Andrew Kirby
Language Technology
10/5/2020

Part 1

My approach:
I opened the Word Similarity Test collection and put it into a pandas dataframe. I then added two columns that contained the first synset result for each word, These columns are named w1syn and w2syn, respectively. I used a cool lambda statement that resembled the stuff I was working with in Semantics 1:

```
df = df.join(df['w1'].apply(lambda word:
pd.Series(wn.synsets(word))))
```

I wasn't able to simply use .apply with the wn.synsets; the lambda is a helpful way to quickly define a function and have it iterate over one of my columns.

Because of this approach, I got an error because the first result of two of the words resulted in synset pairs of different parts of speech. I dropped the two rows, because it was faster; I should have replaced the cells with the appropriate synset codes, but I couldn't quite figure out how to do that with pandas. I then wrote functions for each of the six methods. I put the results of each of these functions into six further columns in the dataframe, one for each method. I then wrote a function to calculate the Spearman correlation for each of these methods. I got the following correlation results:

Method	Spearman Correlation	P-value
Path Similarity	0.4537	0.0000
Wu-Palmer	0.5457	0.0000
Leacock-Chodorow	0.4537	0.0000
Resnik	0.5827	0.0000
Jiang-Conrath	0.4841	0.0000
Lin	0.5693	0.0000

The strongest correlation here was with the Resnik method at .05827. The Path Similarity and Leacock-Chodorow correlation values were identical, despite having different values for the individual word pairs.

I covered 201 word pairs.

Parts 2 & 3 (Not complete 10/5/20)

I struggled with these parts. There are three steps: download a news file, tokenize, and run the given scripts on it. First, I opened the .gz file and converted it to a text file using the gzip library. Then, I used the word_tokenize function on my text file. I initially tried to directly tokenize the gz file, but then it would keep outputting my tokenized data into a blank file.

```
with open ('text1.txt', encoding = 'utf-8') as fin,
open('tokens.txt','w', encoding = 'utf-8') as fout:
     for line in fin:
     tokens = word_tokenize(line)
     print(' '.join(tokens), end='\n', file=fout)
```

For the Word Similarity test collection, there were three columns: word 1, word 2, and the similarity score. I dropped off the score column, and exported the dataframe to a .tsv file.

I used the following command to run the ppmi.py script. Results.tsv is my result file; the ws353_twocol.tsv file is my two-column input file; the tokens.txt file is my tokenized news data file.

```
!python ppmi.py --results_path results.tsv --pairs_path
ws353_twocol.tsv --tok_path tokens.txt
```

I kept getting UnicodeDecode errors with the ppmi script, despite having specified that the encoding should be ‘utf-8’.

For the Word2Vec script, I also ran into a roadblock. I entered the following command to run it:

```
!python word2vec.py --results_path resultsw2v.tsv --pairs_path
ws353_twocol.tsv --tok_path tokens.txt
```

I got a ValueError here; it expected two values (from a two-column tsv?) and so did not generate information after running.

I also tried running the scripts on a test document that I created with four simple sentences, separated in the same way as the news files. The sentences:

> I like trains.
> Chickens are nice.
> Do you like airplanes?
> No, I don't like tigers.

I then tokenized it, and I got the following from the ppmi:

```
INFO: 483 words tracked
INFO: 204 pairs tracked
INFO: 21 tokens counted
INFO: 0 pairs covered
```

I got the following error for the Word2Vec script on my test document:

```
RuntimeError: you must first build vocabulary before training the
model
```