Quora Duplicate Question Detection

Course Instructor: Dr. Vineet Gandhi

Mentor: Apurva Jadhav

By
Ayush Kumar Rai (2020201087)
Chaitanya Kumar P (2020201012)
Neha Agarwal (2020201095)

TEAM ACN

Objective

Our Aim is to detect the text similarity between two texts using the quora dataset.

Identify which questions asked on Quora are duplicates of questions that have already been asked.

In this project we'll explore different kinds of Models that efficiently classifies which pair of questions are duplicate and which are not. In a way we are trying to mimic human judgement with the help of machine learning.

If a new question is asked and if it's similar part already exists then we can instantly provide answer with the help of our Model.

Set up

Tools and Technologies:

- Python
- Numpy
- Pandas
- Scipy
- Scikit-learn
- Tensorflow
- Keras
- Google Colab notebooks

Data Exploration

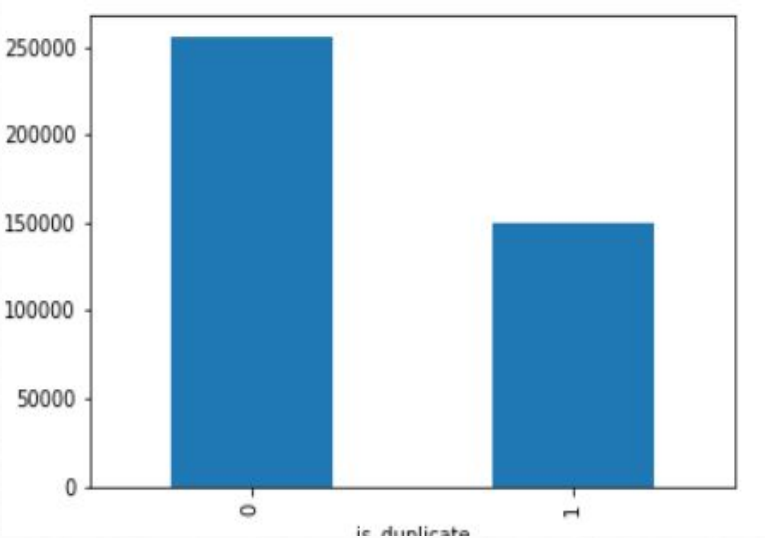
Data set was taken from Kaggle competition page.
Currently it has 404,351 samples of which some of the feature columns have null values in them.

Description of Data set:

- **ID** : The id of the question pair.
- **QID 1** : Unique id of question 1.
- **QID 2** : Unique id of question 2.
- **Question 1** : Text data of question 1.
- **Question 2** : Text data of question 2.
- **Is_duplicate** : If question 1 and question 2 has same meaning then it is 1 otherwise 0.

Data Description cont'd...

We can see in the below image that 61.8 percent of question pairs were not duplicate (i.e is_duplicate = 0) remaining percent were duplicate question pairs.



Data Preprocessing

Firstly, the rows whose one of the features containing NULL values were dropped. Since they are minimal in number. We decided to drop them.

Using regular expressions punctuation marks, some symbols like (, } ,] etc.. are removed.

Sentences such as I'am , I've,don't are replaced by their full forms to capture the actual meaning meaning.

Using Beautifulsoup library html and Xml tags were removed.

Removal of Stop Words are decided based on the Model we are employing.

Non Ascii characters were ignored.

Models

Three kinds of Models were explored in this paper which are as follows:

1. Linear Models
2. Tree Based Models
3. Neural Network Models

Linear Models

——— In the Linear Models all the experiments listed in the paper were explored.

1. Logistic regression on :-

   i) Unigrams

   ii) Bigrams

   iii)Trigrams

2. SVM on :-

   i) Unigrams

   ii) Bigrams

   iii) Trigrams

Logistic Regression:

In the logistic regression apart from unigrams, bigrams and trigrams with some default parameters; hyperparameter tuning is done on Learning rates and maximum iterations.

We used SGDClassifier with loss = ‘log’ which essentially is logistic regression.

Some fixed parameters were given for applying the model on skipgrams.

They are l2 regularisation alpha = 0.00001 and max_iter =20. These were applied on all the all the different skipgrams.

For obtaining unigrams, bigrams and trigrams, we used tf-idf Vectorizer with varying ngram_range.

