CSC2611 Lab

Reid McIlroy-Young

January 24, 2019

Part 1, Meaning from Text

After loading the Brown corpus, removing non words and converting to lower case. The top 5 words were: the, of, and, to, and a, while the last 5 were: letch, haney, killpath, rourke, and vertex.

After creating the bigram count matrix, calculating the PPMI matrix, doing LSA and examining the cosine differences , I calculated the following values for the Pearson coefficients:

The Pearson coefficient between between humans and $M1$ is: 0.1047

The Pearson coefficient between between humans and $M1+$ is: 0.1957

The Pearson coefficient between between humans and $M2_{10}$ is: 0.0708

The Pearson coefficient between between humans and $M2_{100}$ is: 0.1512

The Pearson coefficient between between humans and $M2_{300}$ is: 0.1593

These are pretty low, and I think lower than the original paper, so there might be something wrong with my PPMI calculation.

The code for this section can be found in the Github repo: reidmcy/CSC2611-Lab and the code is in the Implementation Jupyter notebook, section 1.

Part 2, Meaning Construction from Text

After loading the Google word2vecs and calculating the cosine differences. The Pearson coefficient between between humans and the word2vec model is: 0.7721. This is much better than any LSA, suggesting that the concurrences are not the only component of similarity and that word2vec is better at extracting the more nuanced results. Or that the similarities just require scale to measured and the Brown corpus just isn't big enough.

For the analogies test I limited it to the *family* analogies for the semantics and *gram7-past-tense* analogies for the syntactic. This is after a bit of

experimenting to find tasks with good coverage by $M2_{300}$'s vocabulary and at least some hits by the word2vec model in the top suggestion.

When the analysis was run word2vec was correct 12 out of 600 times on the syntactic task and was correct 4 out of 90 times on the semantics task. While $M2_{300}$ never got a single correct result. Both of these are very poor showing and upon inspection part of it looks to be the analogies have a different vocabulary than either model, as many times the model would suggest the stemmed version of the correct answer, now that said the unstemmed was in the their vocabulary. I also was only looking at the first result when doing the word vectors math, which could result in one of the input words being the best match. Doing a top 5 match might be a more accurate depiction of model's skill.

The code for this section can be found in the Github repo: reidmcy/CSC2611-Lab and the code is in the Implementation Jupyter notebook, section 2.

To improve the performance of the models analogies performance could be part of the backpropagation. At some interval words belonging to some analogies task could be sampled and their weights pushed closer to the correct values by backpropagation. This unfortunately removes the language agnostic nature of these models, and as such would be difficult to implement. A less extreme version could be considered, where the syntactic analogies are generated by looking at the stemming rules for words in the language and checking all the unstemmed version of a word for their syntactic analogies.