Quora Question Pairs

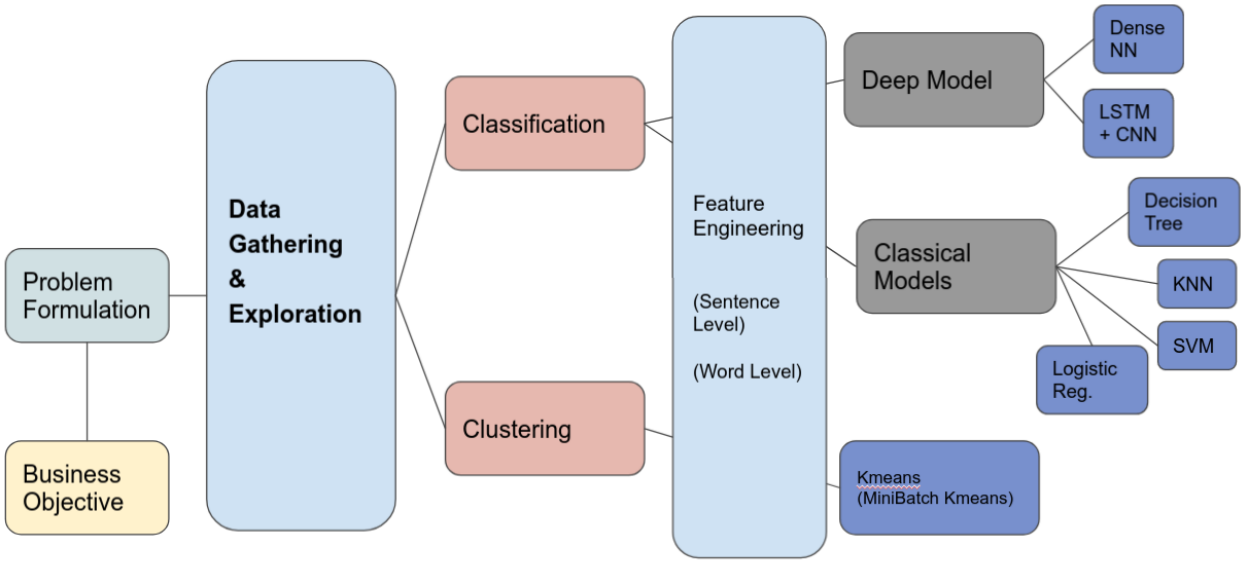
25.07.2021

Abdelrahman Ibrahim
Lamis Sayed
Minah Ghanem
Mohammed Elnamory

Github Repo : https://github.com/sharafcode/quora-question-pairs

Problem Set

Identifying question pairs on quora that have similar intents to solve the problem of rendent questions.



Problem formulation

Quora is a website to share and gain knowledge on multiple aspects. People can ask questions about anything and wait for a variety of answers and opinions from ordinary to expert people. Over 100 million people visit Quora every month, and due to that large number of users ask answers that are already asked and this blocks them from seeing the quality answers on the old posts and also blocks other users from seeing new questions every day.

What is the business objective ?

So detecting and preventing users from asking repeated questions could stop the website from being repeated and boring. It could be a place for a variety of questions from different topics. Reduce the amount of time the user will need to ask a question by recommending similar questions that have been previously asked and offer high quality answers. In that way, we would enhance the user experience and improve the benefit of the website.

How can data science be a solution for this problem ?

Data science and Machine learning could help by analyzing the questions that are being asked on Quora and detect the patterns and behaviours of the users. We can also use similarity measures, classification and clustering techniques to detect similar questions and prevent the user from asking repeated questions.

What is the cost ?

The cost of this is that more computational resources are needed in order to detect similar questions and recommend similar posts to the user. Also this might increase the time that the user takes to post a new question

Data Analysis

In this part, we performed some data analysis tasks in order to help us understand the data and its problems more. It also helped us in making some decisions in the preprocessing steps

Exploration of the dataset

The dataset we used is from Kaggle's quora question pair competition.

The data format is as follows: the ID of each question, question 1, question 2 and the label whether it is duplicated or not.

	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

The dataset contains **404290 entries** and **3** of these entries have **null values** in one of the two questions. We need to remove them as part of the cleaning process

```
train_df.iloc[105780,:]
```

```
id                                        105780
qid1                                      174363
qid2                                      174364
question1       How can I develop android app?
question2                                    NaN
is_duplicate                                   0
Name: 105780, dtype: object
```

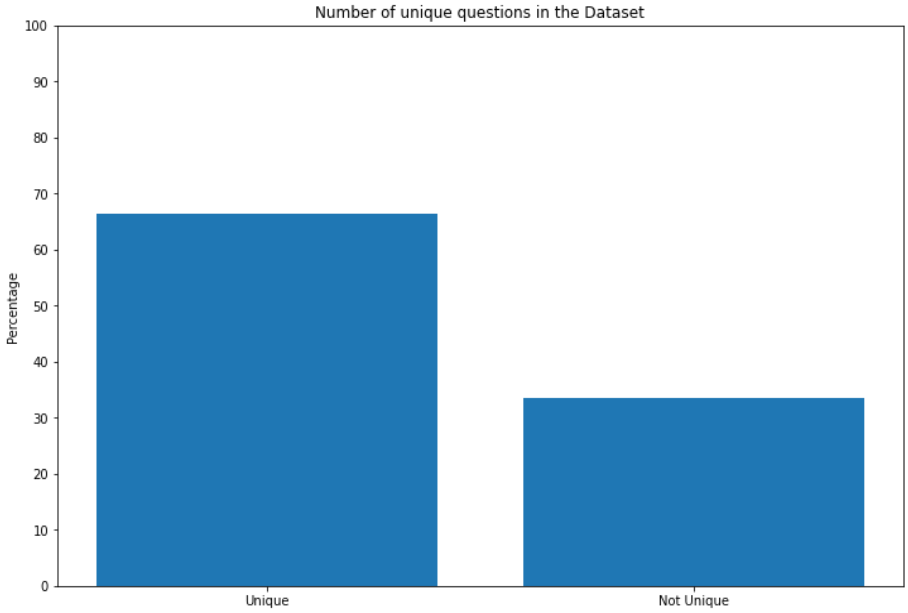
```
train_df.iloc[201841,:]
```

```
id                                        201841
qid1                                      303951
qid2                                      174364
question1       How can I create an Android app?
question2                                    NaN
is_duplicate                                   0
Name: 201841, dtype: object
```

