MCA HW-3

Priysha (2016256)

7 May 2020

1 Question 1

1.1 Pre-processing of data

'abc' corpus from nltk library of python is used to train the Skip-gram model. Before being used, the words are cleaned to replace all capital letters, remove all occurrences of 0-9 numbers and punctuation marks. Moreover, all occurrences of stop words were deleted to increase the quality of word2vec vectors.

1.2 Data for training of the model

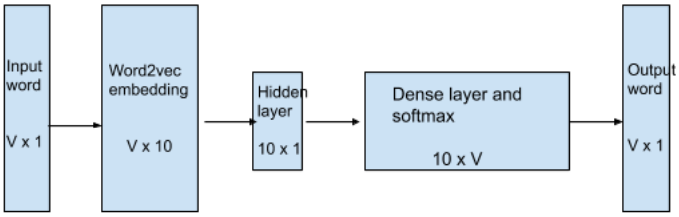
- Vocabulary and Word IDs: The cleaned words are now used to create a vocabulary of all unique words with unique ids.
- Context : Context of a particular word refers to its neighbouring words which occur across all sentences in the 'abc' corpus. A window of size 2 per word per sentence is taken to find context of each word in the vocabulary.
- Train_X for skipgram model : Hotencoded vectors of all words in the vocabulary.
- Train_Y for skipgram model : This vector is ground truth for the output of skipgram model to be compared to. The vector corresponding to each input word contains word IDs of all words in it's context.

Note that words in train_x are repeated such that one input word from train_x list gets mapped to exactly one context word from train_y list. Method for backward pass while training and increasing training speed has been taken from [5]. The article helped combine errors for each unique word to one vector for backpropogation and hence enhance training speed.

1.3 Training

- Input : Hotencoded vector per word. (size of vocab $\times$ 1)
- Output : A vector containing probabilities of different words being present in the context of input word. (size of vocab $\times$ 1)

- Target : A vector containing probability = 1 at words present in context of input word i.e. probability = 1 for word ids passed as train_y.
- Loss : Cross entropy loss[1]
- Model layers :



- Embedding size = 20 chosen
- Initialization : The initial word2vec embedding matrix is randomly initialized.

The model trains to update the matrix for 50 epochs while reducing cross entropy loss to finally return the final learnt word2vec embedding matrix.

1.4 Prediction of context and similar words

1.4.1 Predict context words

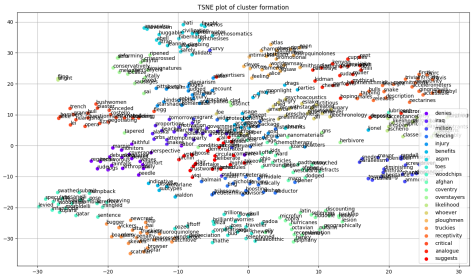
Given an input word, the softmax output from skipgram model can be sorted to get top 'n' most relevant context words.

