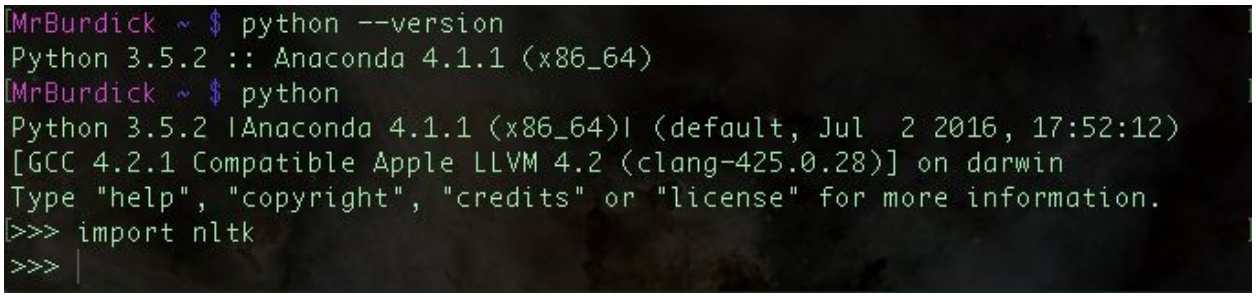
Jack Burdick
20Sept16
CAP6776

Tasks

- Results can be viewed in the /results folder
- Github repository of this solution can be found here:
    - https://github.com/JackBurdick/information_retrieval_TFIDFAndCosineSimilarity

Note: I received permission to use python3, rather than python2 from Dr Wang via email on 17Sept16

1. Install Python and NLTK (I use anaconda, and prefer python3)



2. Tokenize documents into words, remove stop words, and conduct stemming
    - Please see ./results/process_text.txt, sample shown below

```
---------------------------------- word_tokenize --> contents length: 498
['britain', "'s", 'barclays', 'plc', 'said', 'on', 'monday', 'it', 'was', 'in', 'talks', 'with', 'u.s.', 'group', 'morgan', 'stanley', 'abo
----------------------------------------------------------------

---------------------------------- removeStopWordsFromTokenized --> contents length: 310
['britain', "'s", 'barclays', 'plc', 'said', 'monday', 'talks', 'u.s.', 'group', 'morgan', 'stanley', 'global', 'custody', 'business', 'ban
----------------------------------------------------------------

---------------------------------- performPorterStemmingOnContents --> contents length: 310
['britain', "'s", 'barclay', 'plc', 'said', 'monday', 'talk', 'u.s.', 'group', 'morgan', 'stanley', 'global', 'custodi', 'busi', 'bank', 's
----------------------------------------------------------------

---------------------------------- removePunctuationFromTokenized --> contents length: 250
['britain', "'s", 'barclay', 'plc', 'said', 'monday', 'talk', 'u.s.', 'group', 'morgan', 'stanley', 'global', 'custodi', 'busi', 'bank', 's
----------------------------------------------------------------

---------------------------------- convertItemsToLower --> contents length: 250
['britain', "'s", 'barclay', 'plc', 'said', 'monday', 'talk', 'u.s.', 'group', 'morgan', 'stanley', 'global', 'custodi', 'busi', 'bank', 's
----------------------------------------------------------------
```

1. I word tokenized the documents
    a. nltk.tokenize.word_tokenize(),
2. Removed stop words
    a. nltk.corpus.stopwords.words("english")
3. Porter stemming
    a. nltk.stem.PorterStemmer()
4. Remove punctuation
    a. String.punctuation

   b. Custom - these were additional, manual, punctuation I chose to remove from the documents. I could have removed these with string.punctuation if I modified the document in string form, but I choose to perform list comprehensions and minimize string manipulations and therefore needed to include these manually
      i. ``
      ii. ‘’
      iii. --
5. Convert to lowercase
   a. Ensure terms are uniform

3. Calculate tf-idf for each word in each document and generate a document-word matrix
   - Please see ./results/tfidf.txt, sample shown below

