Assignment 3

-Sajag Agarwal, 2016185

1. **Algorithm:**

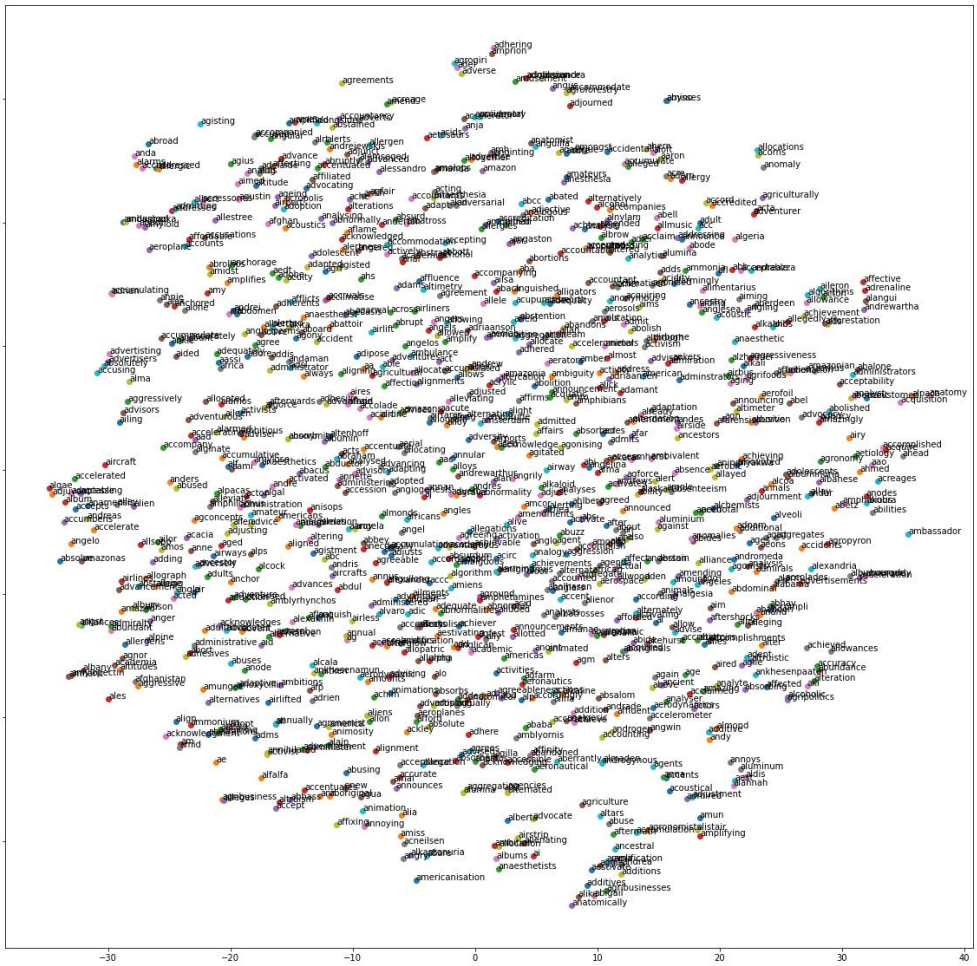
   First, the words are pre processed and then a dictionary is maintained for the unique words to convert them to integers and vice versa. Once done, subsampling is performed. The most frequently occuring words do not provide much context so they are removed. A word w_i is removed with probability = $1 - \sqrt{\frac{t}{f(w_i)}}$, where t is threshold and $f(w_i)$ represents the frequency of word in the dataset. The remaining unique words are now one hot encoded such that the length of each vector is the number of words in the dictionary and the vector is labelled 1 only at position corresponding to the word. The output is also the one hot encoded vector of the same shape as input and consists of probability of that word occurring near the input word. For training, labels are made by picking K words from the past and K words from the future of the current word and they are one hot encoded. Here, K is decided by the user. These training pairs are now fed to a neural network with a linear hidden layer consisting of a number of neurons decided by the user to represent a word in vector form. Hidden layer will be represented by a weight matrix of shape (number of unique words, number of neurons). The output layer uses a softmax layer. Since the size of weight matrices are huge, changing the whole matrix for just 1 sample is time consuming so the method utilises a method called negative sampling. Here, instead of tweaking the whole weight matrix, only a small part of the weight matrix is changed. The method selects a few random negative samples and tweaks weighst only for them along with the weights for the positive sample.