1.4.2 Similar words

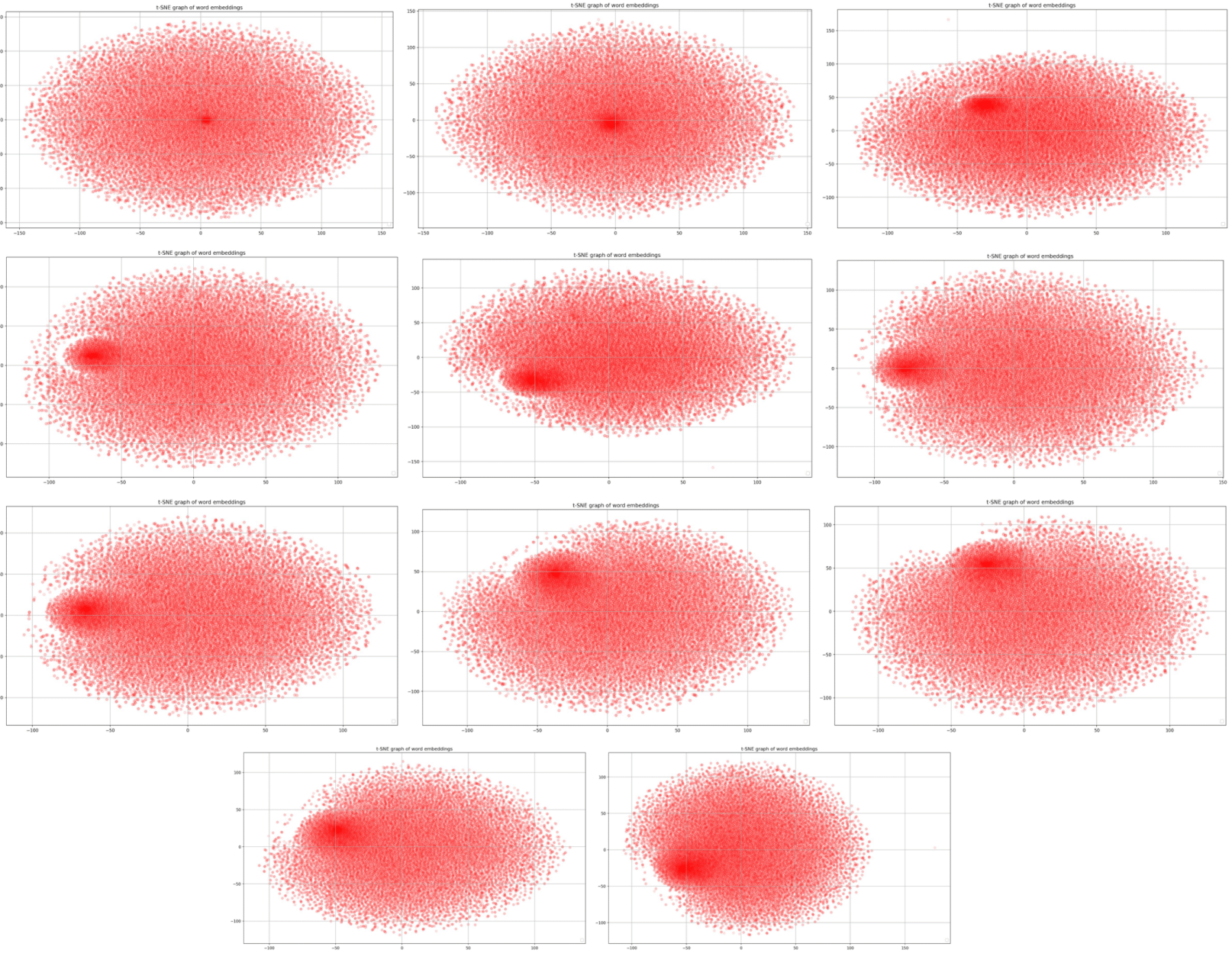
Given an input word, distance from each word in the vocabulary can be calculated using cosine distance formula [7]:

$$1 - \frac{u * v}{||u|| * ||v||}$$

Sort in ascending order using cosine distance to find most similar words.



1.5 TSNE Projections



The above figure represents word-embeddings of all 26743 words in vocabulary of 'abc' corpus, projected on a 2D plane using t-SNE projection[6]. Each sub graph is separated from the previous by 5 epochs. The total of 11 graphs shows how the randomly initialized word embedding matrix changes due to skipgram

traning. Word embeddings start to move closer to each other and size of the small cluster increases as number of epochs increase.

2 Question 2

2.1 Pseudo-Relevance feedback (PRF)

Given a query, the top n documents from cosine similarity are the output of baseline retrieval. Out of these n documents, some will be relevant results and others will be non-relevant.

- Relevant results : Those documents from the output which are also present in the ground truth "relevant documents" for that query are treated as relevant output documents.
- Non-relevant results : The remaining documents from the output which are not present in the ground truth "relevant documents" for that query are treated as non-relevant output documents.

