Assignment 3
Document Similarity

Name: Gopi Trinadh Maddikunta PSID: 2409404

Document is collection of text which either structured or unstructured which serves as fundamental unit in text processing and some other NLP techniques. Where document similarity is a method which is used to measure similarity between two or more text document and one of the main applications was search engines.

Let us dive into task 1 preprocessing, read input data into pandas dataframe and clean data by removing the records which are under “Tagline” and “Overview” columns expect we don’t remove the records even if it has one of the values. I have dataframe “data” which consists of “Title”, “Popularity” and “Full_Overview” where the third of “data” is the combination “Tagline” and “Overview” which is separated by space “ ”. To remove punctuation, symbols and special characters by performing regular expressions. Tokenizing the input documents and removing the stopwords by using SpaCy model ‘en_core_web_sm’ and SpaCy’s Stopword list respectively. There is choice of using lemmatization where I had chosen it because of meaningful representations, improve document similarity scores and Improves model efficiency by reducing unique words.

As result from task 1, Structured dataset with the processed text which is ready for NLP analysis.

Secondly, Task 2 – measuring the similarity using vectors, The main objective of this task is to create TF-IDF representation by using TF-IDF vectorizer function from scikit-learn to make vector embeddings for all words in document. Creating a function which performed cosine distance helps to measure similarity as more cosine distance means less similarity and vice versa. Cosine distance is calculated by using cosine similarity from scikit-learn. It will find out most similar movies to query movies and sort the top 5 movies based on popularity scores them in descending order.

Query Movies: “Taken”, “Pulp Fiction”, “Mad Max”, “Rain Man”, “Bruce Almighty”

Answers for the questions given on Task 2:

1.The number of unique words in vocabulary after transforming text into numerical form using TF-IDF. Vector Size of TF-IDF: 4800 documents × 24272 unique words.

2. The vocabulary size of TF-IDF vectors refers to the number of the unique words. TF-IDF Vocabulary Size: 24272 unique words

3. I agree with TF-IDF recommender system because it is used to measure the document similarity and recommend similar movies based on text description likewise it will be considered as content-based recommendation of movies. Occasionally, a few less-relevant picks appear, but they are not completely off.

Thirdly, Task 3 – Measuring similarity using Dense Vectors in two parts --- 3.1 first part, is for training dense vectors. Train a dense Word2Vec Vector representation of input vectors and set vector size by trail and error method which is ranges from 150 to 300 where I have take [150, 200, 250, 300] which randomly best selected and given window size which ranges from 7 to 13 for training, where I have set it has 10 which captures short range and longer range relationships where they are syntactic and semantic relationships, it was optimized by skip-gram to utilize the benefits for larger windows and when it comes to iterations ranges from 10 and 20, where I have taken iterations = 15 which more balanced. If I take iterations = 20 it may cause potential overfitting. I have create a function for calculating document similarity between query document and all nearly 4800 documents in "data" ("Full_Overview").

def find_similar_movies_dense(query_index, data, model, top_n=5)

Calculate cosine similarity between the dense vector representation of documents identified by the input index with all other documents and return to the top 5 with highest scores. Find the most similar movies to query movies and sort the top 5 similar documents using popularity scores.

```
Top 5 most similar movies to 'Taken':
                Title  Popularity  Similarity_Score
2114     Sliding Doors   15.639016          0.919980
3254  Virgin Territory    6.760922          0.921518
3513       Big Trouble    5.201688          0.923634
4392        The Hammer    0.957022          0.926005
4801        Alien Zone    0.000372          0.919205
```

Fig: 5 recommendations for 'Taken'

For Task 3.2 Using pretrained dense vector, In this I had used genism.downloader to download 'word2vec-google-news-300' which is pretrained model. Using similarity functions and pretrained embeddings, sort top 5 similar documents using popularity scores.

```
Top 5 most similar movies to 'Taken' using Pretrained Word2Vec:
                           Title  Popularity  Similarity_Score
1138                Analyze This   29.415229          0.854923
2114                Sliding Doors   15.639016          0.846045
3254             Virgin Territory    6.760922          0.847204
3466                 Soul Kitchen    5.461487          0.848808
4616  The Ghastly Love of Johnny X    0.209475          0.870943
```