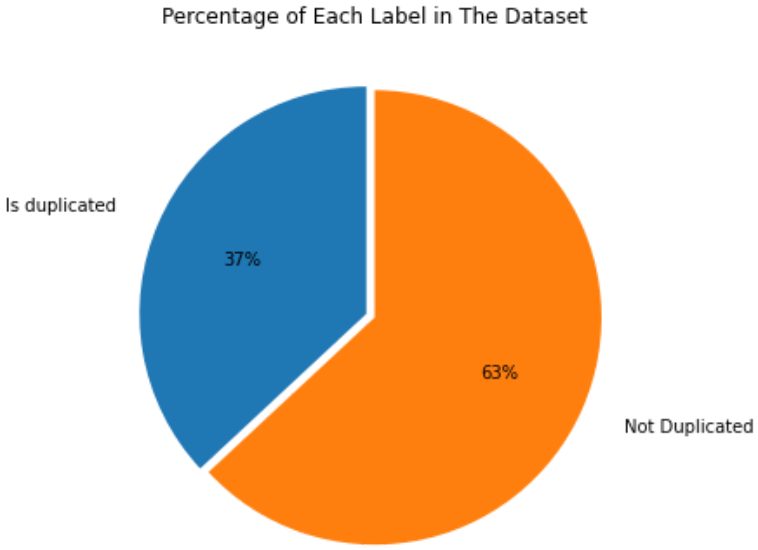
```
train_df.iloc[363362,:]
```

```
id                                                        363362
qid1                                                      493340
qid2                                                      493341
question1                                                    NaN
question2       My Chinese name is Haichao Yu. What English na...
is_duplicate                                                   0
Name: 363362, dtype: object
```

Question 1 column in the dataset contains 290456 unique questions and Question 2 column contains 299174. Counting all the questions in the dataset, we have 537360 unique ones out of 808580 (Counting the two question columns in the dataset) **which make the percentage of the unique questions 66%.**



Now coming to the **percentage of each label in the Dataset**, we found that **37% is duplicated questions and 63% is not duplicated** which makes an unbalanced dataset.



Now we tried to check if there is a case where question 1 is the same as question 2 and we found zero rows under that case,

We conducted that test again after performing some cleaning on the text dataset which contains **removing unnecessary symbols** other than alphabets and numbers, **removing unnecessary excessive spaces and tabs** and finally **removing all the stop words other than the beginning of the question like:**
['who','when','why','where','which','whom','how','what'] because we believe that they are semantically important.

We found that after cleaning there are **11295 records that have question 1 = question 2** and **1979 has the label of not duplicated question**, so we needed to examine these questions more.

Some of these cases are caused by removing the stopping words like the below example. After lowering the Text and removing stop words, the word "IT" is removed so the questions appear to be the same while in fact they are different

```
id                                          222
qid1                                        445
qid2                                        446
question1             How can I find job in Japan?
question2       How can I find an IT job in Japan?
is_duplicate                                  0
Name: 222, dtype: object
```

A lot of these questions are because of human error labeling which is a problem that exists in the dataset like the below examples, these questions have quite the same semantic meaning

```
id                                                          41
qid1                                                        83
qid2                                                        84
question1       When can I expect my Cognizant confirmation mail?
question2          When can I expect Cognizant confirmation mail?
is_duplicate                                                 0
Name: 41, dtype: object
```

```
id                                    277
qid1                                  554
qid2                                  555
question1       How do most people die?
question2            How do people die?
is_duplicate                            0
Name: 277, dtype: object
```

The next part of exploring, we needed to check if there are questions which have length less than two words and we found 66 in question 1 and 22 in question 2. Most of these records are garbage and do not have a meaning as we can see in the below screenshot in question 2.

	id	qid1	qid2	question1	question2	is_duplicate	quest1_len	quest2_len
20072	20072	37898	37899	How could I solve this?	.....	0	5	1
46596	46596	83328	83329	How To Edit DNA?	What?	0	4	1
47056	47056	84067	84068	Is there anywhere in the world offering pain m...	?	0	19	1
51909	51909	92003	83329	What should yellow taxis do to face competitio...	What?	0	12	1
74304	74304	127360	127361	Is there any chances for hailstones tomorrow?	parisflatlist	0	7	1
109009	109009	178981	178982	How do I make a box that I can put my phone in...	Hh	0	25	1
130637	130637	209606	209607	Does the Quran surah 23: 5-6 prove that Allah ...	Does?	0	16	1
144506	144506	228687	228688	How beautiful do you think you are?	Delete	0	7	1
145000	145000	229359	229360	Me and my girlfriend wish to go out and do 'st...	Deleted.	0	28	1
175282	175282	269923	44699	Hh	HH	1	2	1

Preprocessing & Data Cleaning

For the data cleaning stage, we done the following steps:

- Removing rows with null values in the dataset
- Removing questions that have length less than 2 words
- Removing unnecessary symbols other than alphabets and numbers.
- Removing unnecessary excessive spaces and tabs
- As for removing the stop words we set it as an optional parameter as removing it reduced the performance of some models we also kept the beginning of the question like ['who', 'when', 'why','where','which','whom','how','what'] as we consider them semantically important

Classification

In the Classification part, we have tried different strategies and methods. Here we represent 3 different pipelines to solve the solution in different ways:

Pipeline 1

I. Feature Extraction

In this pipeline, we used TFIDF vectors as our features. We got the TFIDF vectors for each question and then got the different vectors of them.

Due to the large number of entries in the data and the large number of words, we used dimensionality reduction method using truncated SVD as it can work with sparse matrices efficiently.

In the next step we trained the multiple algorithms on 200000 entries of the dataset due to low computational resources.