If vec_docs is the tf-idf embedding of all documents, define :

$$d\_rel = \sum_{doc \in Rel} vec\_docs[doc]$$

$$d\_nonrel = \sum_{doc \in Non\_Rel} vec\_docs[doc]$$

For pseudo-relevance feedback using Rochhio's equation, alter tf-idf embedding of input query as :

$$Query = Query + \frac{\alpha}{|Rel|} * d\_rel - \frac{\beta}{|Non\_Rel|} * d\_nonrel$$

This equation aims to move our query vector towards the centroid of the relevant documents and away from the the centroid of non-relevant documents. Also, update the similarity matrix to contain the effect of new query vector.

Select $\alpha = 0.75$ and $\beta = 0.15$ [3, 4]. Typically, positive feedback has more role than negative feedback, so α is chosen to be greater than β.

Any negative occurrences in the final query vector are converted to 0, in accordance to the Rochhio algorithm[3, 4]

The above process can be repeated over multiple iterations to improve retrieval results.

2.2 Results

Comparing MAP scores :

Algorithm	MAP score
Baseline Retrieval	0.4917
PRF($\alpha = 0.75, \beta = 0.15, iter = 1$)	0.6371
PRF($\alpha = 0.75, \beta = 0.15, iter = 10$)	0.7391
PRF($\alpha = 0.75, \beta = 0.15, iter = 15$)	0.7402
PRF($\alpha = 0.75, \beta = 0.15, iter = 50$)	0.7390

As expected, more number of iterations help adjust the query vector more taking into affect the relevant documents, hence the MAP score increases with number of iterations increasing from 1 to 10 to 15. However, when number of iterations is increased to a very high number (50), the MAP score starts to decrease. One of the reasons behind this decrease can be absence of any sort of normalization in the query updation process.

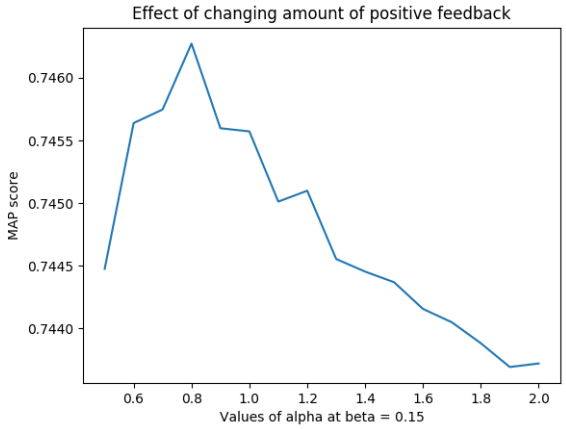
2.2.1 When no positive feedback i.e. $\alpha = 0.0$

At $\alpha = 0.0$ and $\beta \neq 0.0$, the Rochhio equation only takes effect of negative feedback from non-relevant documents. As a result, the relevance feedback algorithm learns which documents not to produce but does not learn which documents to actually produce in the output.

Hence, although the MAP score rises as compared to baseline retrieval but not much.

$$\textbf{MAP = 0.5885; } (\alpha = \textbf{0.0}, \beta = \textbf{0.15})$$

2.2.2 Increasing amount of positive feedback



As α starts increasing, the MAP score also increases till a threshold of α. After that, on further increasing α, the MAP score starts to decrease. There can be two possible reasons for the same :

- Increasing alpha beyond a certain number increases the difference between α and β values. Amount of negative feedback is not growing proportionate

to amount of positive feedback. When positive feedback is proportionately much bigger than the negative one, a situation similar "no negative feedback" occurs and hence MAP decreases.

- As already stated, greater α value leads to a bigger vector being added to query vector. Absence of any normalization operation can be another reason for decrease in the MAP score.

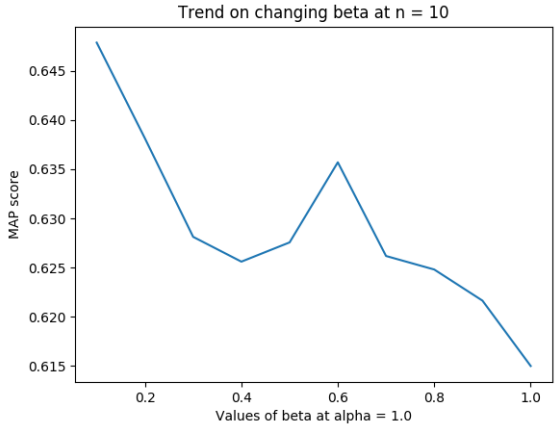
2.2.3 When no negative feedback i.e. $\beta = 0.0$

At $\beta = 0.0$ and $\alpha \neq 0.0$, the Rochhio equation only takes effect of positive feedback from relevant documents. As a result, the relevance feedback algorithm learns which documents to produce but does not learn which documents not to produce in the output.

