Assignment 3

Suryatej Reddy (2016102)

May 2020

1 Word2Vec

1.1 Data Pre-Processing

I loaded the **ABC** dataset using a lazy loader of nltk. The loader returns sentences directly. I just parsed those sentences after removing punctuations and stop words.

1.2 Model

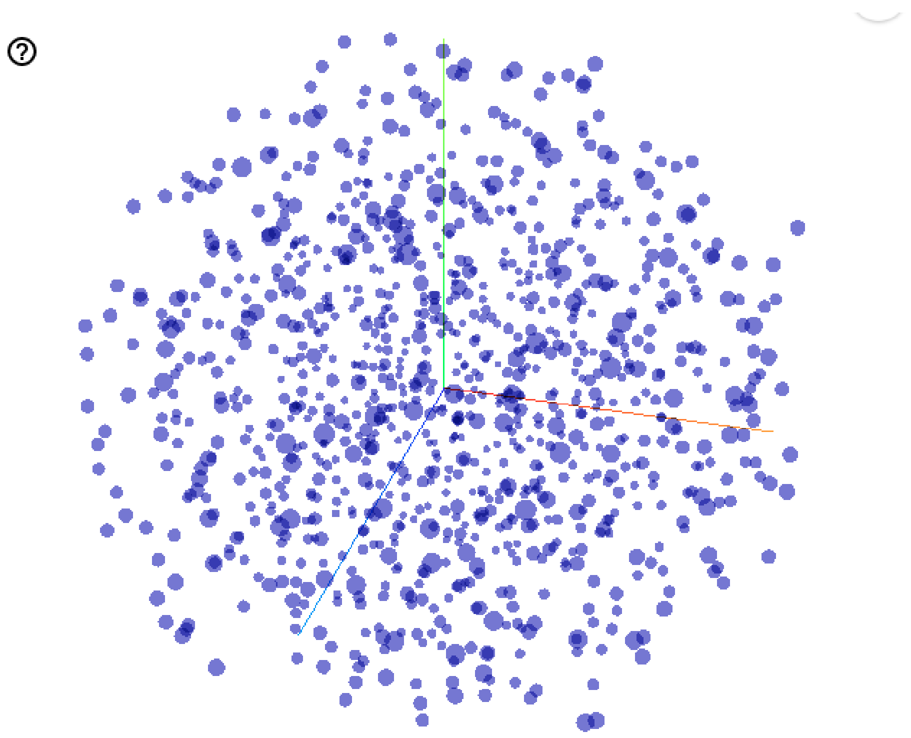
I implemented the **CBOW** version of the algorithm. I chose a context size of two. To the existing vocabulary, I added tokens *#START#* and *#END#* to denote the start and end of sentences. For CBOW, we need to give context vectors as input and then predict the word of interest. For example, consider the sentence *The cat is on the mat*.

For the word of interest *on*, we would have the contest words as *cat*, *is*, *on* and *the*. Basically two words before the and two words after. We pass these to a single layer neural network of dimensions 300 that learns the vector representation of each word.