We have tried (SVM - Decision Trees - Knearest Neighbour- Random Forest)

II. Model Evaluation

For the SVM model, we got the following results:

```
SVM on TFiDF
SVM  Train Accuracy :  75.514375
SVM  Test Accuracy :  72.2275

               TEST DATA METRICS
SVM  Confusion Matrix:  [[21939  3128]
 [ 7981  6952]]
SVM  Report :
              precision    recall  f1-score   support

           0       0.73      0.88      0.80     25067
           1       0.69      0.47      0.56     14933

    accuracy                           0.72     40000
   macro avg       0.71      0.67      0.68     40000
weighted avg       0.72      0.72      0.71     40000
```

For the decision tree, we got the following results

```
Decision Tree on TFiDF
Decision Tree  Train Accuracy :  71.245625
Decision Tree  Test Accuracy :  67.18499999999999

                TEST DATA METRICS
Decision Tree  Confusion Matrix:  [[20888  4179]
 [ 8947  5986]]
Decision Tree  Report :
              precision    recall  f1-score   support

           0       0.70      0.83      0.76     25067
           1       0.59      0.40      0.48     14933

    accuracy                           0.67     40000
   macro avg       0.64      0.62      0.62     40000
weighted avg       0.66      0.67      0.65     40000
```

And for the Random Forest model we got the following results

```
Random Forest on TFiDF
RandomForest  Train Accuracy :  84.606875
RandomForest  Test Accuracy :  71.52

                TEST DATA METRICS
RandomForest  Confusion Matrix:  [[22728  2339]
 [ 9053  5880]]
RandomForest  Report :
              precision    recall  f1-score   support

           0       0.72      0.91      0.80     25067
           1       0.72      0.39      0.51     14933

    accuracy                           0.72     40000
   macro avg       0.72      0.65      0.65     40000
weighted avg       0.72      0.72      0.69     40000
```

For the KNN, we got:

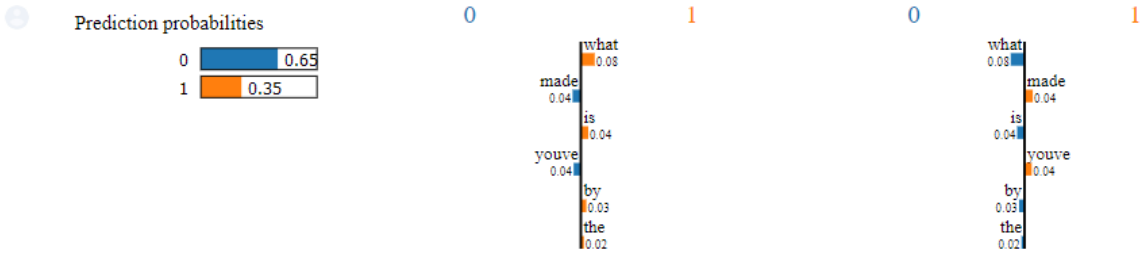
```
KNN on TFiDF
KNN  Train Accuracy :  79.335
KNN  Test Accuracy :  68.8975

                TEST DATA METRICS
KNN  Confusion Matrix:  [[18710  6357]
 [ 6084  8849]]
KNN  Report :
              precision    recall  f1-score   support

           0       0.75      0.75      0.75     25067
           1       0.58      0.59      0.59     14933

    accuracy                           0.69     40000
   macro avg       0.67      0.67      0.67     40000
weighted avg       0.69      0.69      0.69     40000
```

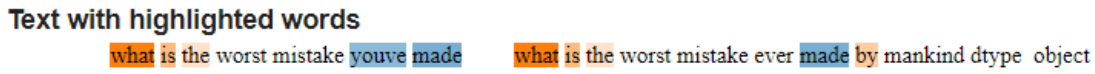
We can see from the above results of all of the models, that the **best model** was the **random forest** as it scored the highest accuracy of 84% on the training dataset and 71.5% on the test dataset. Also KNN scored 79% on the training dataset and SVM scored 75% on the training dataset as well.

We can predict that if we had trained the model on the whole dataset, the accuracy would have been higher. Also deriving new features other than TFIDF might increase the accuracy and that's what we have tried on the other pipelines.

III. Error Analysis

For the Error analysis part, we have used the lime text explainer in order to analyze the source of error. The following example is from the logistics regression model:





For the above example, the actual prediction here that the question pairs are not duplicate , and the prediction that they are duplicate and that's because there are multiple words that are common between the two questions but have a different order and only one word which is different "mankind" , so that's why the model treats them as same question(duplicate).

Pipeline 2

I. Feature Extraction

As part of our search in finding a new way for feature extraction we found out that we can use "Cosine Similarities, Minwoski and Euclidean Distances" as new way of getting the distance between the two vectors of each questions after that we used the LDA as feature reduction and we used both ways as input for Logistic Regression model.

Cosine Similarities: measure of similarity between two non-zero vectors of an inner product space. It is defined to equal the cosine of the angle between them, which is also the same as the inner product of the same vectors normalized to both have length 1.

Minkowski Distance: distance or a metric in a normed vector space.

Euclidean Distance : distance between two points in Euclidean space is the length of a line segment between the two points. It can be calculated from the Cartesian coordinates of the points using the Pythagorean theorem.

- After tokenizing question one and question two.

	id	qid1	qid2	question1	question2	quest1_len	quest2_len	common_word_count	word_len_diff	is_duplicate	q1_tokens	q2_tokens
0	0	1	2	step step guide invest share market india	step step guide invest share market	14	12	5	1	0	[step, step, guide, invest, share, market, india]	[step, step, guide, invest, share, market]
1	1	3	4	story kohinoor kohinoor diamond	would happen indian government stole kohinoor ...	8	13	2	5	0	[story, kohinoor, kohinoor, diamond]	[would, happen, indian, government, stole, koh...
2	2	5	6	increase speed internet connection using vpn	internet speed increased hacking dns	14	10	2	1	0	[increase, speed, internet, connection, using,...	[internet, speed, increased, hacking, dns]

- After measuring the distance between the two tokens for the whole dataset

	id	qid1	qid2	...	cosine	manhattan	minkowsk
0	0	1	2	...	0.942809	1.0	1.000000
1	1	3	4	...	0.615457	7.0	2.645751
2	2	5	6	...	0.365148	7.0	2.645751
3	3	7	8	...	0.000000	8.0	2.828427
4	4	9	10	...	0.282843	11.0	3.316625

II. Model Evaluation

We used accuracy score on both training set and testing set for the two models and results came as follows:

- Using distances as input for the model

```
Logistic Regression  Train Accuracy :  66.40050222817364
Logistic Regression  Test Accuracy :  66.44482929242949
```

- Using output of LDA as input for the model

```
Logistic Regression built on LDA  Train Accuracy :  65.77952059475692
Logistic Regression built on LDA  Test Accuracy :  65.7644730331519
```

