Study of Word Embedding in Supervised and Unsupervised Tasks on Quora Dataset

Reshma Sri Challa
Computer Science
University of Ottawa
Ottawa, Canada
rchal050@uottawa.ca

Raj Kumar Endla
Computer Science
University of Ottawa
Ottawa, Canada
rendl026@uottawa.ca

***Abstract*— NLP based feature learning technique, Word Embeddings maps words into vectors of real numbers that capture the context of the underlying words in relation to other words in the sentence. This conversion outcomes in words that can be depicted as a vector in an n-dimensional vector space and the distance between two such vectors depict the level of resemblance between these words. The recent considerable progress in the quantity of easily accessible Quora text has got to the foreground the necessity for large-scale natural language processing tools for text mining. In this project, a study of several word embeddings including Word2Vec, FastText, Glove, SBERT, and Doc2Vec is conducted for supervised and unsupervised tasks on Quora dataset. Supervised task is to identify duplicate questions in the dataset through binary classification, while unsupervised task is clustering Quora questions.**

Keywords—Word Embeddings, Word2Vec, FastText, Glove, SBERT, Doc2Vec, Supervised Task, Unsupervised Task, Clustering.

I. INTRODUCTION

Unstructured data like text, images, videos comprise a fortune of information. But, due to the complication in processing and analyzing this data, people frequently hold back from spending additional time and effort in taking a chance out from structured datasets to examine these unstructured resources of data, which can be a hidden gold mine. Natural Language Processing (NLP) is all about leveraging tools, practices and procedures to process and understand natural language-based data, which is normally unstructured.

NLP based feature learning technique, Word Embeddings maps words into vectors of real numbers that capture the context of the underlying words in relation to other words in the sentence. This conversion outcomes in words that can be depicted as a vector in an n-dimensional vector space and the distance between two such vectors depict the level of resemblance between these words. This is compared to the thousands of dimensions necessary for sparse word representations, like one-hot encoding. For instance, we can embed the words "effects" and "taking" as dimensional vectors as [0.8 1.0 4.2 7.5 3.6] , [8.3 5.7 7.8 4.6 2.5] respectively.

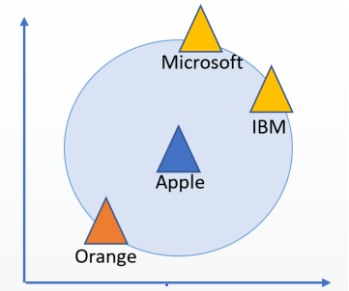

Figure 1: Representation of word vectors

For example, consider the figure 1 shows the representation of words from two categories fruits and companies. Words "IBM", "Microsoft", "Apple", "Orange" are taken. The words IBM and Microsoft are closer as these both are company names which is the similarity between these two words and orange is located further away as this word is not similar to company names, as it is a fruit. The keep focus here should be on "apple" as this can be a fruit or a company, this has similarity with both orange and IBM, Microsoft so it is located in between the two categories in dimensional space.

The word embeddings used for this project are Glove (pretrained on Twitter, common-crawl-840B), Word2Vec (pretrained on Google-News), FastText (pretrained on Wiki-news), Doc2Vec (trained on Quora Dataset) and Siamese BERT-Networks (SBERT). Along with these word embeddings we have also tried with Term Frequency-Inverse Document Frequency (TF-IDF). Now let's discuss how each of these word embeddings work.

1. **Word2Vec**: there are 2 types of model architectures:

    1.1. CBOW: continuous bag of words, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction (as name says it is bag of words assumption) [5]

    1.2. Skip-gram: continuous skip-gram architecture, the model uses the current word to predict the surrounding window of context words. The skip-

gram architecture weighs nearby context words more heavily than more distant context words. CBOW is faster while skip-gram is slower but does a better job for infrequent words.[4]

2. **Glove**: Global Vectors is developed as an open source project at Stanford, which is a model for distributed word representation, model is an unsupervised learning algorithm, training is performed on aggregated global word-word cooccurrence statistics from a large corpus. Glove combines features of both global matrix featurization and local context window methods.[6]

3. **FastText:** This is an extension to word2vec.Instead of feeding individual words into the Neural Network, FastText breaks words into several n-grams (sub-words). For instance, the tri-grams for the word *apple* is *app, ppl*, and *ple* (ignoring the starting and ending of boundaries of words). The word embedding vector for *apple* will be the sum of all these n-grams. After training the Neural Network, will have word embeddings for all the n-grams given the training dataset. Rare words can be properly represented since it is highly likely that some of their n-grams also appears in other words.[7]

4. **Doc2Vec:** Doc2Vec is an adaptation of word2vec which can generate vectors for words. The aim of doc2vec is to build a numeric representation of a document/sentences/paragraphs, irrespective of its length. Word2Vec calculates a feature vector for each word in the corpus but Doc2Vec calculates a feature vector for each document in the corpus. Word2Vec functions on the instinct that the word representation should be sufficient to predict the adjacent words, the underlying instinct of Doc2Vec is that the document representation should be sufficient to predict the words in the document.

