MCAAssignment3

Kshitiz Jain (2016051)

May 2020

1 Question1

1.1 Data Pre-processing

The *nltkabc* corpus gives a matrix with each row as a sentence. It contains various irrelevant information and is removed. Following things are removed from the corpus and a single array of words is generated.

- Punctuation
- Stop words
- Numbers
- Uppercase letter are converted to lowercase

1.2 Training Data generation

Given a array of words and context limit, training data is generated. Context limit represents upto which word the context can be retrieved of a given word i.e. if the context limit is 2 the 2 words before and 2 words after the target word will be used for retrieving the context of the taget word.

Hot encoding is done on the input array and every word is given a id which is their respective index in the hot encoded vector.

Training data is in the for form : $[< target\_word >, < context\_words >]$. Target word is generated by hot encoding only the target word in the vector. While the context word each row represents a hot encoded vector of target_word's context words.

1.3 Training network

A neural network is created with 1 hidden layer to learn the embedding of the corpus. The input layer is of size equal to the vocabulary size, embedding size can be varied (more the size more the context can be learned, but complexity increases), and output layer with with size equal to vocabulary. Softmax layer is applied on the output layer to get the probabilities of word being related to other words.

Error vector is calculated by summing the difference between the softmax output and each of the context words, and that error is back propagated.

Cross-entropy (sum of log of output probabilities at index of context words) loss is measured for every epoch.

1.4 Optimization while training

As the context words are only used to calculate the error after the forward pass and for each word there are around 20 context words, they take a huge space for storing all the hot encoded vectors.

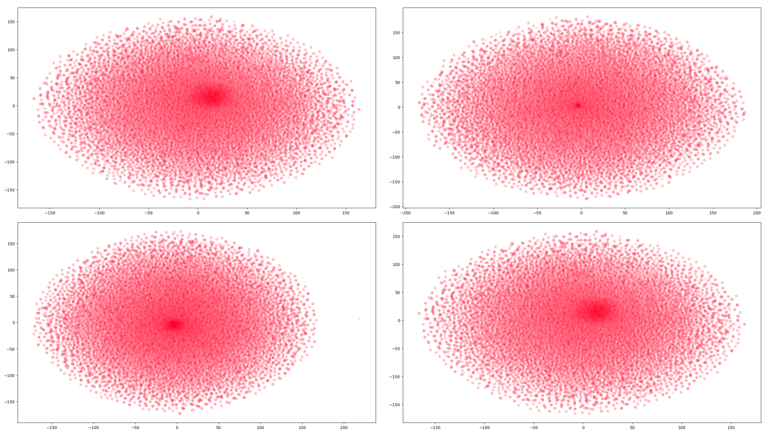
Therefore, while generating the training data, instead of making hot-encoded vectors for each context words only ids of the context words are saved which can be used to calculate the error in training.

1.5 Prediction of Words

Weights of the trained Neural Network (between input and hidden layer) is used as the embedding for all the words. To find similar words for a given word, first it's embedding vector is retrieved from the weight matrix and then for using cosine similarity all words embedding are compared and top k closest words are returned.

1.6 Embedding Plots

The given plots are generated from the weight matrices trained with 25 embeddings. Weights were saved after every 4th epoch and is trained for 20 epochs.



It can be observed as the training progresses, the cluster starts to form a

2 Question2

2.1 Relevance Feedback

Rocchio algorithm is used to for implementing the Relevance feedback algorithm. Following are the steps which are used for implementing the same.

1. For every query, top 10 documents are retrieved from the database by using the similarity matrix.

2. The retrieved documents are then divided into relevant and non-relevant documents. Document is relevant only if it belongs in the ground truth

3. Then for both relevant and non-relevant documents a mean vector is computed, one which represents mean of relevant documents and one which represents mean of non-relevant documents.

4. Then Query is updated using the following Rocchio formula :

$$\vec{q}_m = \alpha\vec{q}_0 + \beta\frac{1}{|C_r|}\sum_{\vec{d}_j \in C_r}\vec{d}_j - \gamma\frac{1}{|C_{nr}|}\sum_{\vec{d}_j \in C_{nr}}\vec{d}_j$$

   Alpha ,Beta and gamma is taken as 1, 0.75 and 0.15 respectively. Alpha, Beta and Gamma represents the weight we want to give each of the feedback. Higher the beta value, higher the weight-age to the positive feedback from the user; similarly higher the gamma value, higher the weight-age of negative feedback. Helps in shifting the query's centroid towards the relevant documents, and shifting away from non-relevant documents.

5. Now after every query is updated, similarity matrix is updated using the new query vectors and documents vectors.

6. The above process can be repeated multiple times. In the implementation it is repeated 10 times.

2.2 Relevance Feedback and Query Extension

For query extension, following procedure is used along with the Rocchio method.

1. For every query vector, it is first updated by the Rocchio algorithm (step 4 above).

2. Now, all the relevant docs (present in the ground truth) are retrieved.

3. For all the relevant documents top n words (highest tf-idf values) are retrieved.

4. Then these words are added in the given query, by adding their tf-idf values in the vector at their respective positions.

2.3 Results and inferences

MAP Scores		
Iteration	**RelevanceFeedBack**	**RelevanceFeedBack with QueryExtension**
0(baseline)	0.4917	0.4917
1	0.6371	0.8814
2	0.7005	0.8876
3	0.7204	0.8891

As expected, Relevance feedback with query extension always out performs the relevance feedback. Since in query extension we add the important words of the relevant document in the query and make is more similar to the relevant documents.