   For ‘abc’ corpus, 26876 unique words were observed. The number of neurons in hidden layer were chosen to be 10, number of weight rows to be tweaked were chosen to be 100 and following results were obtained for training process:

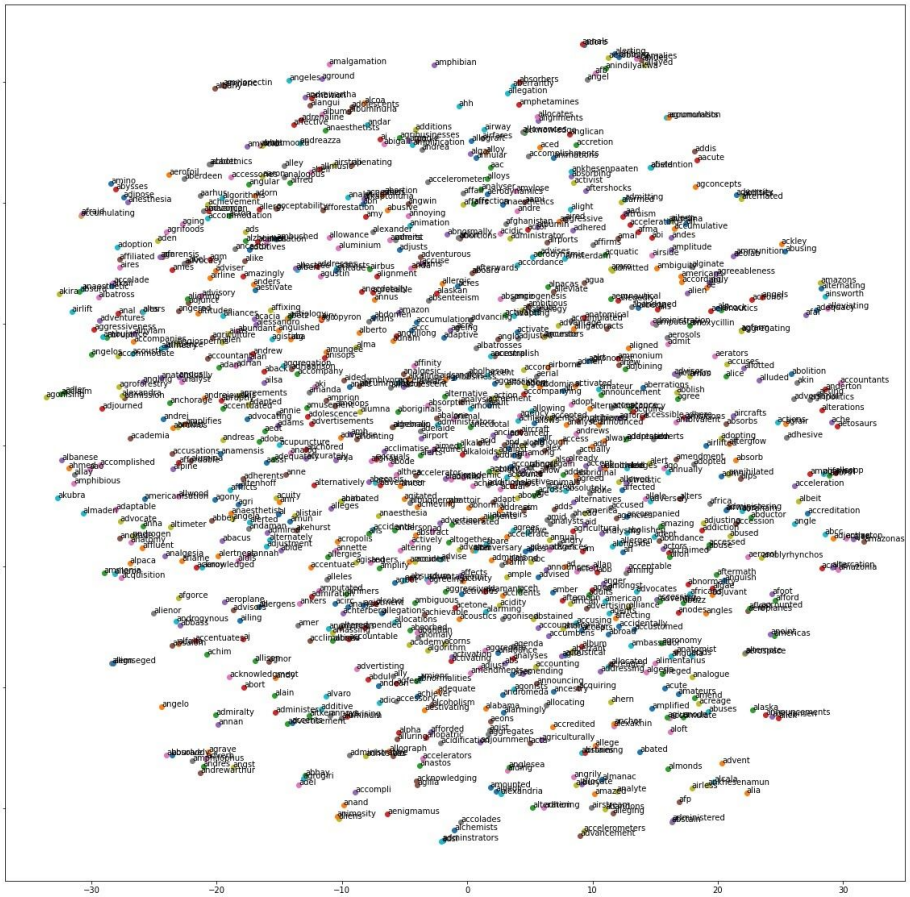
```
Epoch = 1 , Average Training Loss = 3.757390486845215
Epoch = 2 , Average Training Loss = 2.543695963956754
Epoch = 3 , Average Training Loss = 2.0261889680483267
Epoch = 4 , Average Training Loss = 1.8304423670896248
Epoch = 5 , Average Training Loss = 1.6876462212003445
Epoch = 6 , Average Training Loss = 1.6258178838687516
Epoch = 7 , Average Training Loss = 1.5744926588145562
Epoch = 8 , Average Training Loss = 1.542685405066468
Epoch = 9 , Average Training Loss = 1.541545047889925
Epoch = 10 , Average Training Loss = 1.5193894909982364
Epoch = 11 , Average Training Loss = 1.5063191810481844
Epoch = 12 , Average Training Loss = 1.472631295832808
Epoch = 13 , Average Training Loss = 1.455449325616136
Epoch = 14 , Average Training Loss = 1.4323937252359362
Epoch = 15 , Average Training Loss = 1.4408653198650219
Epoch = 16 , Average Training Loss = 1.449514618735501
Epoch = 17 , Average Training Loss = 1.4174606020587726
Epoch = 18 , Average Training Loss = 1.4362473231030761
Epoch = 19 , Average Training Loss = 1.4053213812267984
Epoch = 20 , Average Training Loss = 1.396053689459619
Epoch = 21 , Average Training Loss = 1.3928731272427515
Epoch = 22 , Average Training Loss = 1.4127448123891948
Epoch = 23 , Average Training Loss = 1.3881144265293714
Epoch = 24 , Average Training Loss = 1.3778478457817036
Epoch = 25 , Average Training Loss = 1.3806614980447425
Epoch = 26 , Average Training Loss = 1.3675852823780716
Epoch = 27 , Average Training Loss = 1.366775917372357
Epoch = 28 , Average Training Loss = 1.3770780981819997
Epoch = 29 , Average Training Loss = 1.3612189355158781
Epoch = 30 , Average Training Loss = 1.3753579716629505
Epoch = 31 , Average Training Loss = 1.3549397100293192
Epoch = 32 , Average Training Loss = 1.343201334162509
Epoch = 33 , Average Training Loss = 1.3259767010191494
Epoch = 34 , Average Training Loss = 1.3443302704485587
Epoch = 35 , Average Training Loss = 1.3456989248573599
Epoch = 36 , Average Training Loss = 1.3245499014072533
```

The tSNE plot for 1000 words embeddings for every 10th epoch (till 100 epochs) looks like:

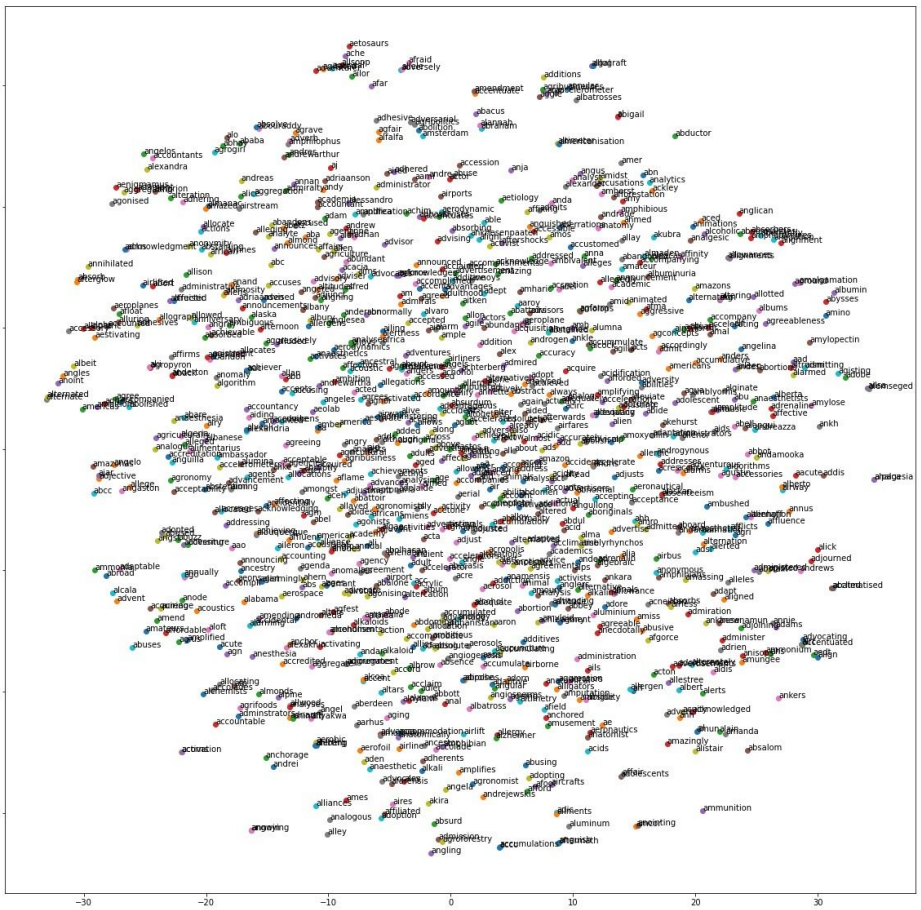
Epoch 1:



Epoch 11:



Epoch 21:



A gif ‘epochs10.gif’ consisting of the plot every 10th epoch and a zip folder ‘epochs’ containing image for all 100 epochs is attached.

2. **Results:**

Baseline Retrieval:

MAP: 0.5183859040856561

Time: 0.004139058458027885 sec

Retrieval with Relevance Feedback:

	MAP
α = 0.75 β = 0.15 Iterations = 10	0.666447928543613
α = 0.75 β = 0.15 Iterations = 3	0.661653916288426
α = 0.4 β = 0.15 Iterations = 3	0.605139746572950
α = 0.75 β = 0.3 Iterations = 3	0.604461558602521
α = 0.5 β = 0.5 Iterations = 3	0.451694543733037

Retrieval with Relevance Feedback and Query Expansion:

	MAP
α = 0.75 β = 0.15 Iterations = 10	0.863477829680270
α = 0.75 β = 0.15 Iterations = 3	0.861518273677807
α = 0.4	

β = 0.15 Iterations = 3	0.864890360291429
α = 0.75 β = 0.3 Iterations = 3	0.857302512410643
α = 0.5 β = 0.5 Iterations = 3	0.853849309342654

MAP witnesses a substantial increase by introducing a relevance feedback system and further increases by introducing query expansion. Since the relevance feedback system aims at finding the query vector that maximizes similarity with the relevant documents and minimize similarity with the non-relevant documents, these results were expected.
As the number of iterations are increased, keeping α and β constant, both the methods witness an increase in the MAP value but it is not much significant. Increasing β leads to reduction in MAP value for both the methods while decreasing α leads to increase in MAP value for relevance feedback + query expansion but a decrease for relevance feedback. The tuning of parameters and obtained MAP value shows that positive feedback is more important than negative feedback.

References:

1. Distributed Representations of Words and Phrases and their Compositionality: https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf
2. Visualizing Word2Vec Word Embeddings using t-SNE: https://towardsdatascience.com/google-news-and-leo-tolstoy-visualizing-word2vec-word-embeddings-with-t-sne-11558d8bd4d
3. Word2Vec (skip-gram model): https://towardsdatascience.com/word2vec-skip-gram-model-part-2-implementation-in-tf-7efdf6f58a27
4. Relevance Feedback and Query Expansion: https://www.cl.cam.ac.uk/teaching/1617/InfoRtrv/lecture7-relevance-feedback.pdf