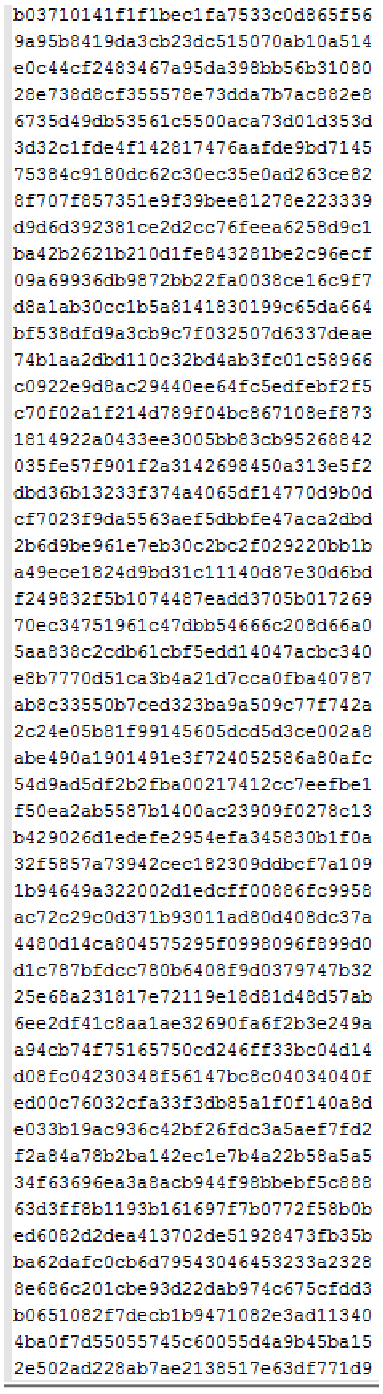


```
#-*- coding: utf-8 -*-

import requests
#import uuid
import os


URIFile = open('uriFile.txt', 'r') #Opens the file w/ all of the uri's to test in read mode
outputFile = open('hashUnedited.txt', 'w')
counter = 0

ScriptPartOne = "echo -n '"
ScriptPartTwo = "' | md5sum >> hashUnedited.txt"
ScriptPartThree = "lynx -source "
ScriptPartFour = " > "
ScriptPartFive = "lynx -dump -force_html "
ScriptPartSix = ".processed"

with open('uriFile.txt') as f:
    lines = f.readlines()
    lines = [x.strip('\n') for x in lines]

for URLS in lines:
    r = os.system(ScriptPartOne + URLS + ScriptPartTwo)
    #outputFile.write(r.text())

HashFile = open('hashUnedited.txt', 'a')
with open('hashUnedited.txt') as z:
    hashName = z.readlines()
    hashName = [x.strip('\n') for x in hashName]
    hashName = [n.strip(' *- ') for n in hashName]

HashFileNames = open('editedFileNames.txt', 'a')
for names in hashName:
    a = os.system(ScriptPartThree + lines[counter] + ScriptPartFour + hashName[counter])
    HashFileNames.write(hashName[counter])
    HashFileNames.write("\n")
    counter += 1
HashFileNames.close()
with open('editedFileNames.txt') as s:
    readFiles = s.readlines()
    readFiles = [i.strip('\n') for i in readFiles]

for loop in readFiles:
    print loop
    q = os.system(ScriptPartFive + loop + ScriptPartFour + loop + ScriptPartSix)
    #outputFile.write(r.text())

URIFile.close()
HashFile.close()
```

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.

