**Assignment Nine**

Jacob Berlin

CS432 – Spring 2016

1. 1. Choose a blog or a newsfeed (or something similar with an Atom or RSS feed). Every student should do a unique feed, so please "claim" the feed on the class email list (first come, first served). It should be on a topic or topics of which you are qualified to provide classification training data. Find something with at least 100 entries (or items if RSS).

Create between four and eight different categories for the entries in the feed:

examples:

work, class, family, news, deals

liberal, conservative, moderate, libertarian

sports, local, financial, national, international, entertainment

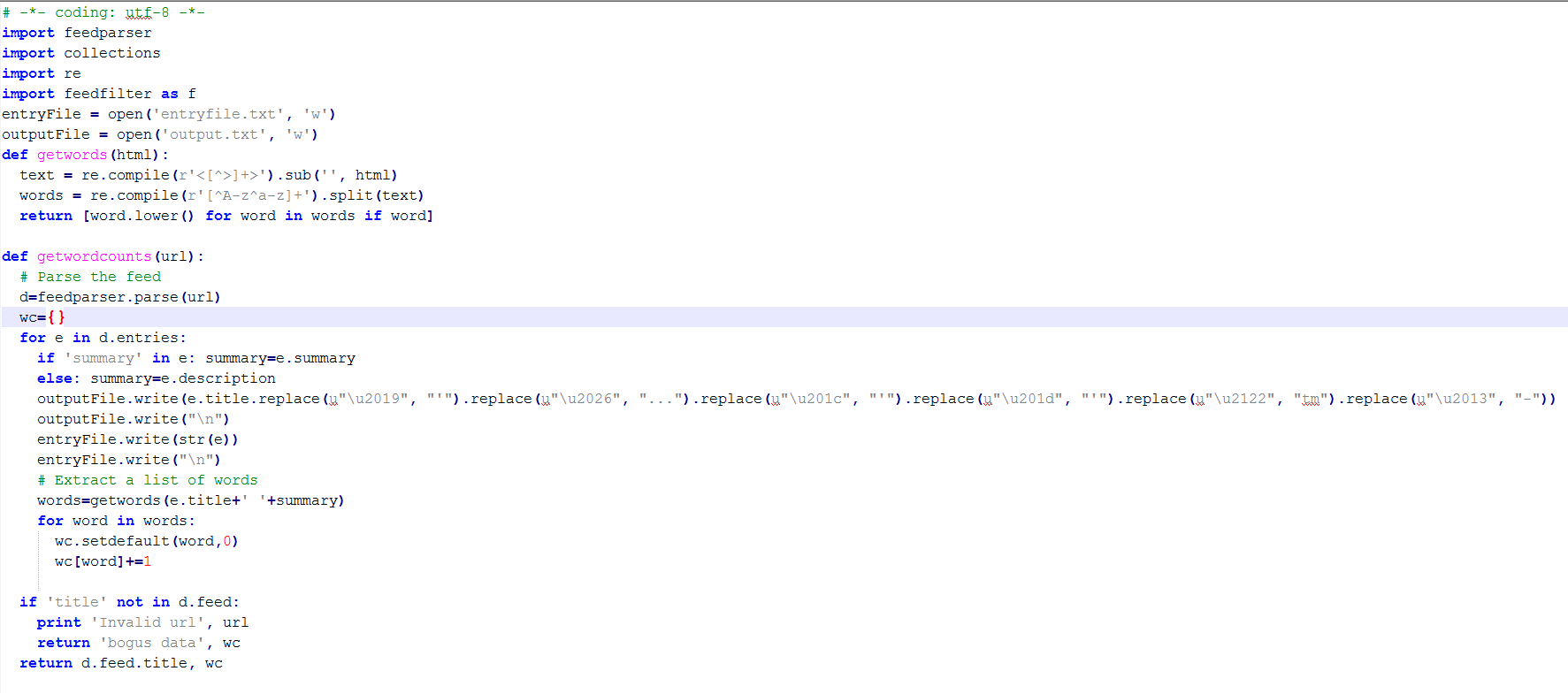
metal, electronic, ambient, folk, hip-hop, pop

