Carlos Adrian Bueso

Dr. Dmitriy Dligach - COMP329 Natural Language Processing - Homework 4 Written Report

For this project, we had to train a word2vec model on a corpus of our choice. In my case, I decided to use the Brown Corpus, easily accessible from the *NLTK* library in python. The Brown corpus consists of 500 texts, each consisting of just over 2,000 words. Another library, *gensim*, was used for implementing the word2vec algorithm. Because I had previous experience with *Google Colaboratory*, I decided to use it for training the model.

Training the model took 29 seconds, and we use gensim's save() function to store it in memory for future use. These are some of the similarity queries we ran for experimentation:

'university','school' = 84.517%

'kid','child' = 78.087%

'clock','watch' = 88.384%

'apple','color' = 76.625%

'airplane','apple' = 79.186%

'music','water' = 66.334%

After training and experimenting with word2vec and a corpus of our choice, we loaded a larger pre-trained model. We used the Google News Dataset model, which is trained on 100 billion words. For testing the big model qualitatively, we used gensim's evaluate_word_analogies() function (inspired by Google's testing word2vec function), which tests for syntactic analogies of comparative types, like: bad is to worse, as good is to x. The overall evaluation score on the entire evaluation set was 65.364%. The analogy sections are listed below with their respective correct and incorrect values.

Section	Correct analogies	Incorrect analogies
Capital common countries	168	14
Capital world	344	33
Currency	4	26
City in state	1240	392
Family	241	65
Adjective to adverb	178	814
Opposite	235	317
Comparative	1029	303
Superlative	486	270
Present participle	674	318
Nationality adjective	848	59
Past tense	815	745
Plural	798	194
Plural verbs	334	368

Some alarms are raised, for examples when dealing with currencies. Also, the adjective to adverb, opposites, and plural verbs seems to have a low accuracy.

After using this type of analogy evaluation, we also evaluated the model with the WordSim-353 dataset, which you can access with the gensim library. The function evaluate_word_pairs() returns the Pearson correlation coefficient with 2-tailed-p-values, the Spearman rank-order correlation coefficient between both WordSim-353 and our model, and also the ratio of pairs with unknown words. The results were as follow:

Pearson correlation coefficient with 2-tailed p-value:

(Correlation = 0.55453, p-value = 1.94521e-27)

Spearman rank-order correlation coefficient between the similarities from the dataset and the similarities produced by the model itself, with 2-tailed p-value:

(Correlation = 0.60194, p-value = 3.18663e-33)

The ratio of pairs with unknown words: 8.49858

Because the Pearson correlation is greater than 0.50 by +0.05453, we can say the model has a strong correlation. The Spearman rank correlation coefficient of 0.60194 means there is a strong association of ranks between the datasets.

Finally, we did some vector arithmetic to solve analogies using the default cosine similarity measure. Here are a couple of examples tested, the resulting output, with its corresponding similarity value:

(1) 'woman' + 'king' – man = **'queen' (0.7118)**
(2) 'ice' + 'liquid' – 'water' = **'Ice' (0.4140)**
(3) 'girl' + 'man' – boy = **'woman' (0.8713)**
(4) 'green' + 'sky' – 'grass' = **'bright' (0.4222)**
(5) 'puppy' + 'cat' – 'dog' = **'kitten' (0.7635)**

Some of these examples turned out pretty good, with the similarity value showing a potential use of acting as a confidence level. For example, (1), (3), and (5) show extremely good results, solving the analogy in a completely accurate way. Another interesting example was (4), where we were expecting the word 'blue' as an adjective for sky, but got 'bright' instead, which is indeed an adjective for 'sky' but not precisely accurate. Finally, some results did show completely unexpected and unreasonable results, like (2), where we got one of the vectors used in the calculations.

Working with word2vec was very interesting. It is amazing to think how useful plain text can be for applications in NLP. It also shows and asserts once again Google's dominance in computing. To think these vectors lie in a 300-dimensional space is mind-boggling, and makes me wonder what will computer scientists be using in the next couple of decades.