MCA Assignment 3

Anish Madan, 2016223

March 2020

1 Q1 - Implementing Word2Vec

Word2Vec makes a representation of words in form of a vector which encodes word meaning based on context in the training corpus. We build a dense vector for each word type, chosen so that it is good at predicting other words in this context. There are 2 major ways to implement this algorithm:

1. Continuous Bag of Words model
2. Skip Gram Model

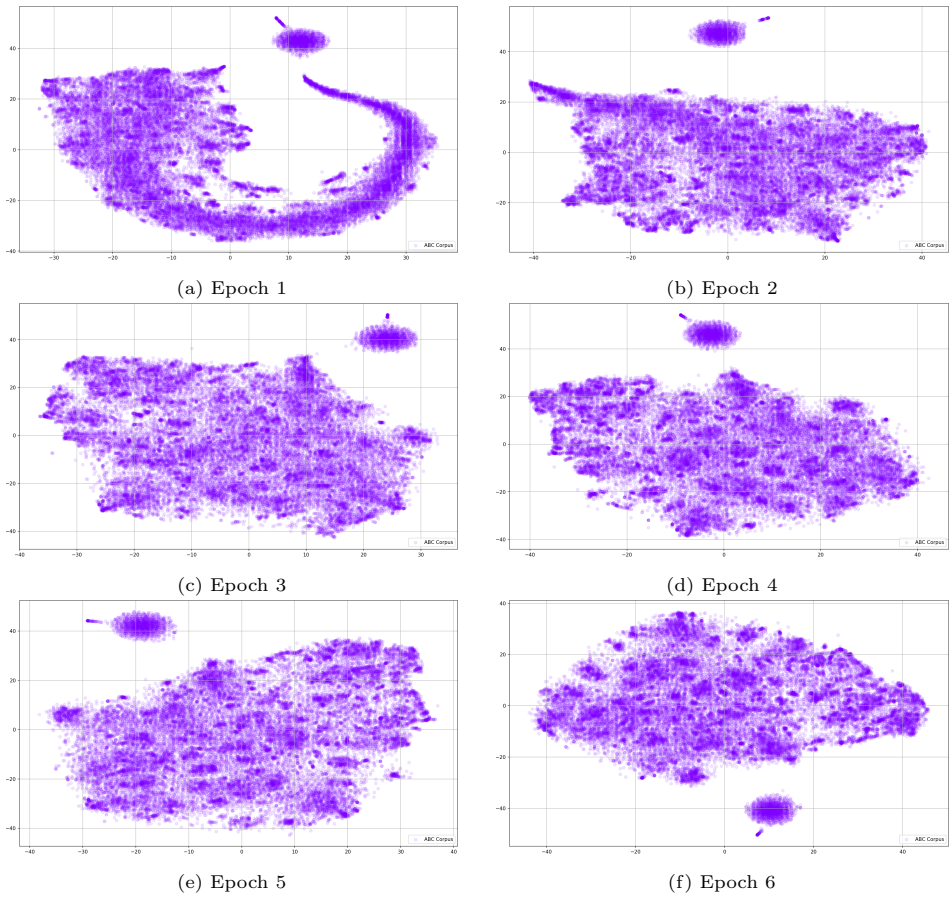
We choose the skip gram model based on its performance on medium-to-large datasets. It tries to predict the context words given the target word. We train this model for 10 epochs and visualize the first 5 as no significant change occurs after that. We use the Adam Optimizer in PyTorch with its default parameters and learning rate set as 1e-3 and a scheduler which decreases the learning rate if loss plateaus. The window size is fixed as 3 to make the dataset from the corpus. The hidden or ebedding size is 128 and our vocabulary size is roughly 31K. During preprocessing, we replace the numbers present in the text by keyword $\langle num \rangle$, so as to reduce vocabulary size. We visualize the results in figures 1, 2 and 3

2 Question 2

We tabulate our results in table 1. We observe that over epochs the performance increases in both scenarios which is expected as the queries refine over the epochs.

In relevance feedback we implement the pseudo relevance feedback, where we mimic the feedback process using the top relevant and top non relevant documents according to the similarity matrix. We update the original query by adding a term consisting of scaled sum of relevant docs and subtract a scaled sum of non relevant docs. This update helps us in refining the query and improves the MAP scores over a few epochs.

In the case of Relevance feedback and query expansion, we use the query generated from the above scenario and update its weights of the query. This



