Lab 1: Word Similarity

Osama Qureshi

## Part 1: WordNet

NLTK provides a very simple API for computing all six path similarity measures. Two minor points that needed attention during implementation were:

1. Resnik, Jiang-Conrath (JC) and Lin similarity measures require an information content table as an additional input, and I used NLTKs pretrained WordNet IC for the Brown corpus.
2. JC similarity result becomes undefined if both words are the same, as the denominator in the formula *1/ (IC(s1) + IC(s2) - 2 \* IC(lcs))* simplifies to 0. NLTKs implemention of the jcn\_similarity junction in this case results in an extremely large number, 1e300, and so I assigned a similarity value of null for such instances.

Additionally, for each similarity measure, there were cases when there was no direct path between the first (primary) word senses for two particular words, but using one of the other possible word senses resulted in a positive similarity. An example of this is for the words ‘live’ and ‘stock’. Computing the path similarity for the primary word senses of both words results in a similarity of None, but there is a path between the two words for a different combination of their possible senses. Therefore, I computed the similarities for both the primary senses of each word, and maximum possible similarity for any combination of senses for each pair of words. Using the maximum possible similarity results in a slightly higher correlation and coverage for each similarity measure.

Overall, Resnik similarity resulted in the highest spearman correlation, but it had very poor coverage. All 3 non-IC similarities, Path, LCS and WUP similarities had the highest coverage, including complete coverage when using the maximum possible similarity. Table 1 and 2 show results for Correlation and Coverage.

|  | **path** | **lcs** | **wup** | **resnik** | **jcn** | **lin** |
| --- | --- | --- | --- | --- | --- | --- |
| **primary** | 0.9955 | 0.9974 | 0.9959 | 0.9974 | 0.9953 | 0.9924 |
| **maximum** | 0.9957 | 0.9979 | 0.9963 | 0.9991 | 0.9955 | 0.9954 |

Table 1: Correlation

|  | **path** | **lcs** | **wup** | **resnik** | **jcn** | **lin** |
| --- | --- | --- | --- | --- | --- | --- |
| **primary** | 201 | 201 | 201 | 160 | 196 | 160 |
| **maximum** | 203 | 203 | 203 | 192 | 201 | 192 |

Table 2: Correlation

## Part 2: PPMI

For Part 2 and Part 3, I used two different new corpora, 2007 and 2018. The 2007 corpus contained 3.38M sentences, whereas the 2018 corpus contained 18.1M sentences. For preprocessing, I removed all punctuation and special characters from both corpora, and used nltk.word\_tokenize function for tokenization, before running the provided ppmi.py script and computing the correlation for both corpora. As the results in Table 3 show, using the 2018 corpora results in a slightly higher correlation, and much larger coverage. This result makes sense, as one would expect higher correlation and coverage when using a larger corpus with much more data.

Additionally, I modified the ppmi.py script to include add-alpha smoothing, and implemented the functionality by changing by simply modifying how px, py, pxy were computed.

Initial*: p(word) = cwords[word] / nwords*

Modified: *p(word) = (cwords[word]+alpha) / (nwords\*len(cwords)\*alpha)*

Initial: *p(pair) = cpair / npairs*

Modified: *p(pair) = (cpair+alpha) / (npairs\*len(cpairs)\* alpha)*

I used 0.5 smoothing for my experiment, and computed the correlation for both corpora. As the results in Table 3 show, introducing add-alpha smoothing ends up reducing the correlation. One potential explanation for this may be that smoothing, even with alpha=0.5, is shifts an excessive amount of the probability distribution to low count words and pairs, and this negatively affects the probability of words pairs relevant for our experiment.

|  | **corpus2007** | **corpus2018** | **corpus2007\_alpha** | **corpus2018\_alpha** |
| --- | --- | --- | --- | --- |
| **correlation** | 0.9751 | 0.9767 | 0.9284 | 0.9383 |
| **coverage** | 153 | 180 | 153 | 180 |

Table 3: PPMI

## Part 3: Word2Vec

Using the data preprocessing as in Part 2, I ran the provided word2vec.py script on both corpora, and computed the spearman correlation. I also modified the word2vec.py script to change the genism model being trained from Word2Vec to FastText. This implementation was very straight forward, and only required changing 2 lines of code in the word2vec.py script.

def main(args: argparse.Namespace) -> None:

sentences = gensim.models.word2vec.LineSentence(args.tok\_path)

w2v = gensim.models.Word2Vec(

sentences,

..

def main(args: argparse.Namespace) -> None:

ft = gensim.models.FastText(

corpus\_file=args.tok\_path,

..

As the results in Table 4 show, as expected, using the larger corpus results in a higher correlation and coverage for Word2Vec. Word2Vec embeddings also have a much higher coverage than PPMI, with full coverage for Word2Vec embeddings trained on the larger 2018 corpus, and just 1 missing pair of words for the embeddings trained on the smaller 2007 corpus. Using the FastText embeddings further increases the correlation slightly, and also results in full coverage for both corpora.

|  | **word2vec2007** | **word2vec2018** | **fasttext2007** | **fasttext2018** |
| --- | --- | --- | --- | --- |
| **correlation** | 0.9958 | 0.9959 | 0.9961 | 0.996 |
| **coverage** | 202 | 203 | 203 | 203 |

Table 4: Embeddings

### Part 4: Summary

Overall, three Wordnet similarity measures result in the highest correlation with human judgments of similarity, followed by word embeddings, in particular FastText. However, using only the primary word sense for each word with WordNet does result in a significant drop in correlation. PPMI performs significantly worse compared to both WordNet and Word2Vec/FastText, and including add-alpha smoothing does not improve the performance either.

For Coverage, Wordnet similarity measures again have very high results, including full coverage for the 3 connection-based similarity measures. However, using only the primary sense results in a significant drop in coverage, particularly for the Information Content based measures. Word2Vec (and FastText) had full or almost full coverage for both corpora used. PPMI again had the worst results compared to WordNet and Word2Vec/FastText, and only achieved a maximum coverage of 180 on the large 2018 corpus, and had the lowest coverage when using the 2007 corpus.