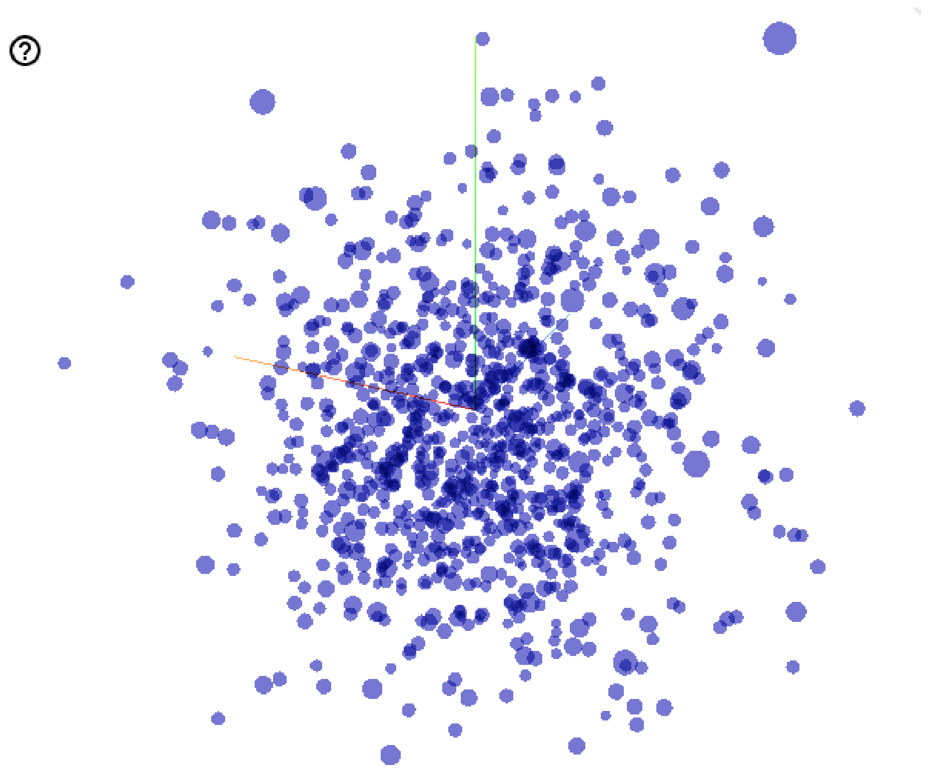
I used pytorch and the embedding layer that it offers to simplify the training process. I trained 3 epochs (3 lakh samples and vocabulary of 25000 words). I used tensorboard to visualise the results of subsampled words.

1.3 Results

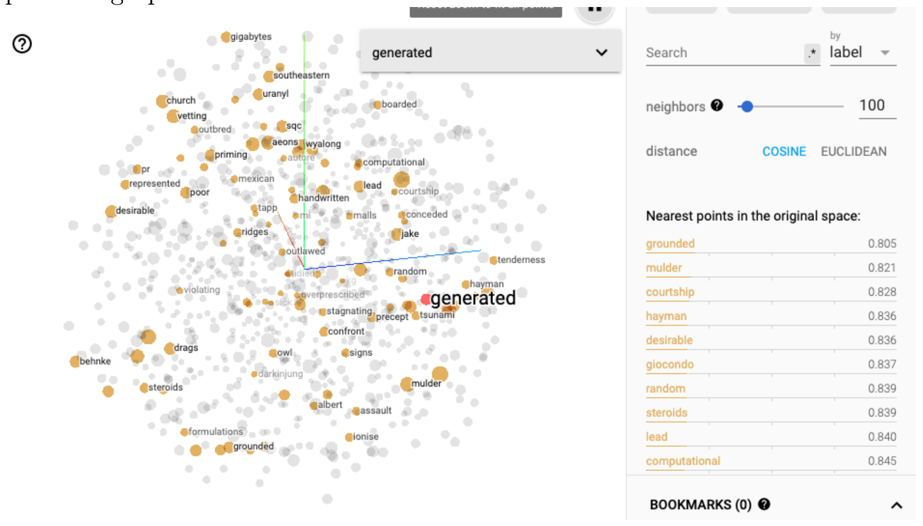
I captured the word embeddings after every 10000 training samples over 3 epochs). I loaded all these into tensorboard at once using another script that generated visualisations (TSNE and PCA) and also calculates similar words. I've attached the images in order of training.



This was after some iterations of the first epoch. The clusters took a lot of time to be processed using TSNE and are quite far apart. The words don't make sense with each other.



This was at the end of the first epoch. This time you can see that words begin getting clustered together. These are visualised using the same dimensions as the previous graphs.