5. **BERT**: Bidirectional Encoder Representations from Transformers models were pre-trained using a large corpus of sentences. In brief, the training is done by masking a few words (~15% of the words according to the authors of the paper) in a sentence and tasking the model to predict the masked words. And as the model trains to predict, it learns to produce a powerful internal representation of words as word embedding.[2]

6. **SBERT:** The idea here is to Fine-tuning BERT to give good Sentence Embeddings. This is orders of magnitude better than having to pass in each pair of sentences through BERT. This is the current state of the art. The general idea introduced is to pass 2 sentences through BERT, in a Siamese fashion. The idea is simple. SBERT adds a pooling operation to the output of pretrained BERT word embeddings to derive a fixed sized sentence embedding. The pooling strategy is Mean strategy. We then *concatenate* the embeddings as follows: (u, v, ‖u-v‖), multiply by a trainable weight matrix $W \in \mathbb{R}^{3N \times K}$, where N is the sentence embedding dimension, and K is the number of labels.[3]

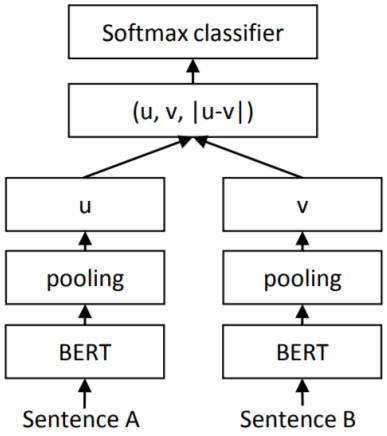

Figure 2: SBERT architecture for Classification -Task 1

7. **TF-IDF**:A popular way to signify each document in a corpus is to use the TF-IDF statistic (term frequency-inverse document frequency) for each word, which is a weighting factor that can be used instead of binary or word count representations. The concept behind the TF-IDF method is that the words that appear less in all the documents and more in individual document participate more towards classification.TF-IDF is a combination of two terms.

TF: Term Frequency, calculates how repeatedly a term appears in a document. Because every single document is distinct in length, it is likely that a term would occur much more times in longer documents than shot ones. Therefore, the term frequency is usually divided by the length of the document, which is the total number of terms in the document ,as a method of normalization.

$$TF = \frac{(frequency\ of\ a\ word\ in\ the\ document\ )}{(Total\ words\ in\ the\ document)}$$

IDF: Inverse Document Frequency, calculates how valuable a term is. While calculating TF, all terms are considered uniformly significant. However, it is known that some terms, such as "is", "of", and "that", may occur a lot of times but have slight significance. Therefore, we must weigh down the frequent terms while scale up the infrequent ones, by calculating the following:

$$IDF = \log(\frac{total\ number\ of\ documents}{number\ of\ documents\ containing\ the\ word})$$

TF-IDF: Term Frequency-Inverse Document Frequency

$$TF\_IDF = TF * IDF$$

If the TF-IDF score is high, it implies that the words are fairly rare and is good at distinguishing between documents. This could be more useful than Count Vectorizer, which only counts how many times a word occurs.

II. CASE STUDY

The dataset used for this project is the Quora question pair dataset from Kaggle [1]. Quora is a platform to acquire and distribute knowledge related to anything. It is a stage to raise questions and connect with people who provide distinctive visions and quality answers. Numerous questions with the similar objective can cause seekers to devote extra time obtaining the best answer to their question, and make writers think they must answer multiple versions of the same question.

Task 1: Supervised - Identification of Duplicate Questions

Identification of duplicate questions problem is handled as Binary classification problem using word embeddings. To determine how 'close' two pieces of text are both in surface closeness and meaning. Rather than performing a word to word comparison, we also should think about the context in order to capture additional semantics. We know that while the words significantly overlap, these two phrases might have different meaning. This is where we use word embeddings to be sensitive to this variation. The big idea is that we represent questions as vectors of features using different word embedding and pass them into the classifier which predicts the labels 1 (duplicate) or 0 (non - duplicate). The dataset contains 404,348 question pairs with 255,042 non-duplicate and 149,306 duplicate questions (figure 4). For task 1, this data was split into 363,913 and 40,435 instances as train and test data, respectively. The train data is further divided into 327,521 and 36,392 instances as train and validation data respectively for deep learning models.

