Sejal Bhalla
2017100

MCA Assignment - III

Q1: Word2vec

Data Used: NLTK *ABC* Corpus. It is a collection of news headlines from the Australian Broadcasting Commission.

Word Embeddings: word embedding is a technique to encode large sets of words or documents. The most basic and naive way to encode a set of words would be to use a one-hot encoding for each word but this method is highly inefficient. Therefore we use word embedding to "compress" large one-hot word vectors into much smaller vectors (a few hundred elements) which preserve some of the meaning and context of the word.

Word2Vec: It is a 2 layer neural network and the most common model which converts a text corpus into word embeddings. Word2Vec ensures that words which have similar context share meaning and that their representations are similar. The context here refers to the words surrounding a target word in a sentence, eg: the context of the word 'book' in the sentence 'The colour of the book is blue.' is ("The", "colour", "of", "the", "is", "blue").

Word2Vec has 2 variants: the **Skip-gram model** and the Continuous Bag of Words Model. For the purpose of this assignment, I've implemented the Skip-gram version wherein, given a target word, the model tries to predict the surrounding context words. Before we proceed to the training algorithm, each word in the corpus is assigned an index (0 to the length of corpus). The embedding layer takes this index as an input and returns a vector of size = 300.

Essentially the embedding layer is a table in which each row represents the embedding vector of a word. This table is updated in the process of training to make similar words have similar representations. A softmax layer after the output layer can then give the probability of each word being in the context of the input word. However, this is very time consuming and inefficient. Therefore, **Negative Sampling** is used. Under this approach, instead of a multiclass softmax layer, a simple binary classifier is used which gives an output of 1 if the word is in context of the target word and 0 otherwise. For every pair of words, the similarity score between the embedding vectors is calculated using cosine similarity and sent to the sigmoid layer which gives the final output.

Effectively, the algorithm works in the following manner:

1. Pairs of (target word, real context word) and (target word, negative context word) are made and sent as input.
2. Given an input of (target word, real/negative context word), the embedding layer gives the embeddings of these words.
3. The dot product of these 2 300-length vectors is calculated to obtain the similarity.

4. Finally, the similarity score obtained above is sent to a sigmoid layer which gives us an output of 1 or 0 which can be matched with the ground truth label given to the Context word (1 for a true context word, 0 for a negative sample).
5. In the process of training, the errors will be back-propagated and this will update the embedding layer to ensure that words which actually share meanings and contexts have vectors which return high similarity score.

2D and 3D Visualisations after every 5 epochs:

(**Note:** All the visualisations pertain to a representation of 10000 words and after every 5 epochs since T-SNE takes over 30 minutes to transform a vector of 31,000 words * 300 dimensions into a 2D space and training over multiple epochs was taking unreasonably high amount of time).