III. Error Analysis

We used confusion matrix here for the error analysis and the result came as follows for the two models

- Model built on distances as features

```
                TEST DATA METRICS
Logistic Regression  Confusion Matrix:  [[37891 13114]
[14012 15823]]
Logistic Regression  Report :
              precision    recall  f1-score   support

           0       0.73      0.74      0.74     51005
           1       0.55      0.53      0.54     29835

    accuracy                           0.66     80840
   macro avg       0.64      0.64      0.64     80840
weighted avg       0.66      0.66      0.66     80840
```

- Model built on LDA output

```
                TEST DATA METRICS
Logistic Regression built on LDA  Confusion Matrix:  [[39193 11812]
 [15864 13971]]
Logistic Regression built on LDA  Report :
              precision    recall  f1-score   support

           0       0.71      0.77      0.74     51005
           1       0.54      0.47      0.50     29835

    accuracy                           0.66     80840
   macro avg       0.63      0.62      0.62     80840
weighted avg       0.65      0.66      0.65     80840
```

Pipeline 3

I. Feature Extraction

In this pipeline, we used Word2Vec vectors as our feature engineering. We got the Word2Vec vectors for each question and then got the different vectors of them.

In the next step we trained the multiple algorithms on 200000 entries of the dataset due to low computational resources.

We have tried (Decision Trees - Random Forest)

II. Model Evaluation

For the Decision Tree, we got the results.

```
Decision Tree on word2vec
Decision Tree  Train Accuracy :  64.42999999999999
Decision Tree  Test Accuracy :  62.395

                TEST DATA METRICS
Decision Tree  Confusion Matrix:  [[17038  8029]
 [ 7013  7920]]
Decision Tree  Report :
              precision    recall  f1-score   support

           0       0.71      0.68      0.69     25067
           1       0.50      0.53      0.51     14933

    accuracy                           0.62     40000
   macro avg       0.60      0.61      0.60     40000
weighted avg       0.63      0.62      0.63     40000
```

For Random Forest, we got the results.

```
Random Forest on word2vec
RandomForest  Train Accuracy :  81.49187500000001
RandomForest  Test Accuracy :  66.8725

                TEST DATA METRICS
RandomForest  Confusion Matrix:  [[21551  3516]
 [ 9735  5198]]
RandomForest  Report :
              precision    recall  f1-score   support

           0       0.69      0.86      0.76     25067
           1       0.60      0.35      0.44     14933

    accuracy                           0.67     40000
   macro avg       0.64      0.60      0.60     40000
weighted avg       0.65      0.67      0.64     40000
```

IIII. Conclusion.

At the end after checking the results we got we cannot use the distances as an in this case because a result of 66% isn't acceptable at all maybe we can do some further work trying to improve the accuracies and interpreting the process but for now the two inputs for the model confused the model because the way of calculating the distance didn't produce a clear difference so the model couldn't determine whether it's duplicated or not.

Clustering

III. Feature Extraction

In the clustering method, we have used 200000 records of the dataset because of the limitations of computational power we had. We have tried multiple feature extraction methods to enhance our results:

A. TFIDF

We used TFIDF on both questions, then we used the difference between the two vectors. After that we used SVD for dimensionality reduction because it's mainly used for sparse matrices. The dimensionality reduction method helped us in the low computational resources

B. Word2Vec

We have also used word2vec model to calculate a vector of the first question and the second question then we tried different techniques

1. The difference between the two vectors.

Word2Vector Differences

```
##Getting absolute difference between the two questions
w2v_diff=abs(w2v_q1-w2v_q2)
```

2. Cosine similarities between the two vectors

Word2Vector Cosine Similarities

```
[ ] ##Converting to tensorflow
    w2v_q1_tf=tf.convert_to_tensor(w2v_q1, dtype=tf.float32)
    w2v_q2_tf=tf.convert_to_tensor(w2v_q2, dtype=tf.float32)
```

```
[ ] ##obtaining cosine similarities
    cosine_scores =tf.keras.losses.cosine_similarity(w2v_q1_tf,w2v_q2_tf)
```

```
[ ] ##Reshaping the tensorflow vec
    cosine_scores=tf.reshape(cosine_scores, [200000,1])
```

C. Word Embeddings

In this approach we computed embeddings for the two sets of questions then we multiplied them by each other.

We used sentence transformers(paraphrase-MiniLM-L3-v2) to obtain word embeddings. It maps sentences & paragraphs to a 384 dimensional dense vector space.

We used a mini batch K Means model in this approach, as it is suitable for large datasets.

IV. Error Analysis

For the Error Analysis Part, we used a method to compare the actual labels in each cluster to the labels of the cluster

For the TFIDF features, we got the following results. The two clusters are not actually presenting the difference between the two labels here. As Cluster 0 contain 59% of label 0 and cluster 1 contain 67% of label 0

		actual_labels
clusters	actual_labels	
0	0	59.559570
	1	40.440430
1	0	67.765624
	1	32.234376

For the word2vec differences we obtained these results, which is better than the first one. Cluster 0 has more percentage of label 0 and Cluster 1 has less of label 0

		actual_labels
clusters	actual_labels	
0	0	79.044528
	1	20.955472
1	0	53.997201
	1	46.002799

For the Word2vec cosine similarity, we obtained the following results,

We can see that Cluster 1 is almost representing label 0

		actual_labels
clusters	actual_labels	
0	0	55.089185
	1	44.910815
1	0	91.149839
	1	8.850161

For the word embeddings we obtained the following results, which is the best results we have got. Cluster 0 almost represent label 1 with a percentage of 96% and Cluster 1 almost represent label 0 with a percentage of 62%

		actual_labels
clusters	actual_labels	
0	1	96.800000
	0	3.200000
1	0	62.830038
	1	37.169962

V. Model Evaluation

For the model evaluation task, we got the numerical scores using multiple matrices:

- **Homogeneity Score:** This Score should be high if cluster contains only samples belonging to a single class
- **Completeness Score:** This score should be high if if all the data points that are members of a given class are elements of the same cluster
- **V-measure Score:** The harmonic mean of homogeneity and completeness
- **Adjusted rand Score:** The Rand Index computes a similarity measure between two clusterings by considering all pairs
- **Kappa Score**:A statistic that measures inter-annotator agreement
- **Silhouette Score:** A method of interpretation and validation of consistency within clusters of data
- **SearmanrResult:** Spearman correlation coefficient with associated p-value