194	fedel	0.000000000000	0.000000000000	0.000000000000	0.000000000000	0.039284057781
195	fee	0.000000000000	0.000000000000	0.046800821384	0.000000000000	0.000000000000
196	ferri	0.040925145224	0.031866754923	0.000000000000	0.039192056059	0.000000000000
197	fifti	0.000000000000	0.000000000000	0.046800821384	0.000000000000	0.000000000000
198	fight	0.000000000000	0.000000000000	0.000000000000	0.000000000000	0.039284057781
199	finalis	0.034427522344	0.026807318853	0.026366766409	0.032969593103	0.000000000000
200	financ	0.000000000000	0.038389502478	0.037758608010	0.000000000000	0.000000000000
201	financi	0.081850290448	0.063733509845	0.000000000000	0.078384112119	0.000000000000
202	fininvest	0.000000000000	0.000000000000	0.000000000000	0.000000000000	0.078568115562
203	firm	0.040925145224	0.031866754923	0.000000000000	0.039192056059	0.000000000000
204	fish	0.000000000000	0.000000000000	0.000000000000	0.000000000000	0.039284057781
205	fix	0.000000000000	0.047582798815	0.000000000000	0.000000000000	0.000000000000
206	flow	0.000000000000	0.047582798815	0.000000000000	0.000000000000	0.000000000000
207	follow	0.049302038059	0.000000000000	0.000000000000	0.047214206055	0.000000000000
208	footbal	0.000000000000	0.000000000000	0.000000000000	0.000000000000	0.039284057781
209	formula	0.000000000000	0.047582798815	0.000000000000	0.000000000000	0.000000000000
210	forward	0.000000000000	0.038389502478	0.000000000000	0.000000000000	0.063388260931
211	franc	0.034427522344	0.053614637706	0.026366766409	0.032969593103	0.000000000000
212	fraud	0.000000000000	0.000000000000	0.000000000000	0.000000000000	0.196420288906
213	free	0.040925145224	0.031866754923	0.000000000000	0.039192056059	0.000000000000
214	french	0.040925145224	0.031866754923	0.000000000000	0.039192056059	0.000000000000
215	friday	0.000000000000	0.000000000000	0.000000000000	0.000000000000	0.117852173343
216	front-end	0.000000000000	0.000000000000	0.046800821384	0.000000000000	0.000000000000
217	futur	0.034427522344	0.053614637706	0.052733532817	0.032969593103	0.000000000000

The main function used was

```
tfidf = TfidfVectorizer(tokenizer=processData, stop_words='english')
tfs = tfidf.fit_transform( rawContentDict.values() )
```

Where processData was a custom function that performed the steps listed above in part 2.

4. Calculate pairwise cosine similarity for the documents
   - Please see ./results/cosine_similarity.txt, sample shown below

COSINE SIMILARITY

	100554newsML.txt	100593newsML.txt	100618newsML.txt	130040newsML.txt	137871newsML.txt
100554newsML.txt	1.00000000	0.77192476	0.13496567	0.97508619	0.10226431
100593newsML.txt	0.77192476	1.00000000	0.17310706	0.74710164	0.11696533
100618newsML.txt	0.13496567	0.17310706	1.00000000	0.13309524	0.07654495
130040newsML.txt	0.97508619	0.74710164	0.13309524	1.00000000	0.09585856
137871newsML.txt	0.10226431	0.11696533	0.07654495	0.09585856	1.00000000

Jack Burdick
20Sept16
CAP6776

This function was imported from sklearn;

```
from sklearn.metrics.pairwise import cosine_similarity
```

Results were produced by looping through a list of filenames and their values. The values were then compared to one another and printed/written into a table format (shown below)

```
def print_CosineSimilarity( tfs, fileNames ):
    #print(cosine_similarity(tfs[0], tfs[1]))
    print("\n\n\n========COSINE SIMILARITY==========================================================================\n")
    numFiles = len(fileNames)
    names = []
    print('                  ', end="")    #formatting
    for i in range(numFiles):
        if i == 0:
            for k in range(numFiles):
                print(fileNames[k], end='   ')
            print()

        print(fileNames[i], end='   ')
        for n in range(numFiles):
            #print(fileNames[n], end='\t')
            matrixValue = cosine_similarity(tfs[i], tfs[n])
            numValue = matrixValue[0][0]
            #print(numValue, end='\t')
            names.append(fileNames[n])
            print(" {0:.8f}".format(numValue), end='        ')
            #(cosine_similarity(tfs[i], tfs[n]))[0][0]

        print()
    print("\n\n=================================================================================================\n")
```

Final Thoughts

I enjoyed nltk and sklearn, as I have not used these before. I'd rather produce a more generic tool to perform these functions that I could reuse, but maybe in the future I'll rewrite a command line function to just compare two files and print the important information (rather than every step).

Questions/comments

Even though I removed stop words in an earlier function by using the below function earlier in the assignment

```
stop_word_set = set( nltk.corpus.stopwords.words("english") )
filteredContents = [ word for word in tokenizedContents if word not in stop_word_set ]
```

If I included the "stop_words='english'" in the TfidfVectorizer function, additional stop words seemed to be removed. The word I noticed was "yet". I found this strange.

```
tfidf = TfidfVectorizer(tokenizer=processData, stop_words='english')
```

Jack Burdick
20Sept16
CAP6776

Known Weaknesses

- My file write system is clunky and dangerous, directories need to be created before running the script. Additionally, these functions should be wrapped in try blocks
- Adding manual punctuation to remove manually isn't ideal…
- My methods for printing the results are brute force, I tried using tabulate, but I wasn't as impressed with the results as I hoped.
- I should include the filename of the steps in the process_text.txt output.

References

Sites I referenced when producing my results

- Print with fp precision
    - https://stackoverflow.com/questions/15263597/convert-floating-point-number-to-certain-precision-then-copy-to-string
- Cosine similarity
    - http://scikit-learn.org/dev/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html
- Function comments
    - https://www.python.org/dev/peps/pep-0257/
- New print format
    - https://docs.python.org/3/tutorial/inputoutput.html