MCA Assignment 3

Question1:

Algorithm explain-
Converting the embeddings into one hot encoding is sometimes not feasible when we have a large dataset. We use negative sampling in this case.

In word to vec approach the words are converted to vector forms so that they can be computed with other surrounding vectors.

The skip gram is used to get the surrounding words to a particular word.

ALGO

1. The words are taken in list format
2. A mapping to vectors is made.
3. We take a target word and get its n number of neighbours(those in n grams)
4. We make them in couple pairs.
5. We even give them labels. So there are both positive and negative samples. This is done to see that whether the couple is a right pair or not.
6. Embedding are made for the context as well as the target words.
7. The embeddings are of particular vector dimensions
8. Similarity is calculated. This is the cosine function. This means normalized dot product.
9. Then we apply the dense layer for the final output.
10. This is with sigmoid function.

Pickle file for the model has been uploaded

Results-
The loss function reduces and gets to a value of 0.64
The accuracy on reaches the values of 0.95

References-
https://adventuresinmachinelearning.com/word2vec-keras-tutorial/
https://towardsdatascience.com/an-implementation-guide-to-word2vec-using-numpy-and-google-sheets-13445eebd281
https://towardsdatascience.com/word2vec-from-scratch-with-numpy-8786ddd49e72

Question2 :

MAP score (Relevance feedback vector form): 0.715
MAP score (Relevance feedback query expansion): 0.723
MAP Score cosine similarity: 0.491

Iteration = 3

We observe that the MAP score of Relevance feedback with query expansion is better than the map score of relevance feedback of vector form. This is inline with what we expected.

I have used the ground truth to find the relevant files.

Process 1:
1. Finding the relevant and irrelevant files with the help of GT
2. Using the formula in notes

Process 2:
1. Finding the relevant and irrelevant files with the help of GT
2. Using the formula in notes
3. Finding the terms with highest tfidf. Taking the top 10 terms.
4. Adding the value of these terms in the the query

EXPLANATION:
Relevance feedback with query expansion performs the best which is followed by relevance feedback in vector form with is finally followed by cosine function.
The cosine function simply tells us the similarities between the query and the word. Based on that we give our prediction. But it does not change the queries in any way.
The relevance feedback in vector form adds the value of vector of relevant docs which increases the similarities of the queries and the docs.
The tfidf tells us how important/relevant a particular term is. Higher the tfidf value more important the term is. Adding the tfidf values of the most important terms improves the performance of the particular query. We add the tfidf of the relevant terms along with the relevant documents.