Dataset is split in the ratio of 60:20:10 for train,validation and test as mentioned in the paper.

Observations

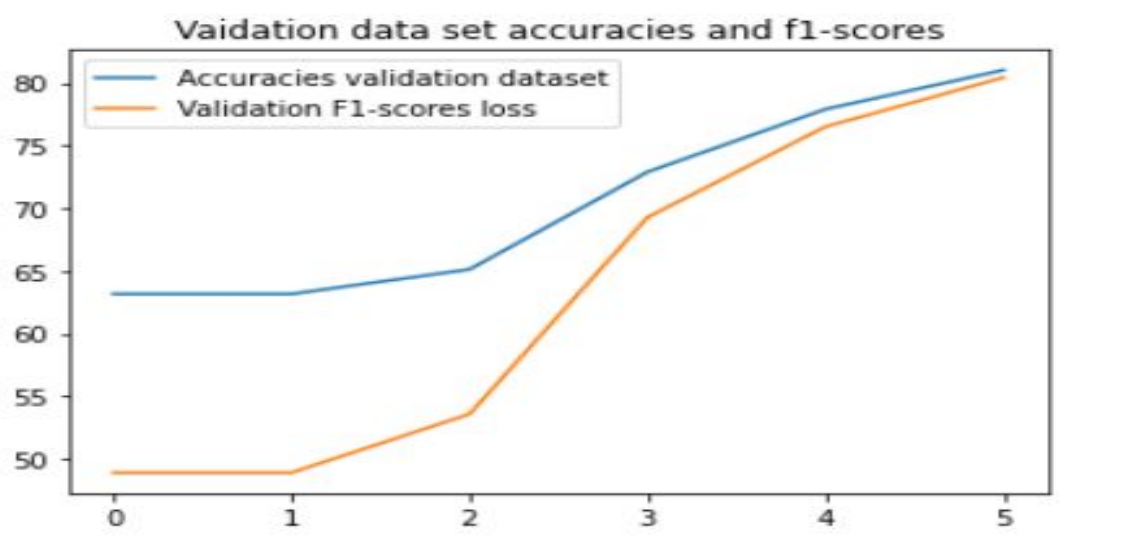
Comparison of Accuracies and f1_scores:

Logistic Regression	Accuracies of our models	Accuracies in the paper
Unigrams	75.55	75.4
Bigrams	77.96	79.5
Trigrams	78.7	80.8

Logistic Regression	f1_score Of our models	f1_score in the paper
Unigrams	74.33	63.8
Bigrams	76.73	70.6
Trigrams	76.87	71.8

HyperParameter Tuning-Learning rate

As mentioned in the paper we experimented with several alpha values which are 0.1,0.01,0.001,0.0001,0.00001,0.000001. Corresponding accuracies and f1_scores are shown in the graph below:



Comparison:

Our Models on Trigrams with max_iter=20

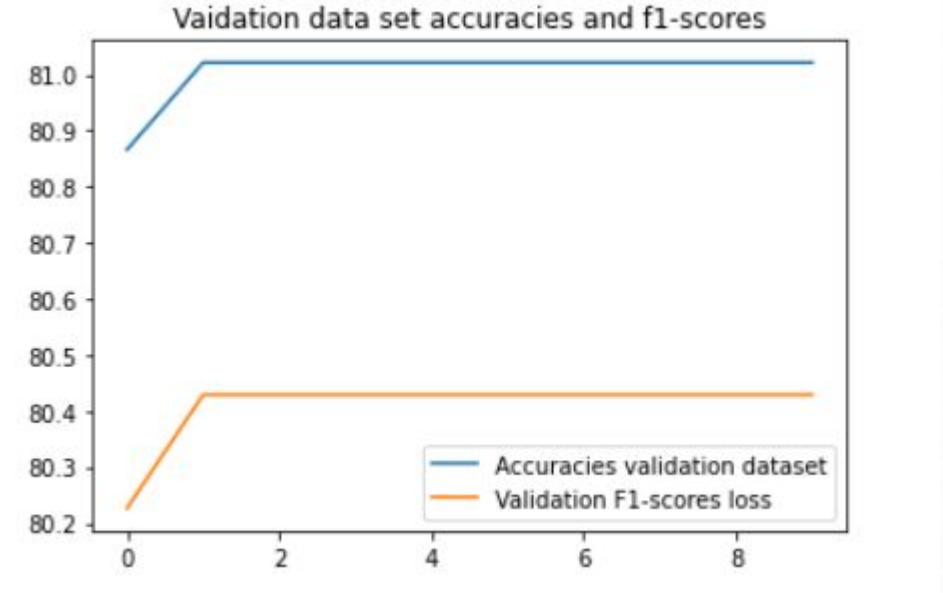
Regularization weight	Validation Set Accuracy%	F-Scores on validation dataset
0.1	63.168597062500865	48.90979914862098
0.01	63.168597062500865	48.90979914862098
0.001	65.12372394652598	53.58285073940648
0.0001	72.8975172773862	69.27584453376078
1e-05	77.90280697415606	76.50678686454503
1e-06	81.02166714754819	80.42902814274474