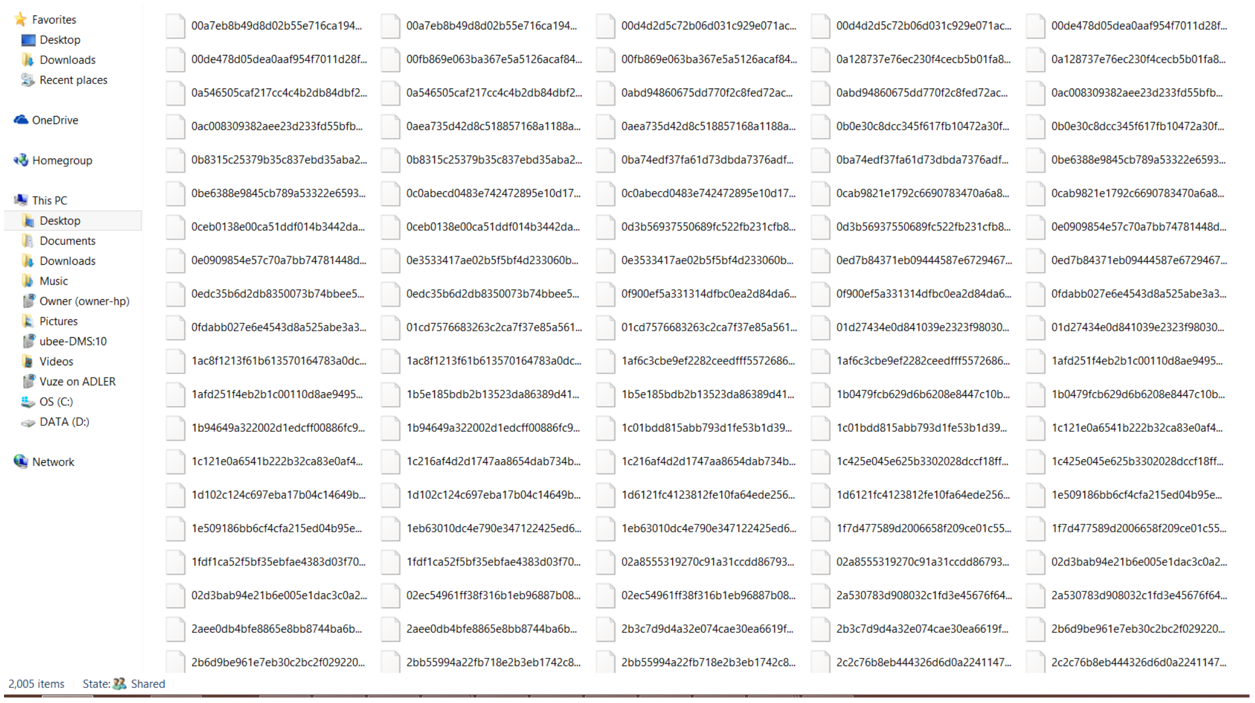


```
b1667f94d89d2eff4af4a1b1cbdb3e47 *-
da6b60b4b4978f3b5b978f1754f3c457 *-
8867e76abff84f75ddc053902662a60b *-
ae17e1966d43b1d062be59fd01f0dc03 *-
ed78788da45ba6db1e0bd0ddad5a73f9 *-
6533c991a236bce3c4df6215577a5921 *-
52026916d5b2c2b679c3a38fcfbda2e7 *-
23d911ba989e5f57ae6aa6088fb0be13 *-
fcb458cd93d86564f0e2f1fa997b1292 *-
7a6b7b5ca1c09b92c24f39d37be1e4b8 *-
c8cd2423db653a2a77820cba738a9c48 *-
0957f31e48544a30f36b7e9ce04cf438 *-
582cf4e4cd48badcc4753cbfcca179ae *-
b81617650d7cc6c4ec5d721e30a8d789 *-
01d27434e0d841039e2323f980302c87 *-
82ff7e215a38ba29fe02431237cee2a4 *-
fe800b15bb93a444d4a0fc105f81f437 *-
fa36cdbbaa1195feb620a44474fbab1c *-
98f1f47bfefb03f1383619ea173477fe *-
4faa22f822bc80b0492352eb28fb52c0 *-
636e7eff34ea92e3d3c890151619c39f *-
d226bdcfa2f60ff5ab90bad47da1bfad *-
dc05aff872329c28730798bff26084df *-
e03db9af018355d695fdb603d3041d99 *-
c1b749306badc0e3f750cec21e482d02 *-
9888c7f9740003c38069951128a14366 *-
47a4b852fe5690f06d70cba90c91142d *-
0fdabb027e6e4543d8a525abe3a378dd *-
d34d3042c5a4171e19a9a0bdac544452 *-
86f13c295cdc558b111d7c3c590cc0e9 *-
45c374d37ce33c2b0b5318f280dcb788 *-
f0c2945d964ed6eeda1ca043c95d56bc *-
176e3c6b9b8078ea45f86451606cab51 *-
0b0e30c8dcc345f617fb10472a30f074 *-
193ee5097813e35b16d3d06893b3cf1c *-
659cf7d31c9c9023389afe28a2887710 *-
b8ba666b724df9c0232d00cfb98a2f4f *-
f404cbf3aed73a6c341d59f1cc69971e *-
033e8b9934af50a191897d6c66d650d9 *-
a25d76bb91630613d052a6fb29e72706 *-
2361024e354eb0b6116e5c296e0fdec6 *-
9ac54005fd43faee97052d875f036424 *-
2c2c76b8eb444326d6d0a22411476517 *-
aa3ef4a9e6a819d143be9a342b9d487c *-
84252e412cdc059c4237916176ef171b *-
ec755aa88e22873d58265c831e43c419 *-
4c3b270e02fea91f2c626c4e790922a4 *-
cfbebe4995a68366cc1712f6213f0785 *-
c85d3d6b62a0264092c7a2e6522b9a42 *-