Task 2: Unsupervised Task - Clustering Quora Questions

We will develop an unsupervised text clustering methodology, which is a sentence clustering in our case, that enables Quora to programmatically bin its data. These bins on their own are programmatically produced depending on the algorithm's comprehension of the data. This would help tone down the volume of the data and comprehend the wider spectrum easily. For this task, we created a new dataset with the first questions from the all pairs and all the second questions from the non-duplicate pairs which is a total of 659,390 questions.

Word Cloud is a data visualization technique utilized for depicting text data, where the size of each word signifies its frequency. Critical textual data points can be emphasized using a word cloud. Word clouds are widely used for analyzing data from social networking websites, here we use it on drug data. The disadvantage of word clouds is they are not suitable for all situations.

Frequency Distribution reveals the frequency of each vocabulary item in the text. It is called "distribution" as it shows how the total number of word tokens in the text are distributed across the vocabulary items. We automatically identify the words of a text that are most informative about the topic and genre of the text. Here we use frequency distribution to find the topmost 50 words in our questions, figure 5 depicts this.

	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0
5	5	11	12	Astrology: I am a Capricorn Sun Cap moon and c...	I'm a triple Capricorn (Sun, Moon and ascendan...	1

Figure 3: Quora Question Pair Dataset for task 1

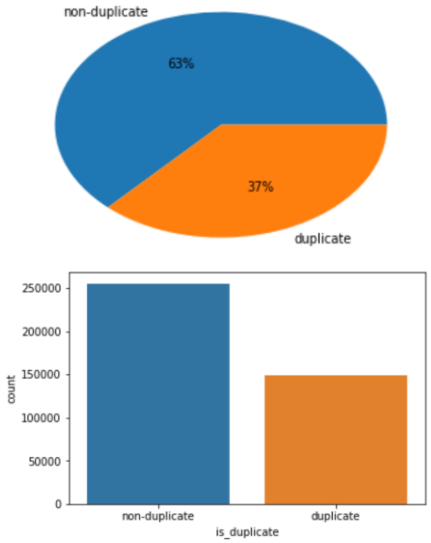

Figure 4: Classification Label for Task 1

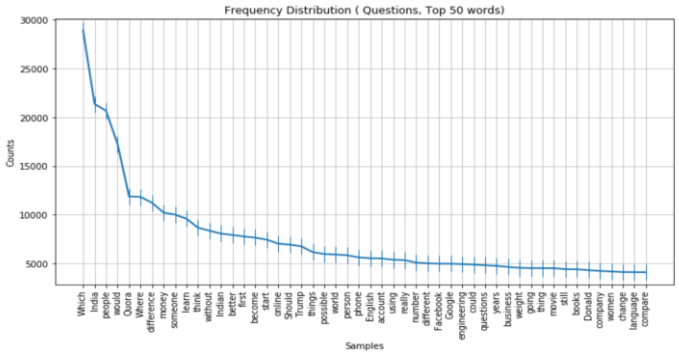

Figure 5: Frequency Distribution for questions in Task 2

III. METHODOLOGY

A. Text Processing

Before we start with the supervised or unsupervised tasks, we need to clean the text in our reviews. Usually, this phase is crucial and typically takes a long time than building Machine Learning models.

Figure 6: Word Cloud for questions in Task 2

Text preprocessing is amongst the highly essential methods in Natural Language Processing (NLP). For example, you would want to remove all punctuation marks from text documents prior to text classification. Likewise, you would want to remove numbers from a text string. Writing manual scripts for such preprocessing methods involves a lot of work and likely to errors. Keeping in mind the significance of these preprocessing methods, the Regular Expressions have been established in different languages in order to make these text preprocessing tasks easier. A Regular Expression is a text string that illustrates a search model which is used to match or replace patterns inside a string.

a. Data Cleaning

Now Let us see what concerns and what does not in these tasks Words are the most crucial part of this analysis. Nevertheless, when we talk about factors like punctuation, you cannot understand the semantics of the question from its punctuation. So, punctuation is not important for these tasks. After obtaining the text, text normalization is applied on it using regular expression.

Text normalization includes:

- converting all letters to lower case
- removing numbers
- removing stop words
- removing other language words apart from English
- removing usernames
- removing special characters
- removing all single characters

Special characters include – [! ” # $ % & ’ ( ) * + , - . / : ; < = > ?]. Stop Words are words which have a bit or no importance, particularly when forming essential features from text. These are mostly words that hold the highest frequency in a corpus. These can be prepositions, articles, conjunctions, etc. Consider the words such as “effect” and “Effect”, from our questions. These words give the same meaning for humans, the only distinction between them for us would be that the initial word is capitalized, for the reason that, it may be the initial word of a sentence or typing error. But for the models, these words will have a different meaning because of their different spelling, so we first convert all the words into lower case.

b. Tokenization

The function of splitting or tokenizing the provided text into smaller pieces termed as tokens is called tokenization. Numbers, Words, punctuation marks, etc. can be treated as tokens. We use Word Tokenization which is Splitting words in a sentence using word tokenize function built in nltk package in python.