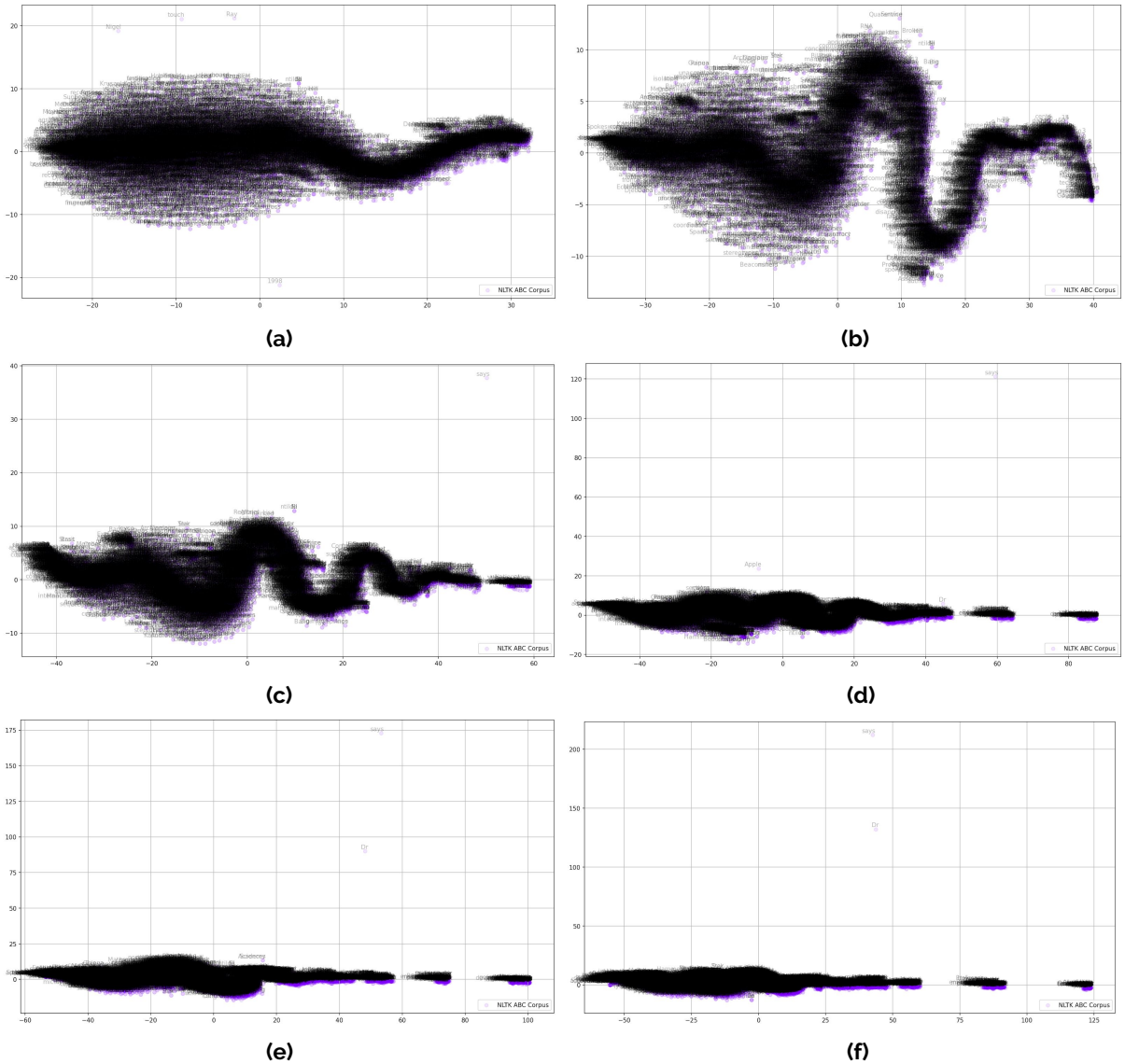

Fig 1. 2D Visual Embedding using TSNE over 25 epochs. (a) Before training, i.e 0th epoch. (b) After 5th epoch. (c) After 10th epoch. (d) After 15th epoch. (e) After 20th epoch. (f) After 25th epoch.