c2ff0d27eace68341b904b03b44de176 *-
ae7c73443ad5c6bdf6348d968b46161b *-
```



```
b03710141f1f1bec1fa7533c0d865f56
9a95b8419da3cb23dc515070ab10a514
e0c44cf2483467a95da398bb56b31080
28e738d8cf355578e73dda7b7ac882e8
6735d49db53561c5500aca73d01d353d
3d32c1fde4f142817476aafde9bd7145
75384c9180dc62c30ec35e0ad263ce82
8f707f857351e9f39bee81278e223339
d9d6d392381ce2d2cc76feea6258d9c1
ba42b2621b210d1fe843281be2c96ecf
09a69936db9872bb22fa0038ce16c9f7
d8a1ab30cc1b5a8141830199c65da664
bf538dfd9a3cb9c7f032507d6337deae
74b1aa2dbd110c32bd4ab3fc01c58966
c0922e9d8ac29440ee64fc5edfebf2f5
c70f02a1f214d789f04bc867108ef873
1814922a0433ee3005bb83cb95268842
035fe57f901f2a3142698450a313e5f2
dbd36b13233f374a4065df14770d9b0d
cf7023f9da5563aef5dbbfe47aca2dbd
2b6d9be961e7eb30c2bc2f029220bb1b
a49ece1824d9bd31c11140d87e30d6bd
f249832f5b1074487eadd3705b017269
70ec34751961c47dbb54666c208d66a0
5aa838c2cdb61cbf5edd14047acbc340
e8b7770d51ca3b4a21d7cca0fba40787
ab8c33550b7ced323ba9a509c77f742a
2c24e05b81f99145605dcd5d3ce002a8
abe490a1901491e3f724052586a80afc
54d9ad5df2b2fba00217412cc7eefbe1
f50ea2ab5587b1400ac23909f0278c13
b429026d1edefe2954efa345830b1f0a
32f5857a73942cec182309ddbcf7a109
1b94649a322002d1edcff00886fc9958
ac72c29c0d371b93011ad80d408dc37a
4480d14ca804575295f0998096f899d0
d1c787bfdcc780b6408f9d0379747b32
25e68a231817e72119e18d81d48d57ab
6ee2df41c8aa1ae32690fa6f2b3e249a
a94cb74f75165750cd246ff33bc04d14
d08fc04230348f56147bc8c04034040f
ed00c76032cfa33f3db85a1f0f140a8d
e033b19ac936c42bf26fdc3a5aef7fd2
f2a84a78b2ba142ec1e7b4a22b58a5a5
34f63696ea3a8acb944f98bbebf5c888
63d3ff8b1193b161697f7b0772f58b0b
ed6082d2dea413702de51928473fb35b
ba62dafc0cb6d79543046453233a2328
8e686c201cbe93d22dab974c675cfdd3
b0651082f7decb1b9471082e3ad11340
4ba0f7d55055745c60055d4a9b45ba15
2e502ad228ab7ae2138517e63df771d9
```