B. Supervised Learning

The data set is a set of labelled examples. Each x_i is a feature (attribute) vector with D dimensions. x_k^j is the value of the feature j of the example k, for $j \in 1 \dots D$ and $k \in 1 \dots D$. The label y_i is either a class, taken from a finite list of classes, $\{1,2, \dots, C\}$ or a real number, or a more complex object (vector, matrix, tree, graph, etc.). The Problem is: given the data set as input, create a “model” that can be used to predict the value of y for an unseen x. Regression and classification are categorized under supervised machine learning. The machine learning techniques used for this dataset is classification. The classifiers we use to identify duplicate questions are logistic Regression, Random Forest and deep learning models.

We used Bidirectional Long Short-Term Memory (BiLSTM) model as deep learning model (Figure 14). The idea of BiLSTM is it includes duplicating the first recurrent layer in the network to create two layers side-by-side, then giving the input sequence as input to the first layer and giving a reversed copy of the input sequence to the second layer. Hence, BiLSTM memorizes the past and future data i.e. words in order to decide on the classification label.

C. Unsupervised Learning

The data set is a collection of unlabelled examples. Each x_i is a feature (attribute) vector with D dimensions. x_k^j is the value of the feature j of the example k, for $j \in 1 \dots D$ and $k \in 1 \dots D$. The Problem is: find the underlying “structure” of the data. The problem is vaguely defined compared to supervised learning. Likewise, measuring performance will also be problematic.

Clustering is one of the most popular method used to obtain an insight regarding the structure of the data. It is described as the task of detecting subcategories in the data in a way that data points in the same subcategory (cluster) are extremely similar while data points in distinct clusters are very distinct. Which is nothing but, we try to obtain homogeneous subcategories inside the data such that data points in every cluster are as identical as possible corresponding to a similarity measure like Euclidean-based distance (used for this dataset) or correlation-based distance. The algorithm used for this dataset is KMeans.

Kmeans needs number of clusters (k) , as clustering parameter. Finding the optimal number of clusters is very significant in the analysis. If k is too large, every point will widely start depicting a cluster and if k is too small, then data points are wrongly clustered. Finding the optimal number of clusters leads to granularity in clustering,so we use elbow method.

Elbow method provides a concept on what a good k number of clusters would be established on the sum of squared distance (SSE) among data points and their allotted cluster centroids. We select k at the point where SSE begins to flatten out and developing an elbow. We have used the *KElbowVisualizer* from *yellowbrick.cluster* library in python, which automatically indicates the selected k value depending on the curve. (Can be observed in figure 13)

D. Under Sampling

It is ideal to apply under-sampling on train data but not on test data. From exploratory analysis we can conclude this is an example of Binary (as there are 2 labels, 0 & 1) Imbalanced Data. A simple under-sampling technique is to under-sample the majority class ,class 0 (non duplicate), randomly and uniformly, was introduced into this framework to reduce imbalance to a great extent. After under sampling using RandomUnderSampler in python (Figure 7), our data reduces to size 242326,which consists of 120990 of duplicate and non-duplicate labels.

IV. EVALUATION

A. Supervised Learning

We will comapre the classification report and confusion matrix for the each model.

Confusion matrix: is a synopsis of prediction outcomes on a classification task. The count of accurate and inaccurate predictions is encapsulated and divided by each class. The confusion matrix reveals the traditions in which our classification model is puzzled when it produces predictions. It provides us vision on the errors produced by a classifier and the kind of errors which are being produced. The classification report reveals a depiction of the major classification metrics on a per-class basis. This gives a greater insight on the classifier performance over overall accuracy which can conceal functional weaknesses in one class of a multiclass problem. Visual classification reports are used to analyze classification models to choose models. The components of classification report are explained below in detail.
Precision: is the capability of a classifier not to label a negative instance positive. For each class it is identified as the ratio of true positives to the sum of true and false positives.
Recall: is the capability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives.
F_1 score: is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. In general, F1 scores are lower than accuracy measures as they implant precision and recall into their calculation.
Support: is the number of actual occurrences of the class in the specified dataset.
Accuracy: is the ratio of number of correct predictions to the total number of input samples.
micro: Calculate metrics globally by counting the total true positives, false negatives and false positives.
macro: Calculate metrics for each label and find their unweighted mean, does not take label imbalance into account.
weighted: Calculate metrics for each label, and find their average weighted by support, does take label imbalance into account.[10]

From figure 4 we can infer that our data is imbalanced and for imbalanced data the best metric is accuracy and weighted F1 Score (weighted), so we conclude which is the best classifier for each approach using these metrics. The best performing classifiers confusion matrix and learning curves are shown in the results section, rest are not listed here to keep the report concise but can refer to the code provided in .pdf format for all the results.