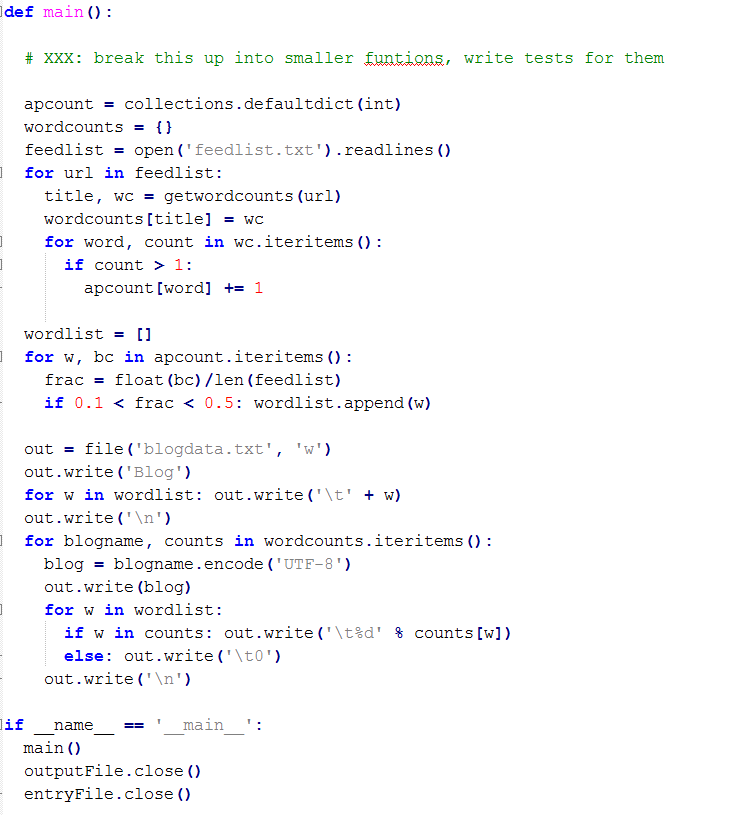
Download and process the pages of the feed as per the week 12 class slides.

Be sure to upload the raw data (Atom or RSS) to your github account.