Sejal Bhalla
2017100

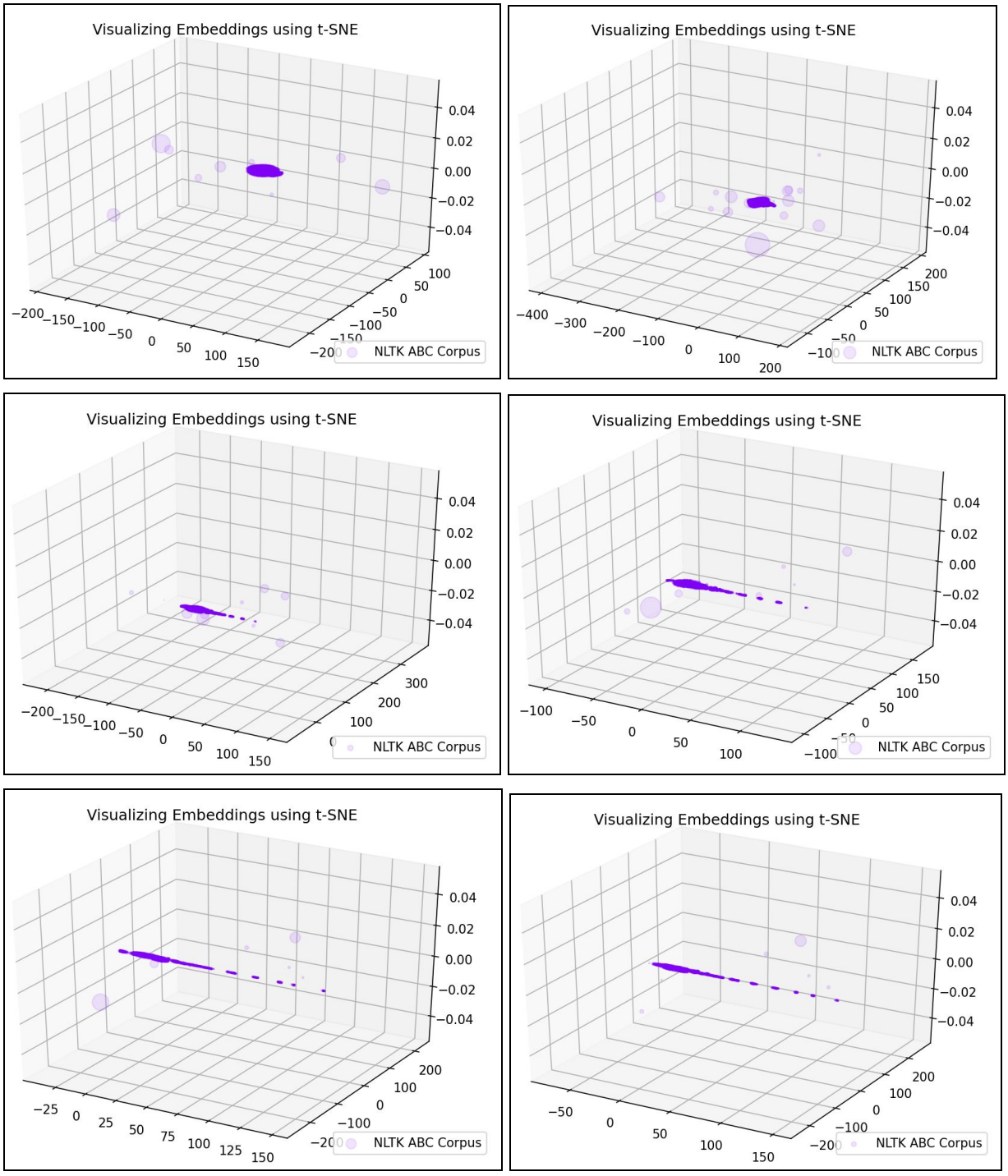

Fig 3. 3D Visual Embeddings using TSNE over 25 epochs.

Sejal Bhalla
2017100

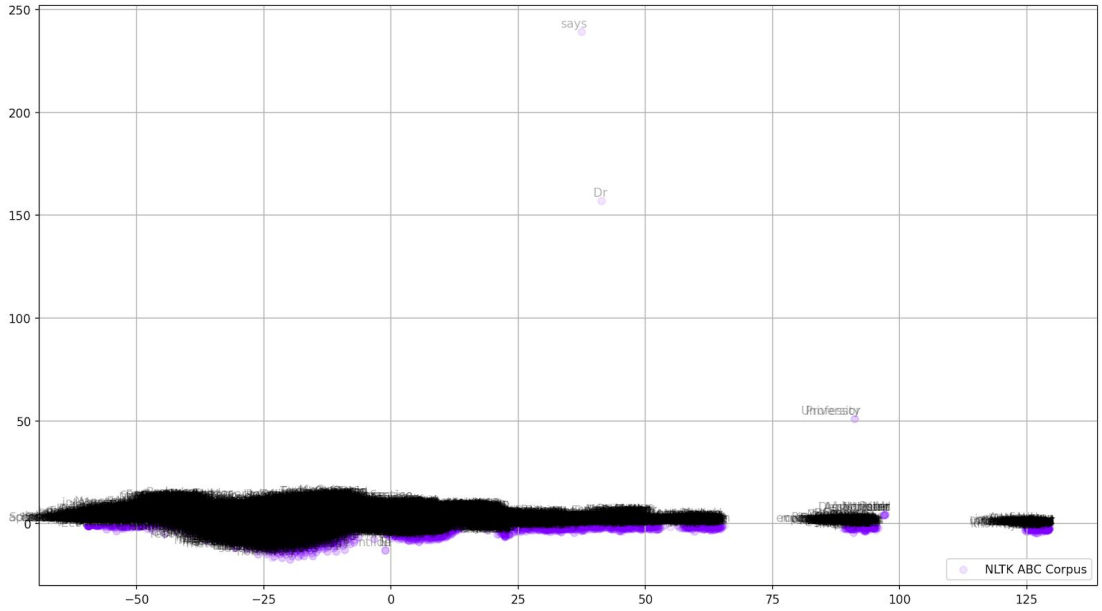

Fig2. 2D Visual Embedding after full training.