generated

Search by label

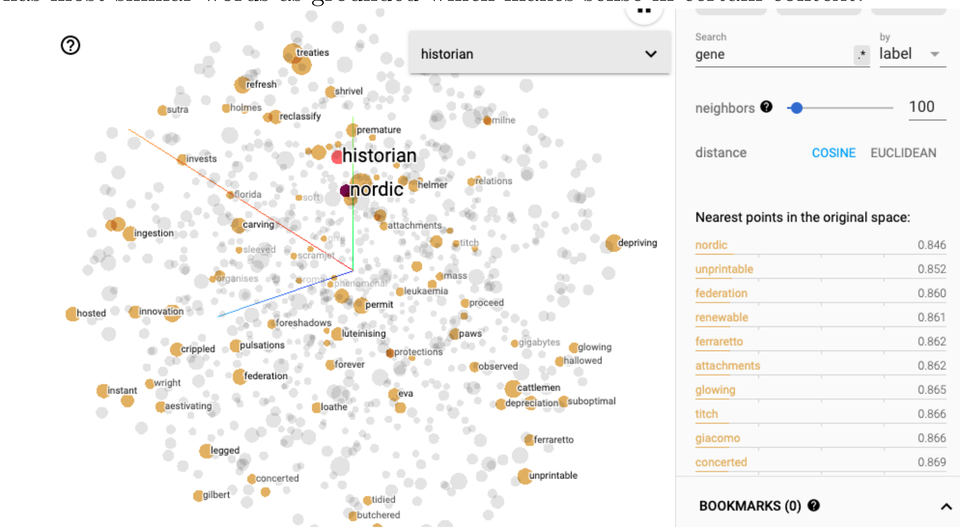
neighbors 100

distance COSINE EUCLIDEAN

Nearest points in the original space:

grounded	0.805
mulder	0.821
courtship	0.828
hayman	0.836
desirable	0.836
giocondo	0.837
random	0.839
steroids	0.839
lead	0.840
computational	0.845

This was at the beginning of the second epoch. There is some sort of semantic similarity forming between close words in the space. We can see that generated has most similar words as grounded which makes sense in certain context.



Search gene by label

neighbors 100

distance COSINE EUCLIDEAN

Nearest points in the original space:

nordic	0.846
unprintable	0.852
federation	0.860
renewable	0.861
ferraretto	0.862
attachments	0.862
glowing	0.865
titch	0.866
giacomo	0.866
concerted	0.869

BOOKMARKS (0)

This was at the end of the training. The similar words are close to each other with high similarity (0.85).

2 Retrieval

2.1 Relevance Feedback

Relevance feedback is a concept of retrieval systems that allows users to get better results based on one set of outputs. The standard implementation is an iterative process where users are shown results based on their query and then they can mark documents as relevant and non relevant. Based on this the results are then updated again to show the user what they want based on their marking.

2.2 Pseudo Relevance Feedback

In certain cases, we do not wish to take user marking as relevant and non relevant. So in such cases we take the top n retrieval results are relevant and apply the same algorithm again.

2.2.1 Implementation

I used the standard rocchio's algorithm for this step. Collected the most similar **n** documents as relevant and least similar as non relevant. Then I applied the rocchio update for 3 iterations.

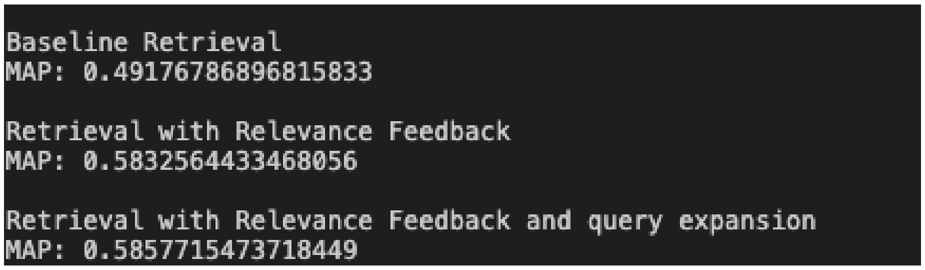
2.3 Query Expansion

In some cases, the user is provided with additional words in the query as suggestions to help them get better retrieval results. This is based on getting similar words of the query and suggest them to user.
Both Psudo Relevance Feedback and Query Expansion help in getting better recall.

2.3.1 Implementation

This step involves everything done in the above step and more. In my implementation, I found similar words of the query words using the vocabulary generated. Then I add these to the query with high value as if it is suggested to the user and they have selected these. I applied this step for 5 iterations.

2.4 Results



```
Baseline Retrieval
MAP: 0.49176786896815833

Retrieval with Relevance Feedback
MAP: 0.5832564433468056

Retrieval with Relevance Feedback and query expansion
MAP: 0.5857715473718449
```

These are the results I got. The retrieval results improve greatly with relevance feedback and show slight improvement with query expansion.