For Evaluating the **TFIDF vectors**, we got the following numerical values

```
Homogeneity score:  0.01
Completeness score:  0.01
V-measure score:  0.01
Adjusted rand score:  -0.0
Kappa score:  -0.08
Silhouette score:  0.08
Correlation:  SpearmanrResult(correlation=-0.08276865334075897, pvalue=6.201587980712732e-301)
```

For the word2 vector differences, we got the following results

```
Homogeneity score:  0.05
Completeness score:  0.05
V-measure score:  0.05
Adjusted rand score:  0.02
Kappa score:  0.21
Silhouette score:  0.2
Correlation:  SpearmanrResult(correlation=0.24705888835317308, pvalue=0.0)
```

For the word2vec cosine similarity, we got the following results

```
Homogeneity score:  0.08
Completeness score:  0.11
V-measure score:  0.09
Adjusted rand score:  -0.01
Kappa score:  -0.28
Silhouette score:  0.64
Correlation:  SpearmanrResult(correlation=-0.3051773020068975, pvalue=0.0)
```

For the word embeddings :

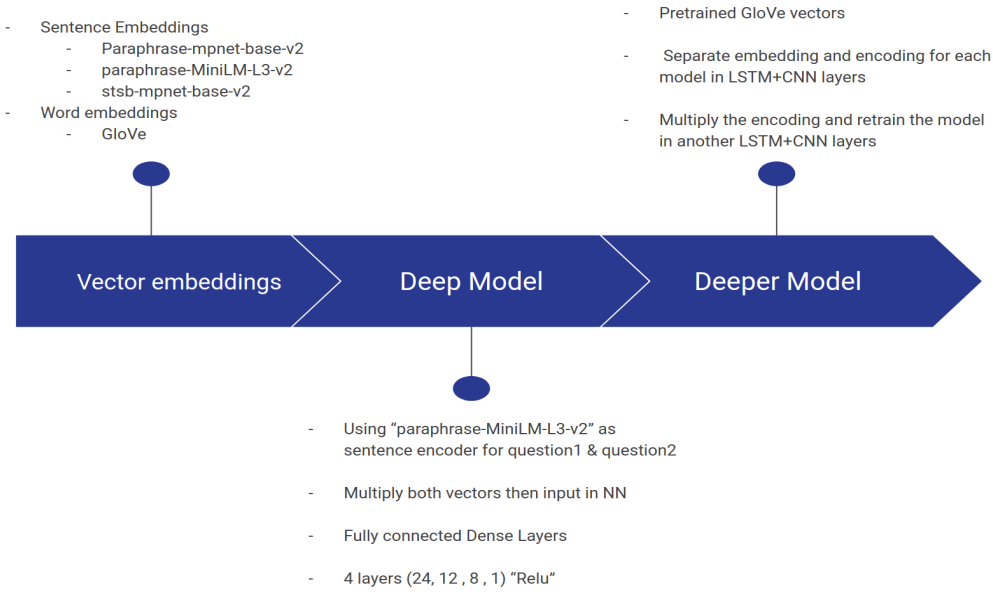
```
Homogeneity score:  0.0
Completeness score:  0.11
V-measure score:  0.0
Adjusted rand score:  0.0
Kappa score:  -0.0
Silhouette score:  0.17
Correlation:  SpearmanrResult(correlation=-0.043580430321308324, pvalue=1.121630903903153e-84)
```

We can see that the numerical results are quite not good enough as the problem is mainly considered as a classification problem. Despite that, models that are based on word embeddings or word2vec will do better. Also considering the low resource of computation. Maybe if we had trained the model on the whole dataset, the clustering algorithm could have worded better.

Classification Using Deep Learning

I. Training Pipeline

- Our pipeline started with text representation, the problem is that we have two questions and we want to represent them in a way that we can detect the similarities between them efficiently.
- We had sentence-level & word-level embeddings.
- First-model is a deep dense model consisting of just 4 layers and an output function.
- Second-model is a very deep model with two encoders and one decoder which is the output decoder as a binary classifier.



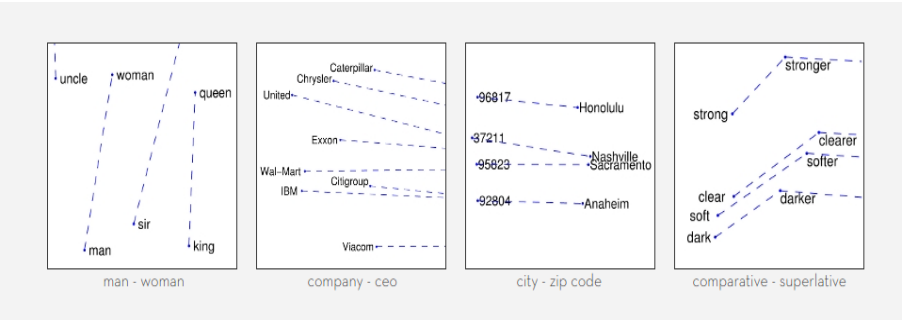
II. Feature Extraction

A. Glove Word Embeddings

- GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

Used this model for the deeper model as an input for the embedding layer for each

question



B. Sentence Embeddings

We used the sentence-transformers python package published by hugging face. We tended to look for the pre-trained models' benchmarks and found some important benchmarks like DupQ which has the same problem as us and found that the pre-trained model **"paraphrase_MiniLM_L12_V2"** has the highest accuracy on it.

Sentence Embedding Models

The following models have been tuned to embed sentences and short paragraphs up to a length of 128 word pieces.

Use **paraphrase-mpnet-base-v2** for the best quality, and **paraphrase-MiniLM-L6-v2** if you want a quick model with high quality.

