COMP6714 Project2 Report

Introduction

In this project, I made an implementation of word embedding for adjectives. The objective of this project is to obtained embedding to preserve as much synonym relation as possible. There several phases in this project including preprocessing, building data set, training and testing. At the test period, my result will be measured against a set of ground truth from development set.

Methodology

Data processing

First, read the data from zip file. I use a dictionary to store the information I got from input file: a list of document, each element is data from a txt file, a dictionary that records the frequency of each word in all txt files and a list of adj_word that only record the word that is judged as adjective word by spacy.
All the data in document list has already been preprocessing by spacy. The method is to take the lemma of each standard word. If the word is tagged as adjective, then put it in the adj_word list.
Then, save the data to a file. The format I use to store data is based on json: using json dump and json load.
After read the data back, I build another dictionary: word to id. In this step, I set a limit that all the word has a lower frequency than it will be marked as UNK. So only part of word will stay still in the dictionary.

Train

The part of build model is like word2vec demo. As required, I use AdamOptimizer and

sample_softmax_loss function. The parameters for this part is:

batch_size = 128

num_samples = 1

num_sampled = 100

learning_rate = 0.002

vocabulary_size = the length of word to id dictionary in previous part

The part of train is based on keeping generating batchs until get a whole mini batch set which is totally different from word2vec demo.

In the skip gram model of this part, I shuffle all the documents and start from first word from a document. Compute the window of current word and then random sample the context words of it. I make it skip the UNK word for both center word and context word. After the session reach the final steps, run the normalized_embedding by session and save all the word to the embedding file.

Evaluation

I use genism to load the model of the adjective embedding file. Then use the most similar function that can compute the top-k most similar word vector by cos similarity. During selecting word, only search other adjective word, this can certainly improve the performance. Use the dev set and record the total and average hits.

Results and Discussion

The current result of hits tested by Docker environment is around 8.8 hits.

The main difference of my code to demo code is that I separate each document and the batch won't generate words in different documents.

I tried three different ways of generating batch: all txt file as a long word list, separate each document and separate each sentence. The result of separate by sentence is around 7.2 hits and the result of long word list is around 7.6 hits, both not good as separate each document.

I am confused at the limit part (transform the word to UNK if they appear less than certain times), because the limit will drop some adjective word too, so some adj in test cases may not appear in the final embedding file. But the limit actually improve the

performance a lot. Finally, I add it because I think the words that have such low frequency should not count much for the embedding file.

Another tunable parameter I found not work as my predict is size of skip window. I used to set it above 6 and random sample about 10 words. And it performs much worse than skip_window =1 in current set.

Conclusion

The result of hits in my final code is still not good as I am looking forward. I think the main reasons are small training set and simple neural network model. The preprocessing and building data set is important that influence the result a lot.