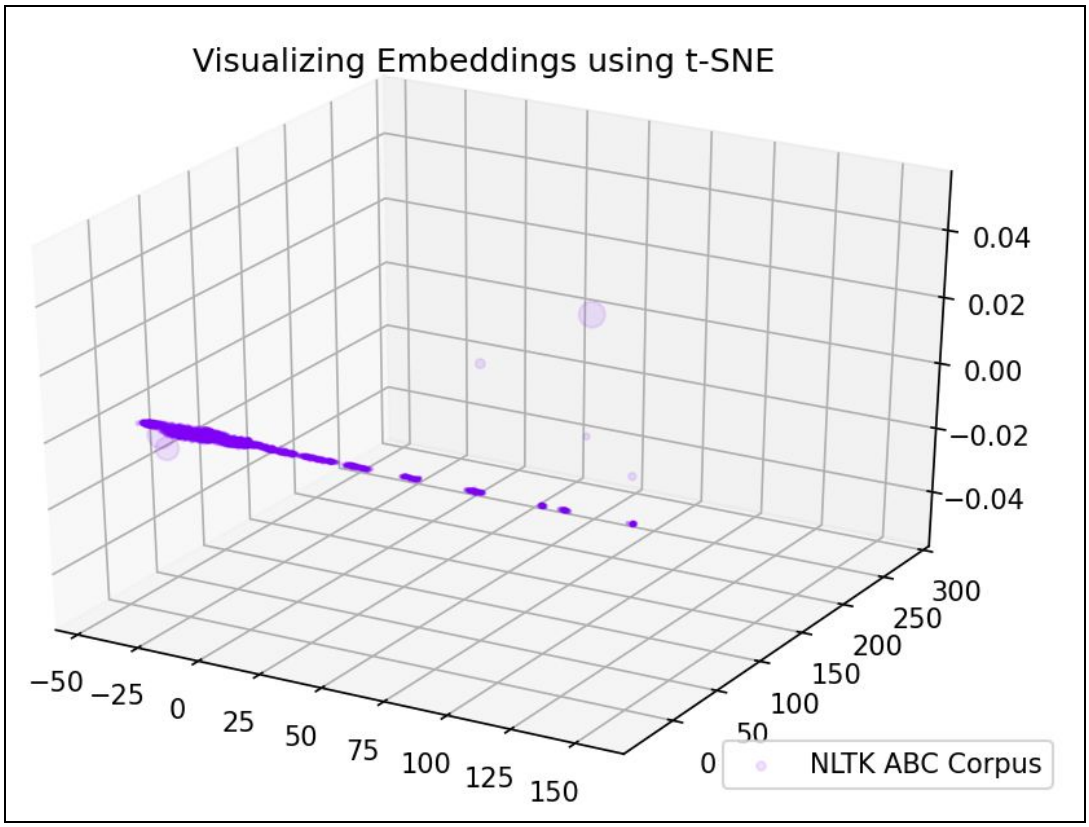

Fig 4. 3D Visual Embedding after full training.

Sejal Bhalla
2017100

Observations from the visualisations:

1. The 2D plots split and combine into clusters with an increase in training (number of epochs). In the initial epochs, we see a large change in their values. However, as the training progresses, they begin to collect and form close groups with a few points being outliers and not being a part of any cluster.
2. Since a higher dimension allows the words to be differentiated in an extra dimension, the 3D visualisations give us more visibly distinct clusters than the 2D visualisations.
3. After 30 epochs, we can observe 14 visible groups from 3D visualisation and 4 visible groups from 2D visualisation.

Results:
Loss after 30 epochs: **0.538**

Q2: Document Retrieval using Query Expansion

The objective of this task is to rank documents for a given query according to relevance. For this purpose, the query and the documents are transformed into vectors of the same size and a similarity score of these vectors is used to rank the documents. If we automatically declare the most similar documents as relevant and least similar as non-relevant and use them as implicit feedback, it is known as pseudo relevance feedback since manual input is eliminated from the process,

a) Relevance Feedback with Vector Adjustment only

The query vector needs to be adjusted in the following manner:

$$Q^{i+1} = Q^{i} + \alpha * \Sigma_{Di \, \varepsilon \, R} D_i - \beta * \Sigma_{Dj \, \varepsilon \, NR} D_j$$

Since positive feedback, i.e the indication of which document is more relevant, is more valuable, the values of alpha and beta were chosen in a way such that $\alpha > \beta$. After iterating over multiple values of α and β, the following values were chosen:

α = **0.8**
β = **0.6**

b) Relevance Feedback with Vector Adjustment and Query Extension

In this approach, vector adjustment is followed by adding the *n* most relevant terms in the query. A different set of α and β were chosen, with α = **0.6,** β = **0.4.**

Sejal Bhalla
2017100

Results and Analysis

1. Baseline Retrieval
   MAP: **0.49176786896815833**

2. Retrieval with Relevance Feedback
   MAP: **0.5889235774243075**

3. Retrieval with Relevance Feedback and query expansion
   MAP: **0.6320823493373311**

The result corresponds to the values of the parameters mentioned above and over 3 iterations.

In the context of this task, precision refers to the percentage of the results which are relevant and Recall refers to the percentage of the total relevant results which are correctly classified and present in the retrieved relevant documents.

The results are in line with what was expected. Relevance feedback with vector adjustment increased the MAP by almost 10%. This is because when we increase the weight of relevant documents and decrease the weight of non-relevant documents, we eventually increase the rank of similar relevant documents and decrease the rank of similar non-relevant documents. This gives us more number of actually relevant documents in the top-10, which increases the average precision.
On the other hand, Query expansion is often effective in increasing recall. Although not very likely, standalone query expansion might even decrease precision in certain cases by including terms which also occur in non-relevant documents quite frequently. However, the vector adjustment takes care of this. Overall, the two processes together increase the mean precision with a value greater than the precision obtained with just vector adjustment because the additional terms added to the query were selected from relevant documents. This will increase the similarity score between the query and relevant documents, giving them a better rank and increasing the precision.