LING83600
Lab 1
Yulia Spektor

Part I

For WordNet synset similarity, I decided to find synsets with the highest similarity for each word pair. To do that, I used two for loops to loop through the synsets for each word in the pair and track the highest similarity value.

The difficult part was figuring out what is required for computing each type of similarity. For instance, Leacock-Chodorow similarity, Resnik similarity, Jiang-Conrath similarity, and Lin similarity all require the words that are being compared to have the same POS tags. Also, the similarity methods that use Information Context have an additional constraint due to certain POS tags ("s", "a") not present in the IC. Once I figured out these requirements, I decided to put all the calculations into one function and iterate over each line of the TSV file containing the word pairs. The similarity values were saved into lists (since they are ordered), which were then used to count Spearman's rho coefficients for each method.

Results

Path similarity: 0.5971,
Jiang-Conrath similarity: 0.5954,
Leacock-Chodorow similarity: 0.615,
Lin similarity: 0.6089,
Resnik similarity: 0.6376,
Wu-Palmer similarity: 0.6389,

I made sure to track words that are not covered by WordNet. I got 100% coverage.

Part II

For this task, I worked with the smallest WMT news crawl data file available (176M) – the one from 2007. After unzipping the file in the command line using *gzip*, I decided to tokenize each line read from the file while iterating over it in a for loop, and then save the tokenized sentence into a new .txt file.

The biggest problem I had was with the encoding of the data. After the initial attempt (and a couple of Unicode decode and encode errors), I realized that I need to check the encoding of the data first. For that, I read the file in the binary mode and used the *chardet* library to determine the encoding of each line and decode it. After decoding, I then tokenized the sentence using the NLTK *word_tokenize* method and saved the tokenized string into a file. I also used the try/except statements to catch any strings that were not decoded or encoded correctly. However, that still resulted in some strings having incorrectly decoded symbols. This was not a problem for calculating PPMI, but it did cause trouble with the word2vec.py script since the *gensim* model could not be trained with incorrectly encoded symbols.

Then, I ran a function to determine the overall encoding of the file, thinking I could just use that format. It was determined to be utf-8, but even when limiting the input data only to the one encoded in utf-8, I was still getting some incorrectly encoded symbols in the tokenized data. So, in the end, I decided to restrict encoding and keep only the data that was in ASCII, since ASCII can encode all English characters, and this way I was sure to get all English words and discard all words and strings that were

not in English and were thus incorrectly encoded. I had to discard 42,670 out of 3,345,329 sentences, which was less than 2% and thus should not have affected the accuracy of the downstream PPMI calculation.

To count the Spearman's rho correlation coefficient, I wrote a function that takes 2 TSV files as an input, creates a dictionary of word pairs and values of the programmatically produced similarity statistics, and then loops through the file with human judgement stats to word pairs covered by both methods. The found values are saved into lists to count the Spearman's rho coefficient.

I played around with the window size a bit, when running the ppmi.py script, and got the following results.

PPMI results

Window=10
- Coverage: 74.88% (152 pairs)
- Spearman's rho: -0.1023

Window = 15
- Coverage: 76.35% (155 pairs)
- Spearman's rho: -0.0745

Window = 7
- Coverage: 70.94% (144 pairs)
- Spearman's rho: -0.097

Window = 5
- Coverage: 65.52% (144 pairs)
- Spearman's rho: -0.0182

Part III

I did not write anything new for this part. I just ran the provided word2vec.py script and then used the script from Part II to compute the Spearman's rho coefficient for the produced data.

Results

Spearman's rho: 0.6055
Coverage: 99.51%

Part IV

The Spearman's rho correlation coefficients produced in Part I are positive and show overall increasing monotonic trend between the human judgements of similarity and the methods used. Same goes for Part III – there is a positive correlation between the Word2Vec similarity data and the human judgements. I was somewhat surprised by the PPMI results, to be honest, since they show a negative correlation and thus a decreasing monotonic trend between the human judgments and the PPMI results. This may be due to the fact that PPMI considers only co-occurrences of the words and is more dependent on the contexts.