Model Name	STSb	DupQ	TwitterP	SciDocs	Clustering	Avg. Performance	Speed
paraphrase-mpnet-base-v2	86.99	87.80	76.05	80.57	52.81	76.84	2800
paraphrase-multilingual-mpnet-base-v2	86.82	87.50	76.52	78.66	47.46	75.39	2500
paraphrase-TinyBERT-L6-v2	84.91	86.93	75.39	81.51	48.04	75.36	4500
paraphrase-distilroberta-base-v2	85.37	86.97	73.96	80.25	49.18	75.15	4000
paraphrase-MiniLM-L12-v2	84.41	87.28	75.34	80.08	46.95	74.81	7500
paraphrase-MiniLM-L6-v2	84.12	87.23	76.32	78.91	45.34	74.38	14200
paraphrase-albert-small-v2	83.40	86.57	74.51	80.28	44.94	73.94	5000
paraphrase-multilingual-MiniLM-L12-v2	84.42	87.52	74.94	78.27	43.87	73.80	7500
paraphrase-MiniLM-L3-v2	82.41	88.09	76.14	77.71	43.39	73.55	19000
nli-mpnet-base-v2	86.53	83.22	76.24	72.90	43.38	72.45	2800
stsb-mpnet-base-v2	88.57	85.04	75.35	72.48	39.16	72.12	2800
distiluse-base-multilingual-cased-v1	80.62	84.82	76.24	70.41	40.07	70.43	4000
stsb-distilroberta-base-v2	86.41	82.70	73.68	69.85	37.68	70.07	4000

C. Our feature Tuning

Besides the above benchmarks, we also did our way of tuning on 10K records and calculated the cosine similarities on all questions

- Then, we get the mean similarities of all duplicated questions (optimal value should be 1.0) and the higher this value the better the model can detect duplicates

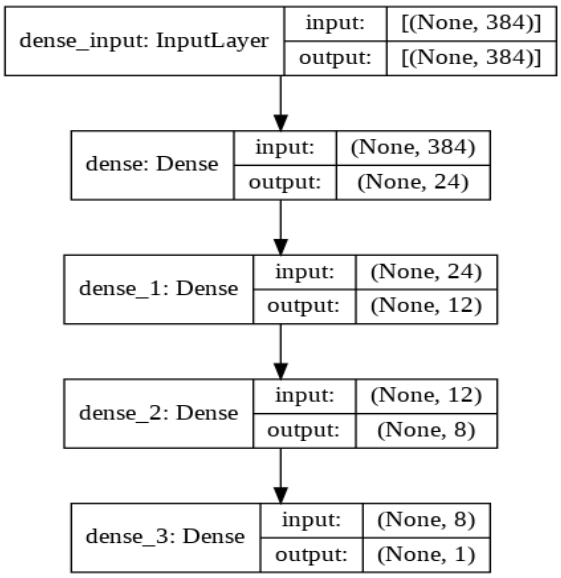
- The same for duplicates but done on non-duplicates and the model with lower similarity on non-duplicates the better it is.

```
Model Confidence of the duplicates (paraphrase_MiniLM_cosine) = 0.8479464698954626
Model Confidence of the non-duplicates (paraphrase_MiniLM_cosine) = 0.5580064015439138


Model Confidence of the duplicates (stsb_mpnet_cosine) = 0.8402853347281994
Model Confidence of the non-duplicates (stsb_mpnet_cosine) = 0.5479525331895995


Model Confidence of the duplicates (paraphrase_mpnet_cosine) = 0.8854321176443257
Model Confidence of the non-duplicates (paraphrase_mpnet_cosine) = 0.5785634213629066
```

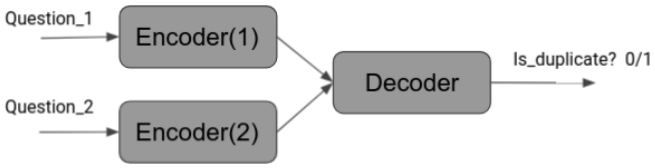
III. Model Architecture

A. Deep Dense Model

- Using “paraphrase-MiniLM-L3-v2” as sentence encoder for question1 & question2
- Multiply both vectors then input in NN
- Fully connected Dense Layers 4 layers (24, 12 , 8 , 1) “Relu”

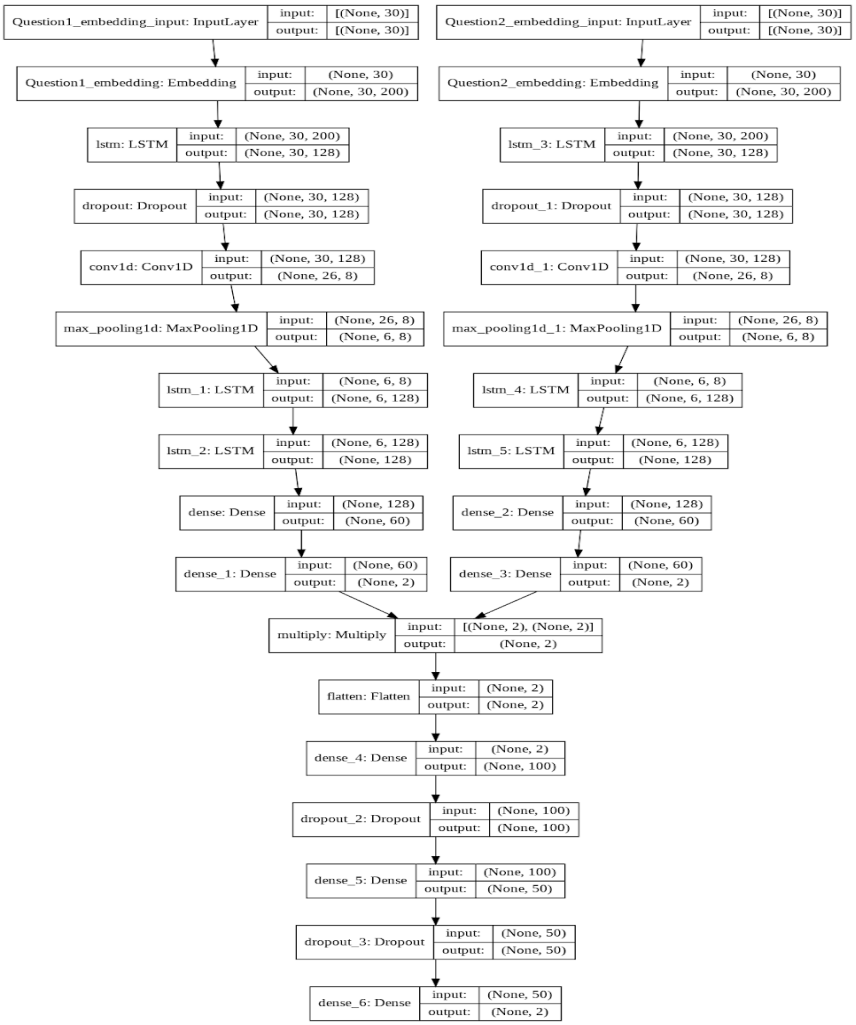


B. Deeper Model (LSTM+CNN)



Here I used the following concepts

- Seperated the embedding layers for questions 1 & 2
- Build separate encoding pipeline for questions 1 & 2
- Aggregate the two encodings in one classification decoder