B. Unsupervised Learning

Silhouette Score

Silhouette analysis can be used to determine the degree of separation between clusters.[9]
The Silhouette Coefficient is calculated using the below formula:

$$SC(i) = \frac{b^i - a^i}{max(a^i, b^i)}$$

where a^i is the average distance of point i from all other points in its cluster and b^i is the smallest average distance of i to all points in any other cluster. The Silhouette Coefficient reveals how properly allocated every single point is. If $SC(i)$ is close to 0, it is right at the inflection point between two clusters. If $SC(i)$ is closer to -1, then we would better off allocating it to the other cluster. If $SC(i)$ is close to 1, then the point is properly allocated and can be understood as belonging to a suitable cluster. Its disadvantage is that it can be highly expensive to calculate on all n points. This is for the reason that we must calculate the distance of i from all other $n-1$ points for every i, that leads to a complexity of $O(n2)$.

Davies Bouldin Index:

DB Index is calculated using the below formula:

$$DB = \frac{1}{n}\sum_{i=1}^{n} max\left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right)$$

where n is the number of clusters and σ_i is the average distance of all points in cluster i from the cluster centroid c_i.
The DB index apprehends the instinct that clusters that are well-set apart from each other and they are extremely dense are possibly a 'good' cluster. This is for the reason that the measure's 'max' statement continually chooses the values where the average point is farthest away from its centroid, and where the centroids are closest together. As the DB index decreases, the clustering is believed to be 'better'.

These metrics do not determine the genuineness of the model's predictions. We do not have any logical way to verify how valid the cluster predictions are. Rather, the metrics calculate the relative performance of models versus each other. The best performing embedding's $k - value$ and clusters are shown in the results section, rest are not listed here to keep the report concise, but can refer to the code provided in .pdf format for all the results.

V. RESULTS

Now let us discuss the results for each task using different embeddings with different models.

A. Task 1: Supervised Learning

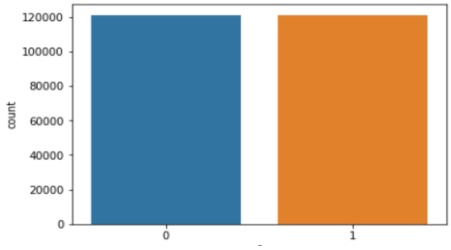

Figure 7: Classification Label for Task 1 after sampling on Train Data. 0 indicates non-duplicate and 1 indicates duplicate questions.

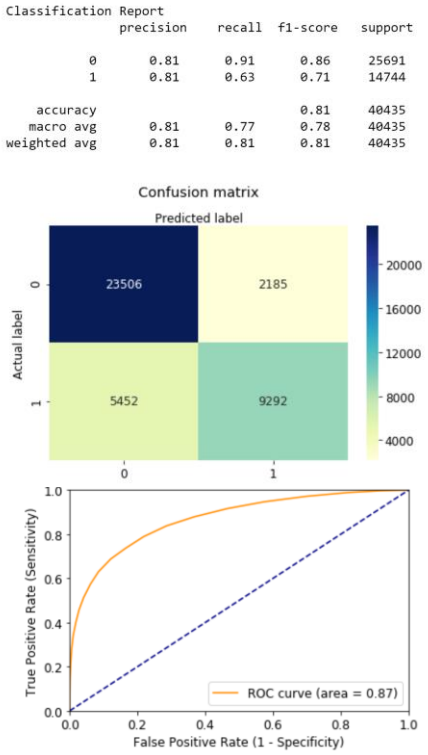

Figure 8: Classification report, Confusion Matrix and ROC Curve for TF-IDF with random forest on Quora Dataset for Task 1

Embedding	Model	Dim	Acc	F1	Sampling
TF-IDF	LR	-	0.72	0.73	Yes
TF-IDF	RF	-	0.77	0.78	Yes
TF-IDF	LR	-	0.76	0.76	No
TF-IDF	**RF**	**-**	**0.81**	**0.80**	**No**
FastText	RF	300	0.76	0.76	Yes
FastText	LR	300	0.67	0.67	Yes
FastText	**RF**	**300**	**0.78**	**0.77**	**No**
FastText	LR	300	0.68	0.66	No
FastText	DL	300	0.68	0.68	Yes
Doc2Vec	DL	300	0.67	0.68	Yes
Doc2Vec	RF	300	0.72	0.73	Yes
Doc2Vec	LR	300	0.71	0.71	Yes
Doc2Vec	RF	300	0.74	0.74	No
Doc2Vec	LR	300	0.73	0.72	No
Glove	LR	300	0.65	0.65	Yes
Glove	RF	300	0.76	0.76	Yes
Glove	LR	300	0.68	0.66	No
Glove	**RF**	**300**	**0.78**	**0.77**	**No**
Glove	DL	300	0.68	0.68	Yes
Word2Vec	DL	300	0.67	0.68	Yes
Word2Vec	RF	300	0.77	0.77	Yes
Word2Vec	LR	300	0.66	0.67	Yes
Word2Vec	**RF**	**300**	**0.79**	**0.78**	**No**
Word2Vec	LR	300	0.69	0.67	No
SBERT	RF	768	0.53	0.53	Yes
SBERT	LR	768	0.49	0.50	Yes
SBERT	LR	768	0.55	0.53	No
SBERT	RF	768	0.55	0.54	No
SBERT	DL	768	0.58	0.56	No
SBERT	DL	768	0.42	0.40	Yes