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’.



00a7eb8b49d8d02b55e716ca194...	00a7eb8b49d8d02b55e716ca194...	00d4d2d5c72b06d031c929e071ac...	00d4d2d5c72b06d031c929e071ac...	00de478d05dea0aaf954f7011d28f...
00de478d05dea0aaf954f7011d28f...	00fb869e063ba367e5a5126acaf84...	00fb869e063ba367e5a5126acaf84...	0a128737e76ec230f4cecb5b01fa8...	0a128737e76ec230f4cecb5b01fa8...
0a546505caf217cc4c4b2db84dbf2...	0a546505caf217cc4c4b2db84dbf2...	0abd94860675dd770f2c8fed72ac...	0abd94860675dd770f2c8fed72ac...	0ac008309382aee23d233fd55bfb...
0ac008309382aee23d233fd55bfb...	0aea735d42d8c518857168a1188a...	0aea735d42d8c518857168a1188a...	0b0e30c8dcc345f617fb10472a30f...	0b0e30c8dcc345f617fb10472a30f...
0b8315c25379b35c837ebd35aba2...	0b8315c25379b35c837ebd35aba2...	0ba74edf37fa61d73dbda7376adf...	0ba74edf37fa61d73dbda7376adf...	0be6388e9845cb789a53322e6593...
0be6388e9845cb789a53322e6593...	0c0abecd0483e742472895e10d17...	0c0abecd0483e742472895e10d17...	0cab9821e1792c6690783470a6a8...	0cab9821e1792c6690783470a6a8...
0ceb0138e00ca51ddf014b3442da...	0ceb0138e00ca51ddf014b3442da...	0d3b56937550689fc522fb231cfb8...	0d3b56937550689fc522fb231cfb8...	0e0909854e57c70a7bb74781448d...
0e0909854e57c70a7bb74781448d...	0e3533417ae02b5f5bf4d233060b...	0e3533417ae02b5f5bf4d233060b...	0ed7b84371eb09444587e6729467...	0ed7b84371eb09444587e6729467...
0edc35b6d2db8350073b74bbee5...	0edc35b6d2db8350073b74bbee5...	0f900ef5a331314dfbc0ea2d84da6...	0f900ef5a331314dfbc0ea2d84da6...	0fdabb027e6e4543d8a525abe3a3...
0fdabb027e6e4543d8a525abe3a3...	01cd7576683263c2ca7f37e85a561...	01cd7576683263c2ca7f37e85a561...	01d27434e0d841039e2323f98030...	01d27434e0d841039e2323f98030...
1ac8f1213f61b613570164783a0dc...	1ac8f1213f61b613570164783a0dc...	1af6c3cbe9ef2282ceedfff5572686...	1af6c3cbe9ef2282ceedfff5572686...	1afd251f4eb2b1c00110d8ae9495...
1afd251f4eb2b1c00110d8ae9495...	1b5e185bdb2b13523da86389d41...	1b5e185bdb2b13523da86389d41...	1b0479fcb629d6b6208e8447c10b...	1b0479fcb629d6b6208e8447c10b...
1b94649a322002d1edcff00886fc9...	1b94649a322002d1edcff00886fc9...	1c01bdd815abb793d1fe53b1d39...	1c01bdd815abb793d1fe53b1d39...	1c121e0a6541b222b32ca83e0af4...
1c121e0a6541b222b32ca83e0af4...	1c216af4d2d1747aa8654dab734b...	1c216af4d2d1747aa8654dab734b...	1c425e045e625b3302028dccf18ff...	1c425e045e625b3302028dccf18ff...
1d102c124c697eba17b04c14649b...	1d102c124c697eba17b04c14649b...	1d6121fc4123812fe10fa64ede256...	1d6121fc4123812fe10fa64ede256...	1e509186bb6cf4cfa215ed04b95e...
1e509186bb6cf4cfa215ed04b95e...	1eb63010dc4e790e347122425ed6...	1eb63010dc4e790e347122425ed6...	1f7d477589d2006658f209ce01c55...	1f7d477589d2006658f209ce01c55...
1fdf1ca52f5bf35ebfae4383d03f70...	1fdf1ca52f5bf35ebfae4383d03f70...	02a8555319270c91a31ccdd86793...	02a8555319270c91a31ccdd86793...	02d3bab94e21b6e005e1dac3c0a2...
02d3bab94e21b6e005e1dac3c0a2...	02ec54961ff38f316b1eb96887b08...	02ec54961ff38f316b1eb96887b08...	2a530783d908032c1fd3e45676f64...	2a530783d908032c1fd3e45676f64...
2aee0db4bfe8865e8bb8744ba6b...	2aee0db4bfe8865e8bb8744ba6b...	2b3c7d9d4a32e074cae30ea6619f...	2b3c7d9d4a32e074cae30ea6619f...	2b6d9be961e7eb30c2bc2f029220...
2b6d9be961e7eb30c2bc2f029220...	2bb55994a22fb718e2b3eb1742c8...	2bb55994a22fb718e2b3eb1742c8...	2c2c76b8eb444326d6d0a2241147...	2c2c76b8eb444326d6d0a2241147...