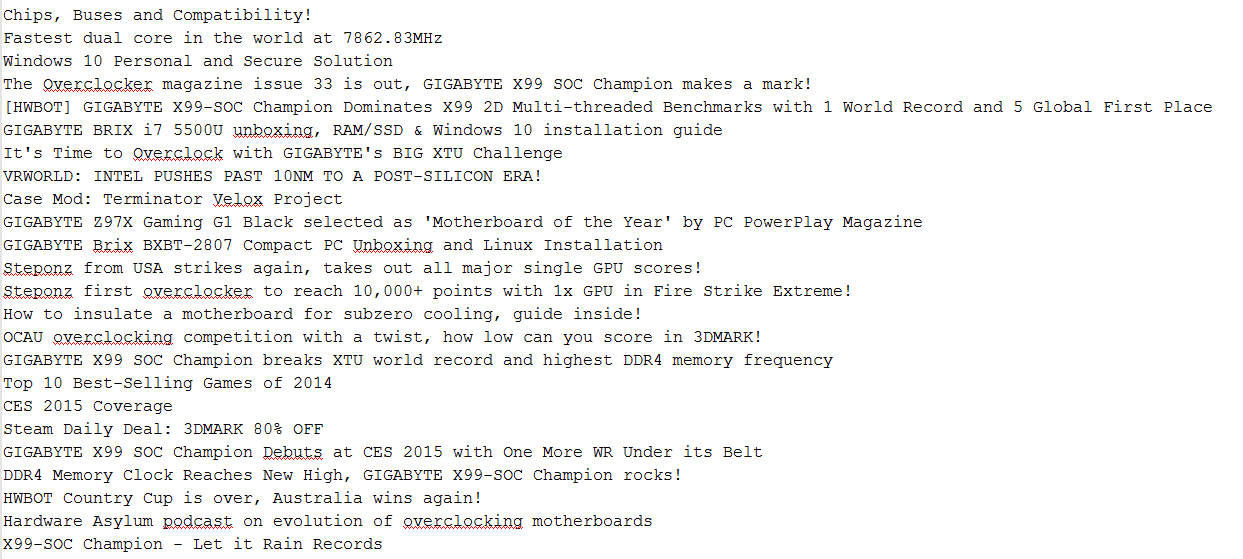
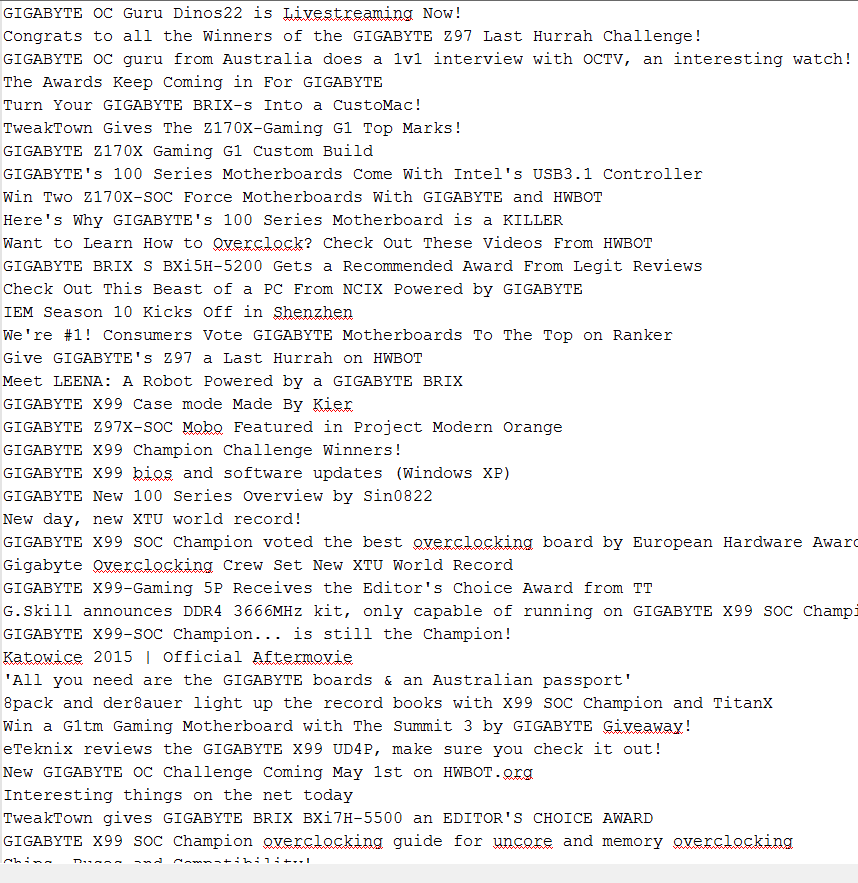
Work:

To start this assignment, I chose the blog ‘http://gigabytedaily.blogspot.com/’ due to my love for building and overclocking computers. This worked for me in that the feed had 150 entries (I stopped looking at that point) and was able to be parsed rather easily. To achieve this task, I modified the program from assignment eight (generatefeedvector.py, PCI) to show and save all of the data that we were looking for.





After running this program I received both a list of the titles of each of the 100 entries (limited in URL - <http://gigabytedaily.blogspot.com/feeds/posts/default?max-results=100>) and the entire feed data. The data is stored in ‘entryfile.txt’.



After looking at all of the articles, I decided that it would be best to make my categories:

1. HowTo – How to guides, articles on building and fixing.

2. Competitions – entries on previous, occurring, or future competitions and outcomes.

3. Sales – Entries attempting to sell a product.

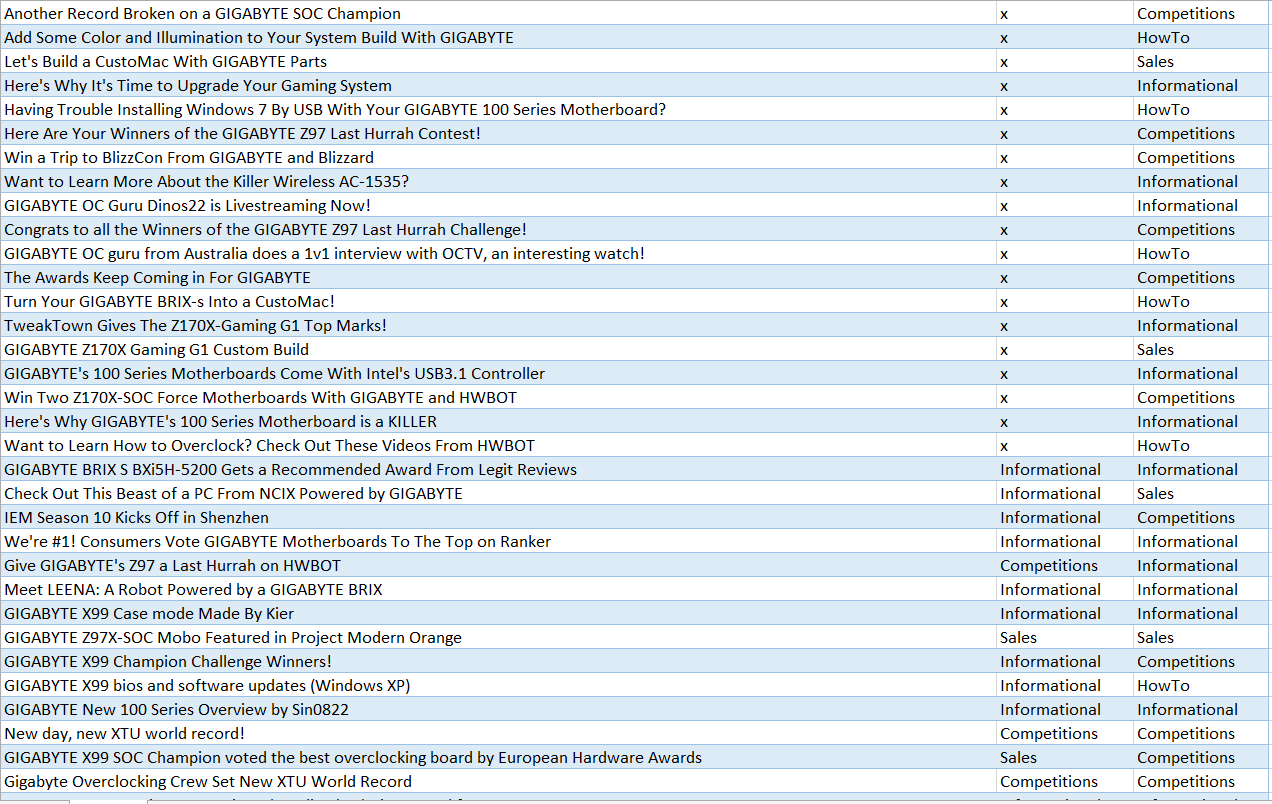
4. Information – General information.

2. Manually classify the first 50 entries, and then classify (using the fisher classifier) the remaining 50 entries.

Create a table with the title, predicted category, actual category, and cprob() and fisherprob() for the actual category.

To begin this problem, I modified PCI’s ‘feedfilter.py’ file to let the user input the first 50 categories and then let the program learn from it and make guesses for the next 50. The output from compilation is shown in ‘computing.txt’. With help from Ryan Condotta’s variation of PCI’s ‘docclass.py’, I was able to run this smoothly and efficiently. It also helped with the compilation of the fisher and cprob scores. I ended up with the table below:

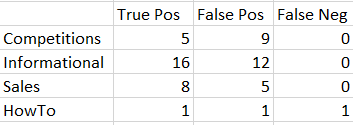




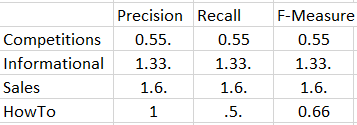


3. Assess the performance of your classifier in each of your categories by computing precision, recall, and F-measure.

To start this problem, I had to look at the predictions made by the program in problem two and compute the amount of TP (true positive), FP (false positive), and FN (false negative) values for each category.



From this, using the formula from the week 12 slides, I was able to calculate the precision, recall, and F-measure values:



The first three categories were the same due to the fact that I did not have any False Negatives, different on the ‘HowTo’ category. This shows that Informational and Sales were very accurate in their assumptions, while Competitions and HowTo were less accurate.