Table 1: Results of various Supervised learning models (Task 1) on Quora dataset with different word embeddings. *Dim* is the dimension of the used embedding, *Acc* is the accuracy, *F1* is the F1 Score, *DL* is deep learning model with the architecture as in figure 14, *LR* is logistic regression, *RF* is random forest. For all the word embeddings the hyperparameters of the deep learning models are the same: 10 epochs, learning rate 0.001, Batch Size 1024, Same sapling technique, ADAM optimizer.

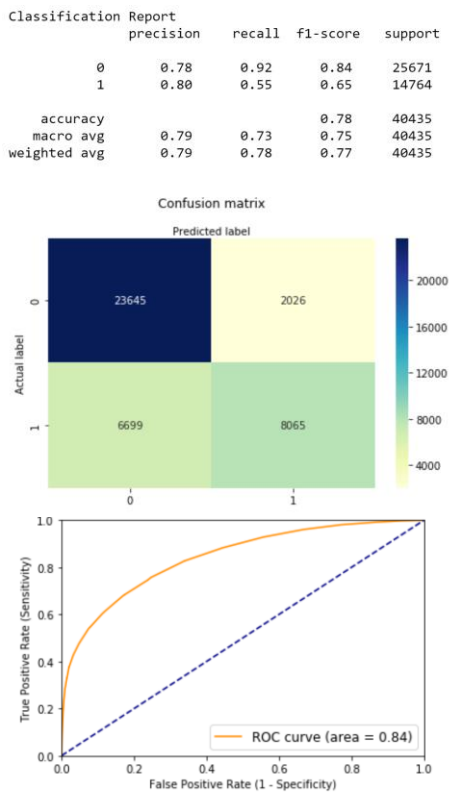

Figure 9: Classification report, Confusion Matrix and ROC Curve for Glove with random forest on Quora Data for Task 1

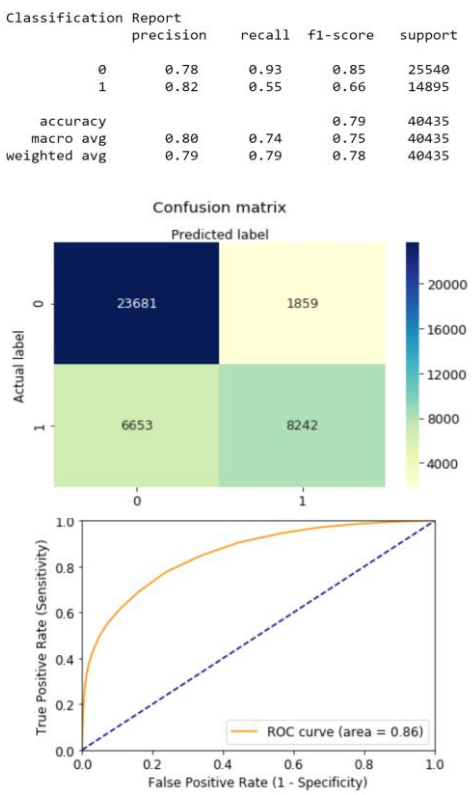

Figure 10: Classification report, Confusion Matrix and ROC Curve for Word2Vec with random forest on Quora Data for Task 1

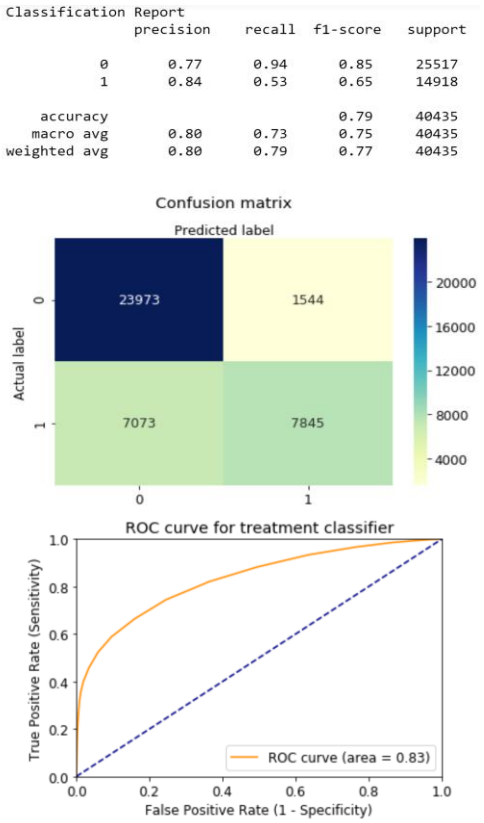

Figure 11: Classification report, Confusion Matrix and ROC Curve for FastText with random forest on Quora Dataset for Task 1