(a) Epoch 1

(b) Epoch 2

(c) Epoch 3

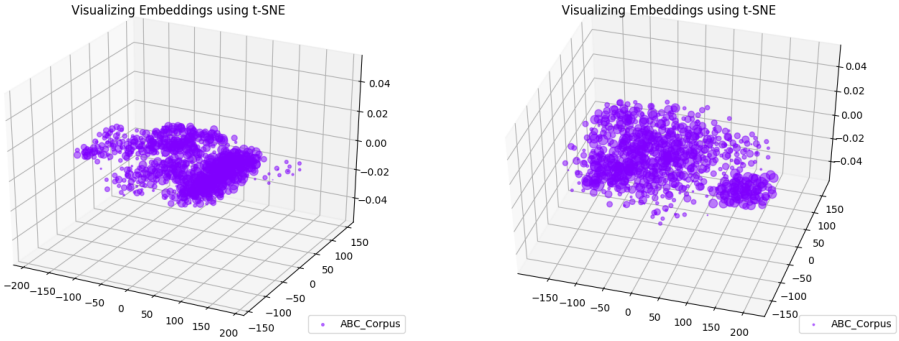
(d) Epoch 4

(e) Epoch 5

(f) Epoch 6

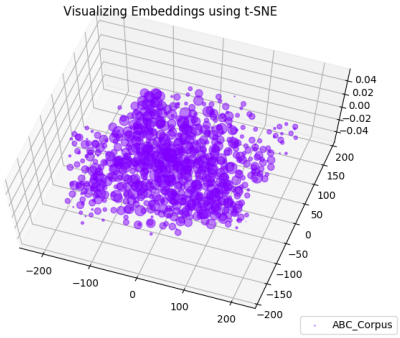
Figure 1: Visualizing embedding via TSNE in 2D. These were optimized using cuda , hence it was possible to visualize the whole dataset.

update process consists of finding the max value of tfidf in the query, and replacing the tfidf values of other words in the query by this value, on the condition that those words are similar to the max value one. This similarity is computed using an automatic thesaurus generated by computing cosine similarity between words. This updation process is referred to as query expansion, as we expand the query to include more words. Hence, this algorithm is expected to give better results as can be seen from the table 1

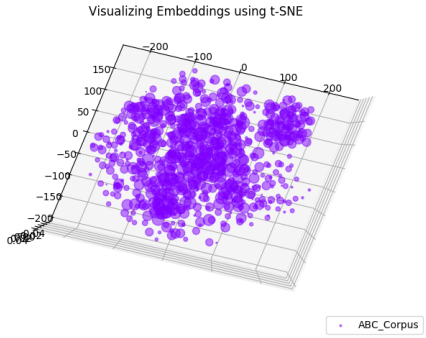


(a) Epoch 1

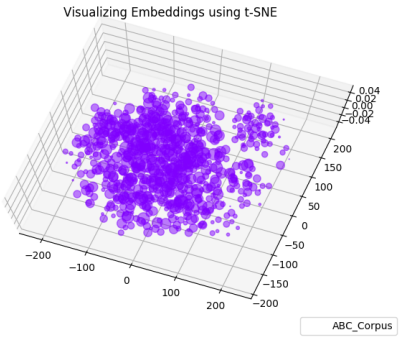
(b) Epoch 2



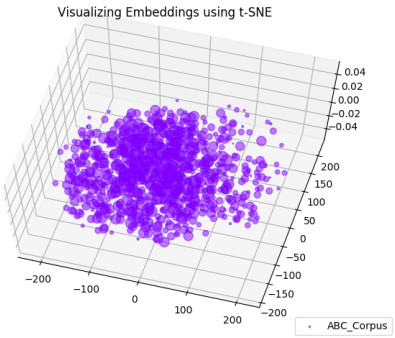
(c) Epoch 3



(d) Epoch 4



(e) Epoch 5



(f) Epoch 6

Figure 2: Visualizing embedding via TSNE in 3D. We visualie 1500 random words from the vocabulary here, as it was quite expensive to generate for the whole data.

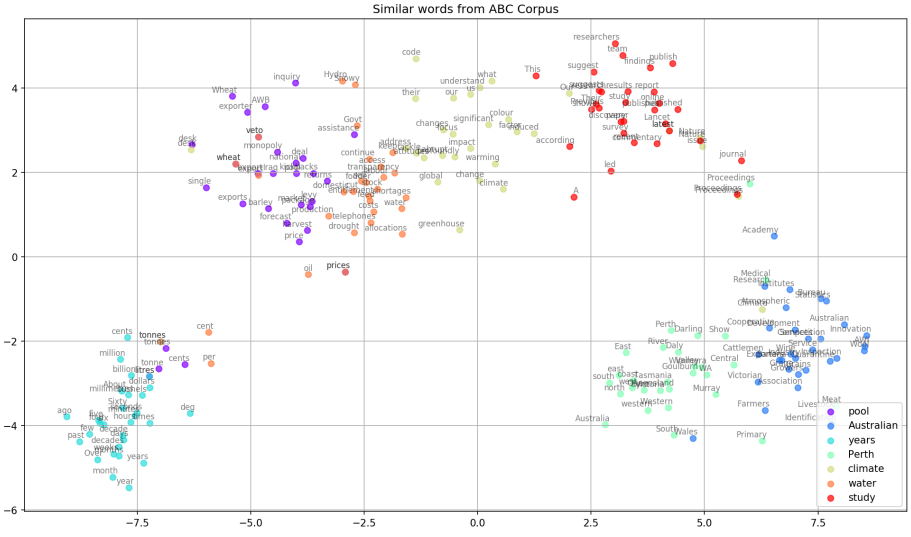


Figure 3: We select 7 words and visualize 30 most similar words to it using cosine similarity and plot those. Most of the similar words are clubbed together in the plot implying that model has been trained well.

Results obtained for Relevance Feedback		
Epoch	Relevance Feedback	Relevance Feedback+Query Expansion
1	60.36	61.74
2	60.57	61.99
3	60.57	62.14

Table 1: MAP scores for our 2 algorithms implemented.