Although, due to the positive feedback, high number of relevant documents are included in the output and MAP score rises high above the Baseline Retrieval score, but not as high as case of including both negative and positive feedback.

$$\textbf{MAP} = \textbf{0.6942; } (\alpha = \textbf{0.75, } \beta = \textbf{0.0})$$

2.2.4 Increasing amount of negative feedback

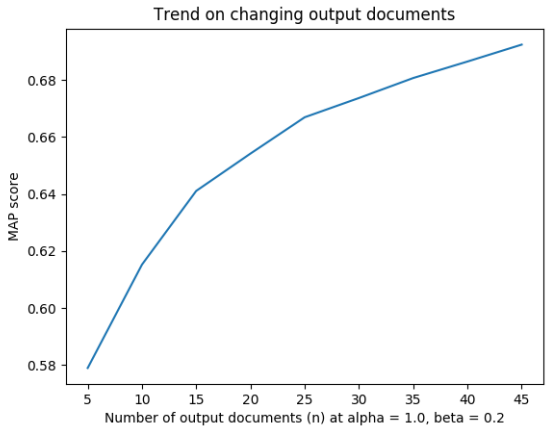


As β starts increasing, the MAP score starts decreasing. There can be two possible reasons for the same :

- Increasing β beyond a certain number increases the amount of negative feedback and it starts growing out of proportion to amount of positive feedback. When negative feedback is highly weighted, a situation similar "no/less positive feedback" occurs and hence MAP decreases.

- As already stated, absence of any normalization operation can be another reason for decrease in the MAP score.

2.2.5 Increasing number of documents retrieved from the base line



As number of documents initially retrieved increase, there are chances of more documents being relevant. Hence, the MAP score continues to increase at fized α and β.

2.3 Pseudo-Relevance feedback with Query extension

After a run of pseudo relevance feedback algorithm, the query vector can be further altered by appending more words to it which are similar to the words present in the original query. The method chosen to do this is as follows:

- Assume that given a query, words which occur frequently or that have high tf-idf score in its corresponding relevant documents in the ground truth are the ones which are most similar to the words in input query.
- For each input query, take 10 relevant documents from its ground truth.
- For each chosen document, extract top 10 words with highest tf-idf values in that document. Totalling to 100, these words are similar to the input query and need to be added to its vector.
- For each of the top 10 words in each relevant document, add its tf-idf score to the relevant index in the query vector.
- Repeat above steps for all queries.

The algorithm results in an extended version of the input queries utilizing the idea of query extension with synonym/similar words.

All of the above steps can be repeated for more number of iterations for better results.

2.3.1 Results

Comparing MAP scores :

Algorithm	MAP score
Baseline Retrieval	0.4917
PRF+QE($\alpha = 0.75, \beta = 0.15, iter = 1$)	0.8387
PRF+QE($\alpha = 0.75, \beta = 0.15, iter = 10$)	0.8401
PRF+QE($\alpha = 0.75, \beta = 0.15, iter = 50$)	0.8407

Clearly, query extension algorithm gives significantly better results than pseudo-relevance feedback.

Note that, taking all documents from the ground truth as relevant rather than taking first 10 will result in higher MAP score but at the cost of higher computational time.

References

[1] Word2Vec - Towards Data Science

[2] Word2Vec - Geeks for geeks

[3] Rochhio, relevance feedback : PDF, Stanford University

[4] Rochhio algorithm - PDF, University of Cambridge

[5] Word2Vec - Towards Data Science

[6] t-SNE - Towards Data Science

[7] Cosine Distance - Scipy