B. Task 2: Unsupervised Learning

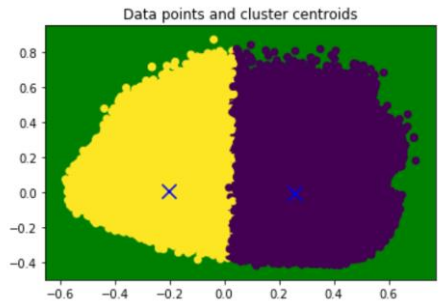

Figure 12: Clusters in Quora dataset by Doc2Vec using *KMeans* Algorithm

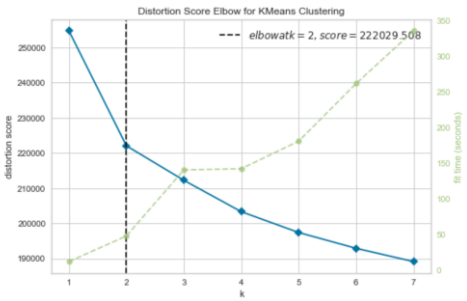

Figure 13: No of optimal clusters in Quora dataset by Doc2Vec using *KElbowVisualizer*

Embedding	Trained on	Dim	k	SS	DBI
Doc2Vec	**Quora data**	**50**	**2**	**0.13**	**2.48**
Doc2Vec	**Quora data**	**100**	**2**	**0.12**	**2.49**
Doc2Vec	**Quora data**	**300**	**2**	**0.12**	**2.58**
Word2Vec	Quora data	100	4	0.07	3.17
FastText	Quora data	100	3	0.05	3.75
FastText	Wiki-news	300	5	0.01	4.34
Word2Vec	Google-News	300	3	0.03	4.55
Word2Vec	Quora data	100	4	0.07	3.17
Glove	Twitter	50	3	0.07	3.07
Glove	Common Crawl	300	3	0.04	3.79
TF-IDF	Quora data	-	6	ME	ME

Table 2: Results of clustering with various word embeddings (Task 2). Second column indicate the data on which the word embeddings were trained. We used pretrained word embeddings on Wiki-news, Google-News, Twitter, Common Crawl are data. *Dim* is the dimension of the used embedding, *k* is the number of clusters detected using the elbow method, *SS* is the Silhouette Score, *DBI* is the Davies Bouldin Index and *ME* is memory error, due to which results were not obtained.

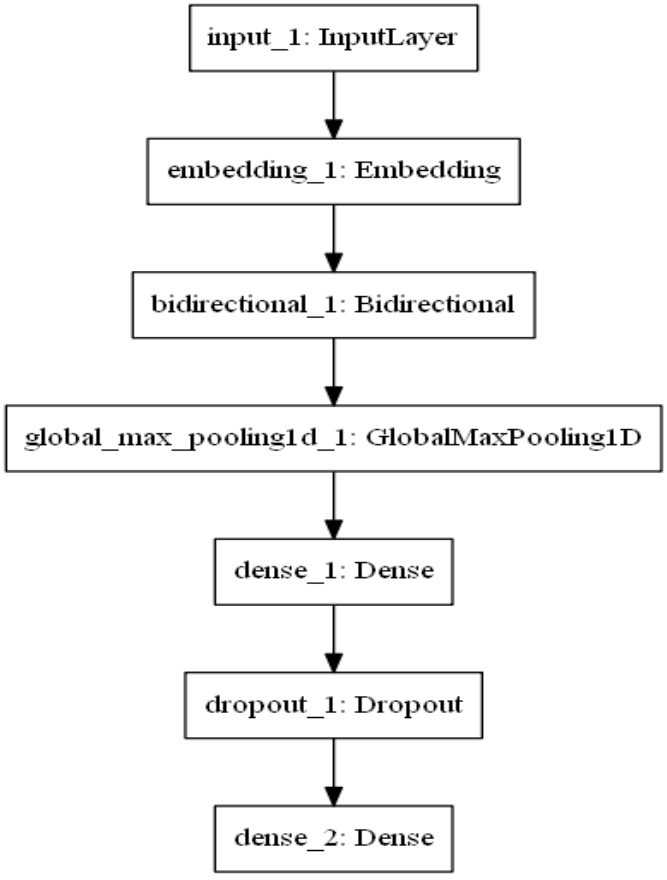

Figure 14: Deep Learning Model Architecture used with few embeddings for task 1

VI. DISCUSSION

In this section we will discuss the challenges faced and the future work for our tasks on Quora dataset.