2,005 items   State: Shared

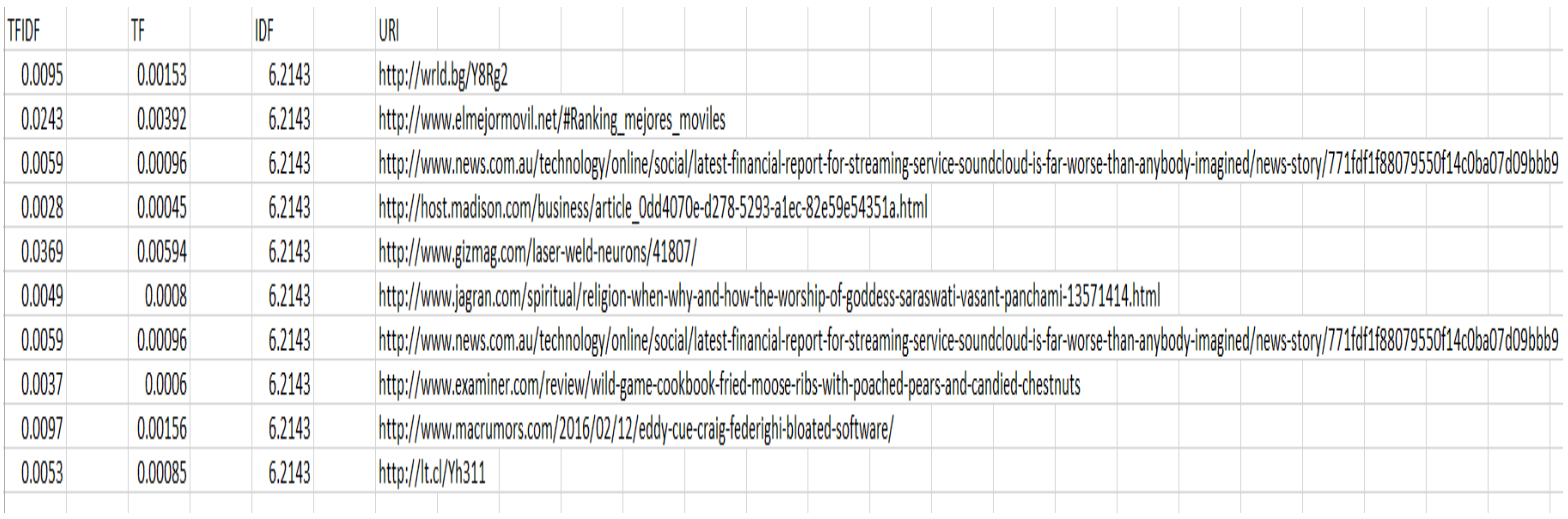
After that, I ran each generated file through the 'lynx –dump' command to get the 1000 processed files without the HTML structure thus completing question 1.

```
Social Media Marketing for Business

     * MavSocial Secure Login
     * ____________________
     * ____________________
     * [_] Remember Me [1]Forgot Your Password?
     *
     * Don't have an account?
     * [2]Sign up here. (Submit) Login

   [mavsocial_login_image.png]

   Software for Social Media Marketing Powered by MavSocial

                (c) Copyright 2013-2016 Maventus Group Incorporated

   (BUTTON) ×

Getting Started with MavSocial

   1
   [3]Social Networks
   Add any of your Social Networks and start using MavSocial.
   2
   [4]Digital Library
   Manage your digital content effectively with MavSocial Digital Library
   .
   3
   [5]Post Manager
   Share your content with the powerful MavSocial Post Manager.

References

   Visible links
   1. file://localhost/cygdrive/c/Users/jacob_000/Desktop/CS432/assignmentThree/HTMLWork/0a128737e76ec230f4cecb5b01fa8d1f#forgot_password
   2. http://www.mavsocial.com/sign-up/
   3. https://app.maventus.com/account/manage-social-network
   4. https://app.maventus.com/media/image-library
   5. https://app.maventus.com/post-manager/index

   Hidden links:
   6. https://app.maventus.com/home/home
```