Fig: 5 recommendations for 'Taken' using pretrained Word2Vec

Answers for the questions given on Task 3:

1. The word2vec based recommender system mostly provide fair recommendations and some picks are slightly off.

Query Movie	Recommended movies
Taken	Homefront, The Next Three Days, Kit Kittredge An American Girl, Darling Lili, We Have Your Husband
Pulp Fiction	Analyze This, Sliding Doors, Virgin Territory, Soul Kitchen, The Ghastly Love of Johnny X,
Mad Max	American History X, Legend, Hobo with a Shotgun, The Proposition, Defendor
Rain Man	The Beaver, Flirting with Disasterm, Pocketful of Miracles, Saint Ralph, The Blue Bird
Bruce Almighty	Superman, SpiderMan 2, Hellraiser, Krrish, Saint Ralph

Fig – Describes the Fair recommendations movies for every input movie.

2. Pretrained Word2Vec performed slightly better than trained once, as it was trained on a large dataset which would capture large word relationships where it trained model required tunning to good results.

3. Based on results, Pretrained Word2Vec dense vector takes first place based on semantic similarity and Task-Specific trained Word2Vec better than TF-IDF in semantic meaning which require hyper parameter tunning.

Lastly task 4 – Analysis required, I have used pretrained Word2Vec model which is 'word2vec-google-news-300'. I have created a function which takes input word analogy and calculates top 5 most similar movies. Here are the analogies: [google it]

king – man	woman	queen
doctor - man	woman	nurse
france – spain	madrid	barcelona
florida – texas	austin	miami
Table : Correct analogies resulted by browsing		

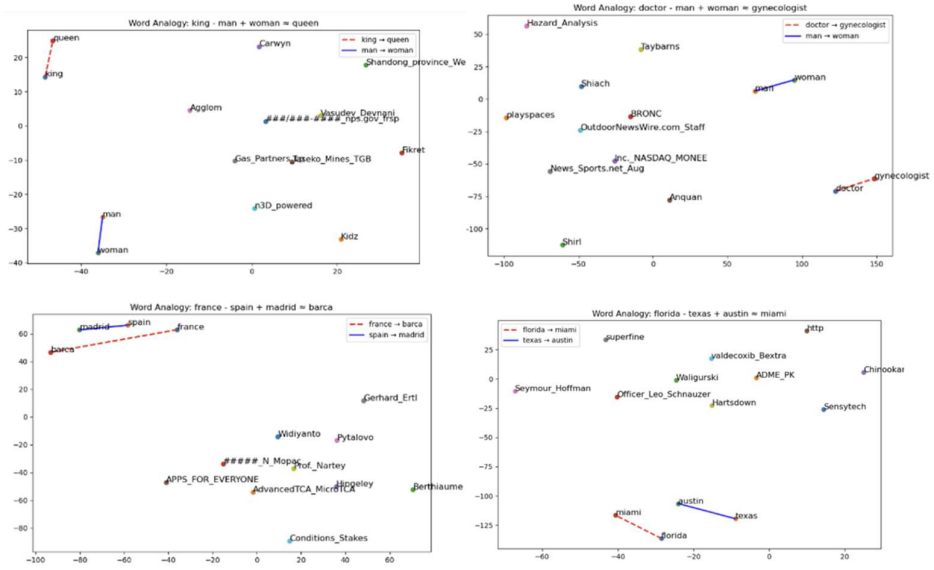

Fig : Plotting word analogies

Answers for Questions on Task 4:

1. No, all analogies are not correct but only first one and fourth one are correct "king – man + woman = queen", "florida – texas + austin = miami". I think the reason is limitations on vector arithmetic and lack of conceptual understanding.

2. I thought dimensionality reduction is main cause, because word embeddings are typically high dimensional, and another one may be randomly sampled words allow for an unbiased exploration.

3. Yes, Rerunning the t-SNE plot can change the alignment of points because t-SNE does not preserve distances perfectly and it is non-deterministic due to its stochastic nature.