Report – Assignment 1

Course: - Advanced NLP

Name – Dhaval Taunk | Roll No.- 2021701028 | Date-11-09-2021

1. **Code and model details:-**
    a. Code files: -
        i. SVD_co-occurence.ipynb
        ii. word2vec-cbow.ipynb
    b. Models: -
        i. Models.zip
            1. tokens_list.txt
            2. co_occurence.npz
            3. svd_matrix.txt
            4. tokenizer.pickle
            5. weights.best.hdf5
            6. cbow_vecs.txt

2. **Download models.zip:-**
    a. Link:- https://drive.google.com/file/d/1ICRS5sgw6xlf6AKlHk0o5RxD-qVoK_gU/view?usp=sharing

3. **Unique tokens used:-**
    a. SVD_co-occurrence.ipynb:- 100679
    b. Word2vec-cbow.ipynb:- 100715

4. **How to run the code:-**
    a. First download the models.zip from the above given link. Extract it to the same folder where the code is present. The directory structure should look like below:
        i. SVD_co-occurence.ipynb
        ii. word2vec-cbow.ipynb
        iii. models folder after extracting models.zip
    b. SVD_co-occurrence.ipynb:-
        i. This notebook uses the following files:-
            1. **tokens_list.txt** :- contains unique tokens used
            2. **co_occurence.npz** :- contains co_occurrence matrix
            3. **svd_matrix.txt** :- contains trained word_vectors
        ii. This file contains code to calculate word embeddings using Co-Occurrence Matrix by applying Singular Value Decomposition (SVD).
        iii. There are 2 sections in the file:-
            1. **Training**:- Contains code to train the embeddings.

2. **Inference**:- Contains code to test the trained model.

iv. To check the model performance, directly section 'Inference' can be run.

v. The notebook already contains the output shown for question 3 and 4 I.e. "Display the top-10 word vectors for 5 different words" and "the top 10 closest words for the word 'camera'".

vi. Output can be seen just by loading the script to jupyter notebooks.

c. Word2vec-cbow.ipynb:-

i. This notebook uses the following files:-

1. **tokenizer.pickle** :- contains tokenizer
2. **weights.best.hdf5** :- contained trained cboe model
3. **cbow_vecs.txt** :- contains generated word vectors

ii. This file contains code to calculate word embeddings using CBOW model.

iii. There are 2 sections in the file:-

1. **Training**:- Contains code to train the embeddings.
2. **Inference**:- Contains code to test the trained model.

iv. To check the model performance, directly section 'Inference' can be run.

v. The notebook already contains the output shown for question 3 and 4 I.e. "Display the top-10 word vectors for 5 different words" and "the top 10 closest words for the word 'camera'".

vi. Output can be seen just by loading the script to jupyter notebooks.

5. Results:-

a. The results for both the models (t-SNE plots) are saved in their respective notebooks. They can be visualized by loading the notebooks to jupyter notebooks.

6. Challenges faced:-

a. SVD_co-occurence.ipynb:-

i. First tried making co-occurene matrix by using simple for loop over the entire dump. Google colab keeps on crashing.

ii. Then tried removing the words with frequency less than 5 and applying the above method. Google colab keeps on crashing.

iii. Then removed stop-words as well. Google colab crashed again.

iv. Then tried scipy.sparse.csr_matrix. But colab still keeps crashing.

v. Tried some optimizations like removing punctuations and other. Still colab crahsed.

vi. Then tried sklearn CountVectorizer to make word count matrix on words with frequency >=5. This time colab didn't crashed.

vii. Then built co-occurrence matrix from count matrix. Saved it in a file. So that crashing colab doesn't lose the data.

viii. Then tried making SVD matrix from the co-occurrence matrix using scipy.linalg.svd. Colab crashed.

ix. Then tried making SVD using sklearn's TruncatedSVD. This time colab didn't get crashed.

x. The output embeddings are of shape (256, 1) which are saved in svd_mat.txt.

b. word2vec-cbow.ipynb:-

i. First tried trained the entire dump by writing model structure in tensorflow (using model.fit()). Kaggle kernel crashed.
ii. Then tried the words whose frequecy >=5. Kaggle kernel again crashed.
iii. Then used generator's concept to iteratively generate the training data. So that kernel doesn't get crashed. It worked by using model.fit_generator(). Trained on the entire dataset. But the results were not good.
iv. Then trained using on the words whose frequency >=5. Model's performance improved than the previous one.
v. The output embeddings are of shape (256, 1) which are saved in cbow_vecs.txt.