Andrew Pelkey
MP1: Word similarity

Copy of original notes:

I paste these here only for good record keeping. I've made no revision to my code or answers for Part 1.

> Part 1
>
> Answers:
> Spearman correlation using Path Similarity: 0.4291
> Spearman correlation using Wu-Palmer Similarity: 0.5309
>
> Coverage: 201/203
>
> My approach here was very basic. I used the first synset delivered for each word. Originally I wanted to wrap everything in a function that I could call for each method, but discovered that the methods needed different arguments.
>
> Running Path Similarity and Wu-Palmer went off without a hitch and I got the Spearman Rhos listed above. However with the other methods, even when defining an IC argument as required, I got the same error in each case that I wasn't able to resolve: computing the least common subsumer requires the synsets being compared to have the same part of speech. I can imagine a way of ensuring that this error doesn't throw, but I ran out of time.
>
> As for coverage, I was confused by the meaning of a similarity method returning "None". This was true only in two cases, and in order to compute the Spearman Rhos I just removed the similarity score and the corresponding human score from the calculation. But there were no words that did not have a WordNet synset, so I would have thought coverage was actually 203/203.

PART 2

The notes below replace my original notes for Part 2.

Data and code used
news.2013 from WMT news crawl directory
news_tok.py
ppmi.py (supplied)
ppmi_rho.py

Results
- Spearman correlation coefficient: -0.0952
- Pairs covered: 186

Problems and roadblocks

- My original implementation coded a space before and after the tab (\t) separating the words in each pair from the ws353 file. Thus, the ppmi.py script could easily read the file, but it could not find any of the pairs in the tokenized data. I revised and finally got results.
- I did not know how to run something from the command line and use command line arguments. Overcoming this obstacle feels great!

Comments
My code doesn't save the actual word pairs anywhere as I didn't see a need for that. I just try to find each pair from the output of the ppmi.py script within the original ws353 word pairs file, and if there's a match (in either order), I save the corresponding numbers to their respective lists, and then calculate the Spearman rho from those two lists.

The Spearman correlation coefficient seems awfully close to 0, suggesting the two variables are not monotone functions of each other at all. This seems wrong. I've carefully reviewed the two sets being compared in the function and I have confirmed that they are of equal length and that the figures in the set correspond to the expected pairs. So I don't have a good way to explain the poor performance here.

I knew that Cass had also run into an issue with the ppmi script returning 0 pairs covered, so I did reach out to him for a hint. But his issue didn't map to mine at all, so I had to debug myself.

PART 3

Data and code used
news.2013 from WMT news crawl directory
word2vec.py (supplied)
word2vec_rho.py

Results
- Spearman correlation coefficient: 0.6564
- The script as designed returns a word2vec score for all 203 pairs

Comments
This part was considerably easier to do: the tokenization had already been done and working with the terminal and the python required to do the calculation was much more fluid (because of all the trial and error to this point).