Assignment 5

Sharath Chandra Bagur Suryanarayanaprasad
800974802
sbagursu@uncc.edu

Accuracies-

Accuracy of Glove: 24.53%
Accuracy of Lexvec: 39.48%

Discussion-

Initially all the classes are filtered from the given test file. A loop runs iterating through every line and segregates lines into their respective classes. Having obtained these, I use the Word2Vec model provided in the gensim package. The Word2vec model utilizes the following pretrained embeddings and generates respective vectors,

→glove.42B.300d.txt (Link: https://nlp.stanford.edu/projects/glove/)
→lexvec.commoncrawl.300d.W.pos.neg3.txt (Link:https://github.com/alexandres/lexvec)

Utilizing these pretrained embeddings in the Word2Vec models we use the Mikolov's analogy formula and compute cosine similarity. The equation used is,

$$d = \arg\max_{d \in \mathcal{V} \setminus \{a,b,c\}} \cos\left(\mathbf{v}_d, \mathbf{v}_b - \mathbf{v}_a + \mathbf{v}_c\right).$$

in order to compute the cosine similarity we first iterate through all the words in the vocabulary of each and every class mentioned in the test set. We the fetch a, b, c and d words from every line. Then using the Word2vec model, we fetch the vectors for these words. Further we compute the cosine similarity based on the equation for all words in the vocabulary. This gives us a dictionary of key value pairs which contain all words in the vocabulary as keys and their corresponding computed cosine similarities as values. Pulling the max value from the dictionary gives the most similar word. We then compare this fetched word with the word d in the given test set. If there is a match we consider it to increase the overall accuracy. The dimensions of the vectors obtained from the Word2Vec gensim model is a 300 dimensional vector. The functions computing the cosine similarity have been implemented using numpy arrays and respective math functions to obtain results based on the given equation.

Difficulties faced-

The main difficulty faced was dealing with large amounts of datasets of pretrained embeddings. Parsing through dataset and comparing values is processor intensive and takes a lot of time. Word2Vec model generated already generates vectors and these vectors are high dimensional vectors. Handling computations over large data gets intensive and hence numpy arrays were used which proved to be better in terms of processing.

Execution Steps-

Utilizing the links provided, kindly download the pretrained embeddings. Filenames after downloading have not been altered and have been used as is to directly load into program. The following filenames have been used,

➔glove.42B.300d.txt (Link: https://nlp.stanford.edu/projects/glove/)
➔lexvec.commoncrawl.300d.W.pos.neg3.txt (Link:https://github.com/alexandres/lexvec)

After downloading, run the file embeddings_glove.py to get the accuracy of glove model and file embeddings_lexvec.py to get the accuracy of lexvec pretrained embeddings model.

Problem 2-

One of the problems of word embeddings as mentioned in the question is that of antonyms having similar embeddings. From the Word2Vec model it can be observed that positive and negative words form a part of the list of similar words of a given word. For example running Word2vec similarity for the word hot gives the following as output,

As we can see even antonyms forms a part of the list of entries.

Having obtained data from pretrained embeddings which are nothing but matrices, we can establish 2 types of similarity which is first-degree co-occurence or sintagmatic association and second-degree occurrence or paradigmatic association between words that occur in similar contexts. First-degree co-occurences is extracted directly from the matrix - words that frequently appear near each other will have high values. The second degree co-occurence requires more effort with observations to be made on vectors and comparing these vectors. After the vectors are created the similarity between these vectors is computed by calculating the proximity between them. Further we can work out a way to find the cosine similarity between the vectors. Doing so gives measure of angle rather than the proximity distance. Hence if the vectors are orthogonal if the cosine similarity is 0 and 1 if the cosine similarity is almost the same between the vectors. There is, nonetheless, a problem with the notion of semantic proximity: it is impossible to distinguish between synonyms and antonyms of a word, between co-hyponyms, or even between hyponyms and their hypernyms, since they usually occur in the same contexts. Their resulting vectors are, as expected, extremely similar.

Citation: "Distinguishing Antonyms from Synonyms in Vector Space Models of Semantics", Bruna Thalenberg

Problem 3-

Analogies:

:biological-nomenclature-animals

toad Mesobatrachia lobster Nephropidae
oyster Ostreidae woodpecker picidae
squirrel Pteromyini squid Oegopsina

:country-cash-crop
africa coffee australia lentils
america oranges vietnam coconut
india rice russia barley

The above analogies are not a part of Mikolov's analogy dataset. When the pretrained models are run on these analogies the accuracy obtained is insignificant and equal to that of zero.