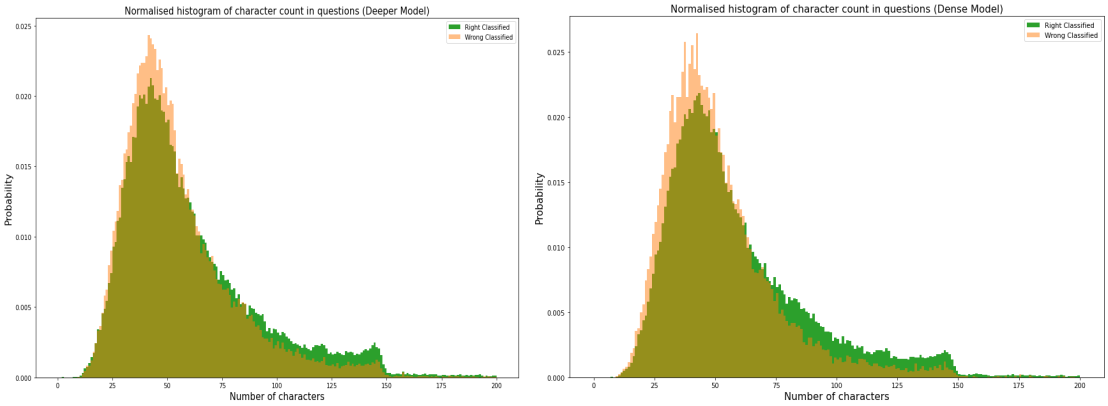


- Layers used
    - LSTM (Recurrent layer for capturing dependency parsing)
    - 1D CNN & 1D Max pooling for highlighting the important parts in the text for classification
    - Dropout as model regularization
    - Dense Layers for encoding/decoding aggregations

IV. Error Analysis

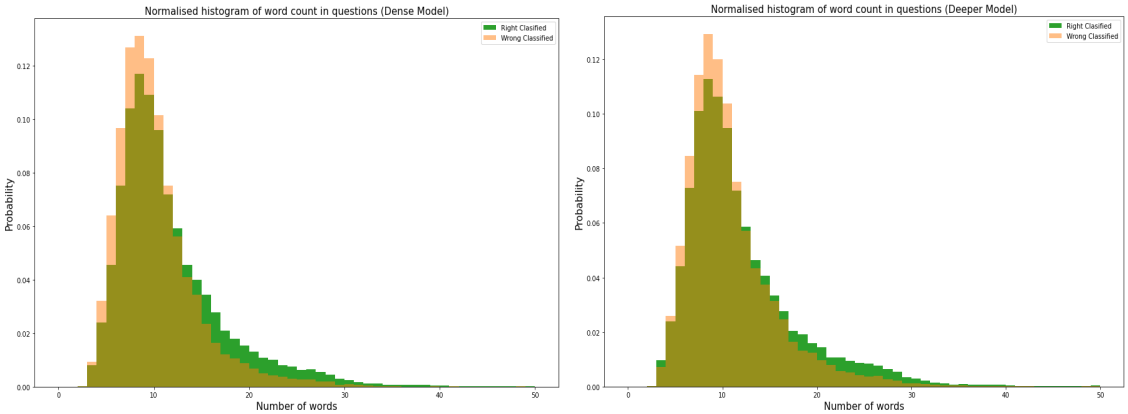
Started to dig deeper in the model misclassified outputs to know more what could be the causes in the data of misclassified vs good classified.

- First analysis on the number of characters in the two questions concatenated, comparing the misclassified and good classified.



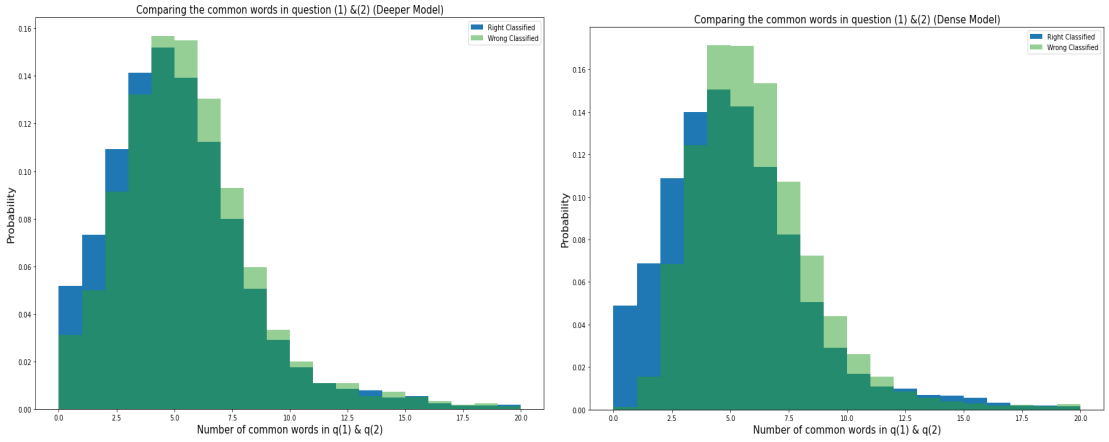
We found that the right classified samples are more length and tend to have more samples with more characters.

- Second analysis on the same as above but on the words count which could be using fewer characters but many descriptive words, this will help us know more about the misclassified samples.



The same found as the first analysis that right classified samples tend to have more words to be more descriptive, and this means that by being more descriptive in the question text you can easily identify whether this question is a duplicate or not, so it will be easier for the model.

- Third analysis on the common words, detecting the number of words/characters wasn't enough for us to know more about the misclassified samples. Because it may be lengthy and descriptive but also the model can fail to detect it is duplicate because there are many common words between the two questions.