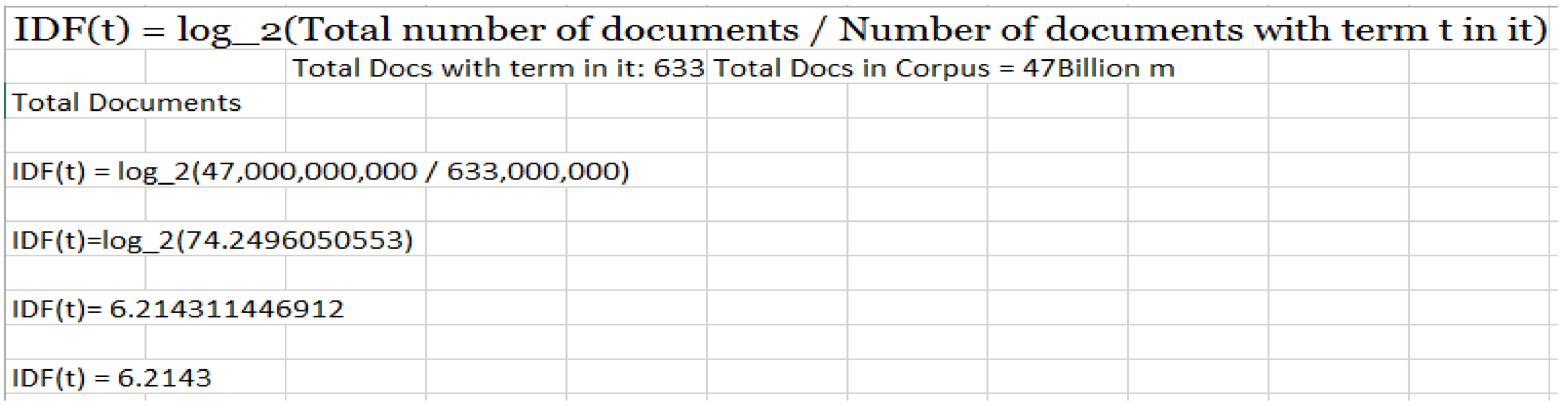
2. Choose a query term (e.g., "shadow") that is not a stop word (see week 5 slides) and not HTML markup from step 1 (e.g., "http") that matches at least 10 documents (hint: use "grep" on the processed files). If the term is present in more than 10 documents, choose any 10 from your list. (If you do not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



TFIDF	TF	IDF	URI
0.0095	0.00153	6.2143	http://wrld.bg/Y8Rg2
0.0243	0.00392	6.2143	http://www.elmejormovil.net/#Ranking_mejores_moviles
0.0059	0.00096	6.2143	http://www.news.com.au/technology/online/social/latest-financial-report-for-streaming-service-soundcloud-is-far-worse-than-anybody-imagined/news-story/771fdf1f88079550f14c0ba07d09bbb9
0.0028	0.00045	6.2143	http://host.madison.com/business/article_0dd4070e-d278-5293-a1ec-82e59e54351a.html
0.0369	0.00594	6.2143	http://www.gizmag.com/laser-weld-neurons/41807/
0.0049	0.0008	6.2143	http://www.jagran.com/spiritual/religion-when-why-and-how-the-worship-of-goddess-saraswati-vasant-panchami-13571414.html
0.0059	0.00096	6.2143	http://www.news.com.au/technology/online/social/latest-financial-report-for-streaming-service-soundcloud-is-far-worse-than-anybody-imagined/news-story/771fdf1f88079550f14c0ba07d09bbb9
0.0037	0.0006	6.2143	http://www.examiner.com/review/wild-game-cookbook-fried-moose-ribs-with-poached-pears-and-candied-chestnuts
0.0097	0.00156	6.2143	http://www.macrumors.com/2016/02/12/eddy-cue-craig-federighi-bloated-software/
0.0053	0.00085	6.2143	http://lt.cl/Yh311



IDF(t) = log_2(Total number of documents / Number of documents with term t in it)