The biggest challenge here is the relationship of memory and data. This project models were executed with a windows laptop (core i7) of 16 GB RAM and a 4 GB graphics card. It took us a long time to obtain the obtain the word embeddings for BERT, ,XLNET, GPT2 but while running the models we observed that the program was crashing and were not able to proceed further due to memory constraints. Also, with the memory contraints it was difficult to execute the models after obtaining the word embeddings with this dataset (404,348 question pairs for task 1 and 659,390 questions for task 2). Therefore, we could not obtain results for these embedding with respect to our tasks. We also faced memory error while computing the SS and DBI for TF-IDF in Task 2, we could only compute the *k* value but could not compare its performance with other embeddings. The future work would be to implement these tasks on other embeddings for both the tasks, implement the other existing state of the art clustering methods like DBSCAN for task 2,along with able to find the SS and DBI for TF-IDF to compare with results obtained using word embeddings.

VII. CONCLUSION

Our early focus for this project was clear, we tackled some challenges during the project, but we conclude the following observations made on Quora Dataset.

Task 1: Supervised Learning

For most of the embeddings random forest classifier worked better than other classifiers. Using same dimensions for different embedding resulted in similar accuracy for each type of model. Word2Vec with random forest model have the highest accuracy which is 79% and F1 Score of 0.78. We can also observe that for this Quora Dataset, as the accuracy goes up so does the F1 Score. Comparatively for this dataset, TF-IDF performed with Random Forest (without sampling) better than using word embeddings. This might be due the fact that TF-IDF considers the frequency of the word in questions. For the RF and LR models with respect to each word embedding, without sampling results in more accuracy.

Task 2: Unsupervised Learning

For clustering we cannot verify if the clusters we obtained are exactly correct or not, but we can conclude on the best word embedding with KMeans for this dataset using the Silhouette Score and Davies Bouldin Index. For all the word embeddings, training word embeddings on Quora data had higher Silhouette Score and lower Davies Bouldin Index, which proves that they are well allotted clusters compared to clusters allotted using pertained word embeddings. The minimum and Maximum number of clusters found for this dataset using these word embeddings is 2 and 5 respectively. We can observe that Doc2Vec has high Silhouette Score and low Davies Bouldin Index (using any number of dimensions), which proves that this is the best allocated clusters for this dataset among what we have, with number of clusters equal to 2.But usually having such low values of k does not seem to be ideal.

VIII. REFERENCES

[1] Kaggle Internet: https://www.kaggle.com/c/quora-question-pairs 2017

[2] Anirudh.S "Word Embedding Using BERT In Python" Internet: https://towardsdatascience.com/word-embedding-using-bert-in-python-dd5a86c00342

[3] Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks by Nils Reimers and Iryna Gurevych ,Ubiquitous Knowledge Processing Lab (UKP-TUDA) Department of Computer Science, Technische Universitat Darmstadt- Aug 2019

[4] Internet: http://www.ashukumar27.io/Word2vec/ 19 Feb 2018

[5] A combination of active learning and self-learning for named entity recognition on Twitter using conditional random fields by Van Cuong Tran , Ngoc Thanh Nguyen , Hamido Fujita , Dinh Tuyen Hoang , Dosam Hwang

[6] Sentiment Aware Word Embedding Approach for Sentiment Analysis by Win Lei Kay Khine, Nyein Thwet Thwet Aung, 2nd International Conference on Advanced Information Technologies (ICAIT) Nov 2018

[7] Jatin Mandav,"Sentiment Analysis Using Word2Vec, FastText and Universal Sentence Encoder in Keras" Internet: https://jatinmandav.wordpress.com/2018/07/29/sentiment-analysis-using-word2vec-fasttext-and-universal-sentence-encoder-in-keras/ Jul 29, 2018

[8] Soner ,Medium Article"Top Machine Learning Algorithms for Clustering by Soner" Internet: https://towardsdatascience.com/top-machine-learning-algorithms-for-clustering-a09c6771805 Apr 2019

[9] Imad Dabbura,Medium article"K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks" Internet: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a Sep 2018

[10] GraphLab Create API "Model Evaluation" Internet: https://turi.com/products/create/docs/generated/graphlab.evaluation.f1_score.html 06 Jul 2016

[11] "Text Clustering with doc2vec Word Embedding Machine Learning Model" Internet:https://ai.intelligentonlinetools.com/ml/text-clustering-doc2vec-word-embedding-machine-learning/ 22 Sep 2018