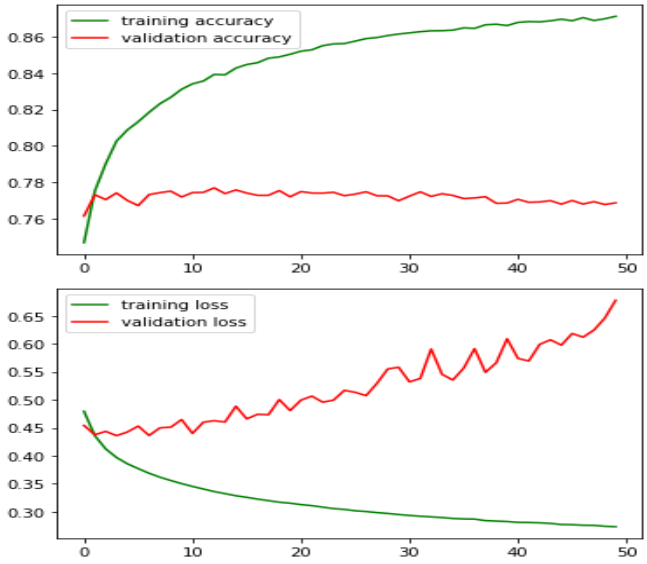


Here We can easily see that the misclassified samples have more common words than the good classified. That's quite reasonable when you have many common words in your questions and just one is different that changes the whole meaning. This will make it so hard on the model to detect those tiny differences.

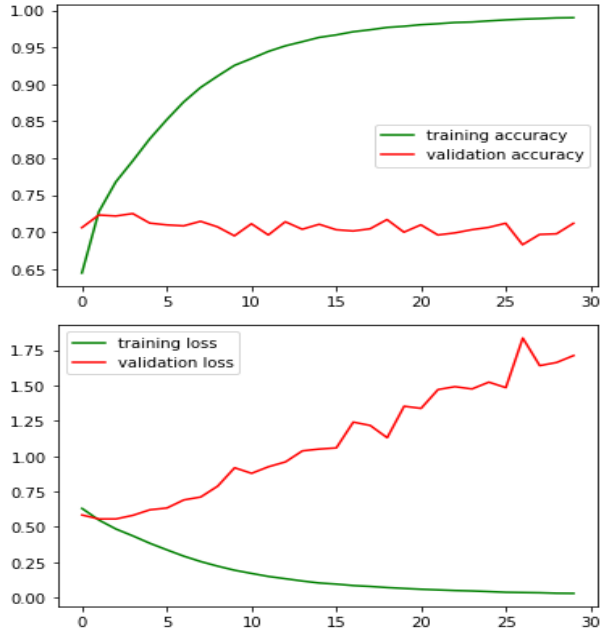
V. Model Evaluation

Evaluating the model while training on training and validating. We can see clearly that it faces an **overfitting problem** which can be solved in the future by more advanced regularization methods.

- Deep Dense Model training Evaluation on 50 Epochs



- Deeper LSTM+CNN Model training Evaluation on 30 Epochs



- Accuracy , Recall & Precision Metrics on the Test data

Dense Model on Test Data

	precision	recall	f1-score	support
0	0.82	0.85	0.83	12561
1	0.73	0.68	0.71	7439
accuracy			0.79	20000
macro avg	0.77	0.77	0.77	20000
weighted avg	0.79	0.79	0.79	20000

Deeper Model on Test Data

	precision	recall	f1-score	support
0	0.79	0.72	0.75	12639
1	0.58	0.67	0.62	7361
accuracy			0.70	20000
macro avg	0.68	0.69	0.69	20000
weighted avg	0.71	0.70	0.70	20000

Final Results

After running over all experiments and different approaches here are the aggregated results between different models and different feature representations.

Supervised Learning					
Model	**Feature represntation**	**Train accuracy**	**Test accuracy**	**Test Precision**	**Test Recall**
Decision Tree	TFIDF	71.24%	67.18%	64%	62%
	Word2Vec	65.60%	63.37%	60%	59%
RandomForest	TFIDF	84.60%	71.52%	72%	65%
	Word2Vec	81.52%	66.60%	64%	60%
Logistic Regression	TFIDF	66.40%	66.44%	64%	64%
	LDA	65.77%	65.76%	63%	62%
KNN	TFIDF	79.33%	68.89%	67%	67%
SVM	TFIDF	75.51%	72.22%	71%	67%
Deep Dense NN	MiniLM - Sentence Transformer	87.53%	79%	77	77
LSTM + CNN Deep NN	GloVe	98.99%	71%	69%	69%

Unsupervised Learning (Mini-batch Kmeans)				
Feature represntation	**Kappa**	**Silhoutte**	**Completness**	**Homogeneity**
TFIDF Differences	0	0.08	0.01	0.01
MiniLM - Sentence Transformer	0	0.17	0.11	0
Word2Vec Cosine Similarity	-0.28	0.64	0.11	0.08
Word2Vec Differences	0.21	0.2	0.05	0.05

We decided to choose the Deep Dense NN model as our champion model for this competition and submit it to the Kaggle competition.

Why do we choose Deep Dense NN as our champion model?

- **Because it is the most generalized model over all other models**
- **It achieves the highest test accuracy , precision & recall**

Innovativeness

In this project, we have tried multiple methods from choosing the feature extraction method to the algorithms. We have tried TFIDF, word2vec differences, multiplication and cosine similarities and word embeddings. We have also tried **Minkowski Distance, Euclidean Distance , Cosine Similarity** and used all of these as an input to the **LDA** model which is used as a dimensionality reduction method.

We have also tried clustering as a method to solve the problem which is not a common thing for this problem. And got the best results from using “paraphrase-MiniLM-L3-v2” to extract 384 dimensional dense vector space.

In Deep Learning Models, We used a new architecture of two **encoders (encoder1 & encoder2)** . This was a novel idea that came to our mind on how we can encode different questions to different spaces and make the model differentiate while learning.
Another innovative idea we used is using **LSTM and 1D CNN** in an NLP task, it isn’t commonly used to use CNN in NLP problems as CNN is more popular with computer vision tasks, but it was a quite interesting idea that we can leverage the idea of CNN filtering and highlighting the text to the LSTM to focus on specific parts it acts similar to attention layers in transformers as well.

Our final loss reached 0.18 which is considered a high score. We have also introduced an inference model that could be run by a simple command.

We made a code for the inference model with all the model weights and you can run it by a simple command and ready to go. Please check our Github repo for more instructions about it.
Github Repo : https://github.com/sharafcode/quora-question-pairs