Assignment3 Report

Junhan Liu(20848916)

Running instruction

1, `cd #the assignment folder`, i.e. `cd /user/junhanliu/614_a3`

2, In your command line type `python3 main.py` or `python3 main.py /xxx(directory)/.../#assignment folder/data` to initialize script

3, The driver script(main.py) takes only one parameter or no parameters. If passing in a parameter, feed the data folder path to the script. If none is given, the script is going to use a default path of the current running directory using `os.getcwd()+'/data'`

4, The data folder contains `words.txt` which contains words for generating similar tokens, `pos.txt` and `neg.txt`. Ouput files include `1, similar_terms.csv` which has words similar to the given input words. `2, w2v.model` which is the trained model.

5, Every file needed are in `/data` folder.

Discussion

The model is looking for similar terms based on vectors; it is learning words as vectors. Thus it is not able to learn the true semantics of words. Using `gensim.word2vec.wv.most_similar()` function without giving positive and negative arguments to the function, the result is very interesting. It is not more similar to any side, since `good,bad,0.7670988440513611` good is 76% similar to bad, and vice versa. The reson is, like described before, the model does not consider the semantics, rather it learns the similarity based on the context. I.e. `1, the product is not good`, `2, the product is not bad`. The two words (good, bad) are similar based on the vectors. If using `model.wv.most_similar(positive=[row['word'], 'good'], negative=['bad'], topn=20)`, the result is more similar to `good`, and vice versa. The logic of the argument is `word+good-bad` to fully negate the word bad and similar words to `bad`.