Total Docs with term in it: 633 Total Docs in Corpus = 47Billion m

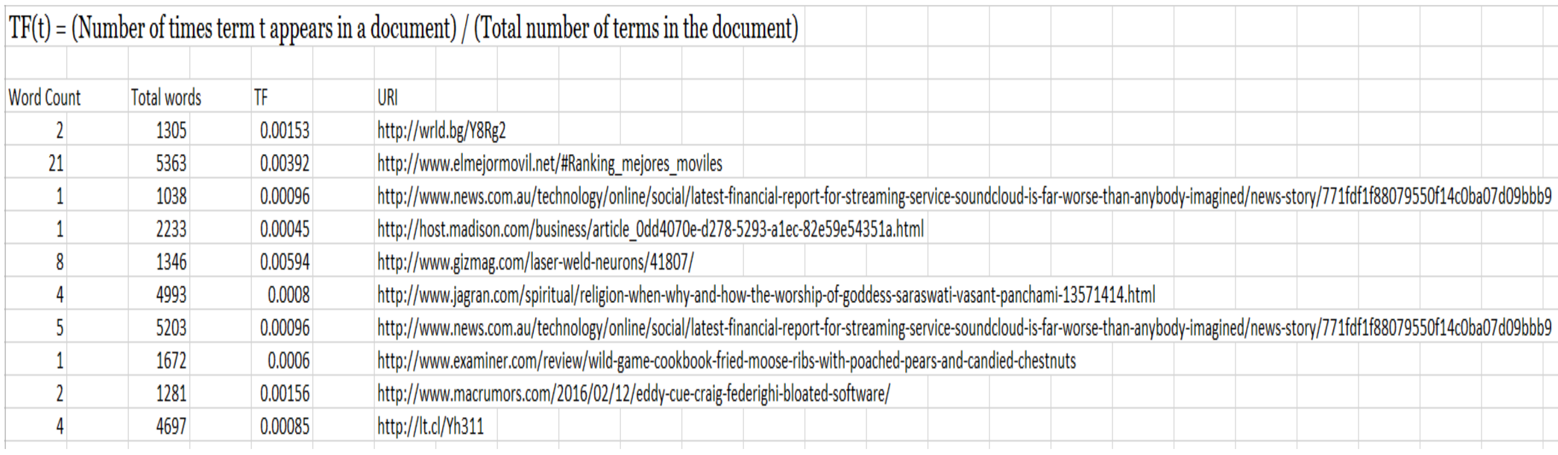
Total Documents

IDF(t) = log_2(47,000,000,000 / 633,000,000)

IDF(t)=log_2(74.2496050553)

IDF(t)= 6.214311446912

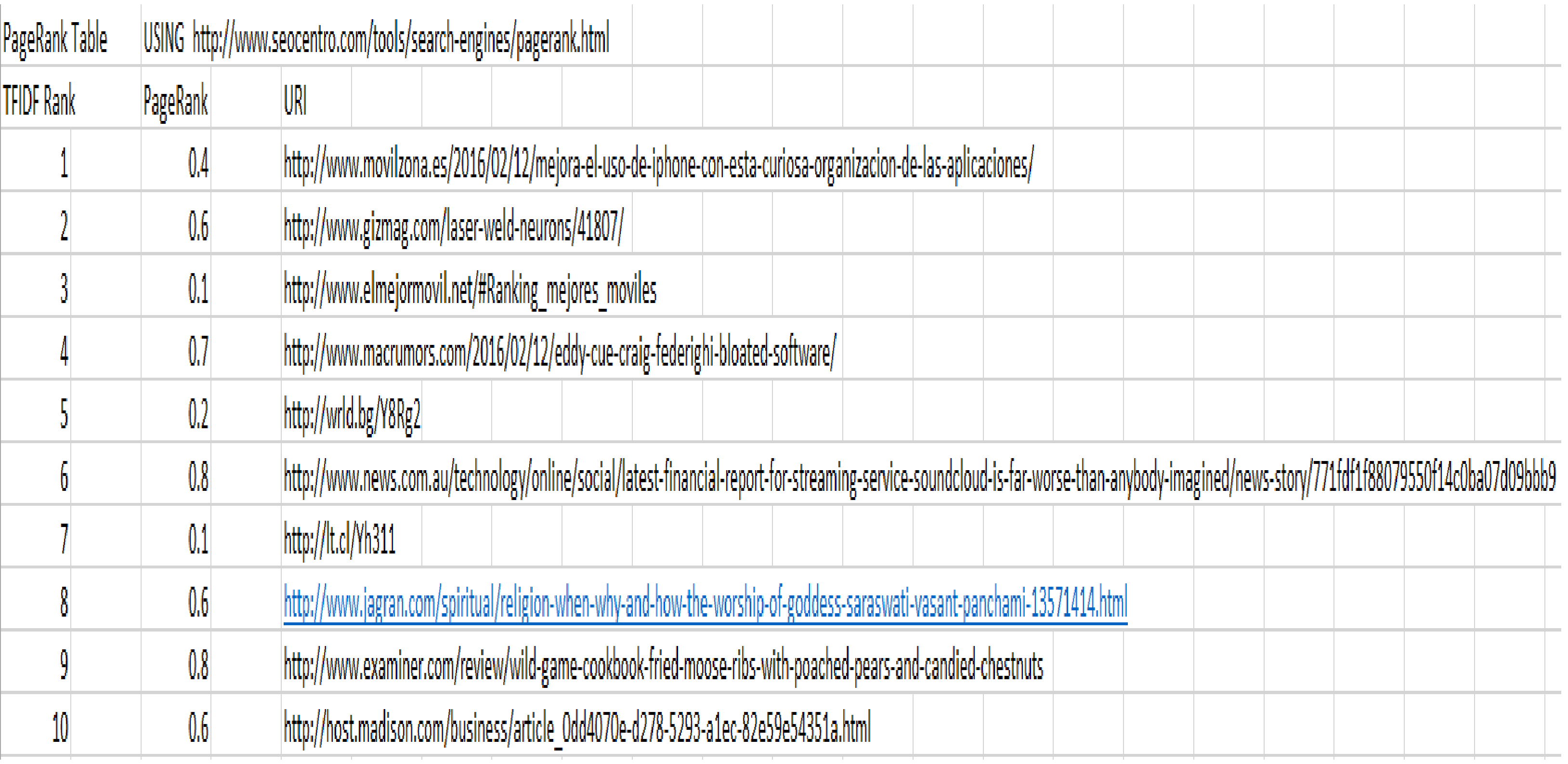
IDF(t) = 6.2143



TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document)

Word Count	Total words	TF	URI
2	1305	0.00153	http://wrld.bg/Y8Rg2
21	5363	0.00392	http://www.elmejormovil.net/#Ranking_mejores_moviles
1	1038	0.00096	http://www.news.com.au/technology/online/social/latest-financial-report-for-streaming-service-soundcloud-is-far-worse-than-anybody-imagined/news-story/771fdf1f88079550f14c0ba07d09bbb9
1	2233	0.00045	http://host.madison.com/business/article_0dd4070e-d278-5293-a1ec-82e59e54351a.html
8	1346	0.00594	http://www.gizmag.com/laser-weld-neurons/41807/
4	4993	0.0008	http://www.jagran.com/spiritual/religion-when-why-and-how-the-worship-of-goddess-saraswati-vasant-panchami-13571414.html
5	5203	0.00096	http://www.news.com.au/technology/online/social/latest-financial-report-for-streaming-service-soundcloud-is-far-worse-than-anybody-imagined/news-story/771fdf1f88079550f14c0ba07d09bbb9
1	1672	0.0006	http://www.examiner.com/review/wild-game-cookbook-fried-moose-ribs-with-poached-pears-and-candied-chestnuts
2	1281	0.00156	http://www.macrumors.com/2016/02/12/eddy-cue-craig-federighi-bloated-software/
4	4697	0.00085	http://lt.cl/Yh311

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.



PageRank Table	USING http://www.seocentro.com/tools/search-engines/pagerank.html	
TFIDF Rank	PageRank	URI
1	0.4	http://www.movilzona.es/2016/02/12/mejora-el-uso-de-iphone-con-esta-curiosa-organizacion-de-las-aplicaciones/
2	0.6	http://www.gizmag.com/laser-weld-neurons/41807/
3	0.1	http://www.elmejormovil.net/#Ranking_mejores_moviles
4	0.7	http://www.macrumors.com/2016/02/12/eddy-cue-craig-federighi-bloated-software/
5	0.2	http://wrld.bg/Y8Rg2
6	0.8	http://www.news.com.au/technology/online/social/latest-financial-report-for-streaming-service-soundcloud-is-far-worse-than-anybody-imagined/news-story/771fdf1f88079550f14c0ba07d09bbb9
7	0.1	http://lt.cl/Yh311
8	0.6	http://www.jagran.com/spiritual/religion-when-why-and-how-the-worship-of-goddess-saraswati-vasant-panchami-13571414.html
9	0.8	http://www.examiner.com/review/wild-game-cookbook-fried-moose-ribs-with-poached-pears-and-candied-chestnuts
10	0.6	http://host.madison.com/business/article_0dd4070e-d278-5293-a1ec-82e59e54351a.html