Trigram Model from the given paper

Regularization weight	Validation Set Results Accuracy (%)	F-score
0.1	64.0	5.7
0.01	71.1	45.9
0.001	75.5	59.5
0.0001	78.2	68.2
0.00001	**80.7**	71.5
0.000001	80.0	**71.6**

HyperParameter Tuning-max_iter

As mentioned in the paper, we experimented with several iterations values which are 5,10,15,20,25,30,35,40,45,50. Corresponding accuracies and f1_scores are shown in the graph below:



Comparison:

Our Models on Trigrams with alpha =0.000001

Number of iterations	Validation Set Accuracy%	F-Scores on validation dataset
5	80.86641111248505	80.22656445420743
10	81.02166714754819	80.42902814274474
15	81.02166714754819	80.42902814274474
20	81.02166714754819	80.42902814274474
25	81.02166714754819	80.42902814274474
30	81.02166714754819	80.42902814274474
35	81.02166714754819	80.42902814274474
40	81.02166714754819	80.42902814274474
45	81.02166714754819	80.42902814274474
50	81.02166714754819	80.42902814274474

Trigram Model from the given paper

	Validation Set Results	
Number of iterations	**Accuracy (%)**	**F-score**
5	79.5	71.4
10	80.3	71.3
15	80.0	**72.3**
20	**80.6**	71.5
25	**80.6**	71.8
30	80.4	72.2
35	80.3	72.1
40	**80.6**	72.0
45	80.5	71.1
50	80.5	72.1

Observation

As we can see from the above experiments results are very similar to those mentioned in the paper.

In some cases like alpha =0.000001 our model has produced better results when compared to the one given in the paper.

In hyper parameter tuning of max-iterations the accuracy remained constant after 10 iterations and is better as compared to the paper's with alpha = 0.000001.

On test data with alpha = 0.000001 and max_iter = 30 the accuracy of our model is 81.24 which is higher than that given in the paper which is 80.7

SVM

We use scikit-learn's SVM implementation for our experiments. We ran SVM on unigrams, bigrams and trigrams.

Parameters for the model were set to--

C (penalty parameter C of the error term): **1.0**
Kernel (kernel type used in feature matrix): **Linear**
max iter (maximum number of iterations: **no limit**

Accuracy for different ngrams-

N- grams	Accuracy (%) of our model	Accuracy (%) in paper
Unigrams	74.55	75.5
Bigrams	77.97	79
Trigrams	79.91	80.5

Tree Based Models

Three models are given in the paper and implemented the same.

It is evident that we cannot use the unigrams, bigrams, trigrams that we used in the Linear Models since the dimensions are greater than 120,000. The tree models become inefficient for such high dimensions.

As given in the paper we need to Engineer new features out of the textual data.

We used three models on several different features and took average of them.

The models are:

- Decision Tree classifier with Max_depth = 10, min_samples_per_leaf =10.
- Random Forest with Max_depth = None, min_samples_per_leaf =5, num_estimators = 50
- Gradient Boost classifier with Max_depth = 4 and no_estimators = 500.

Feature Engineering

Following are the extracted features:

For the below features we need to make use of stop words so the preprocessing is minimal in these models.

1. **L** : Length for question 1 (l1) and question 2 (l2) , difference of those lengths (l1-l2) and ratio of those lengths (l1/l2).
2. **LC** :Number of common lower cased words,count/length of longest sentence
3. **LCXS** :Number of common lower cased words excluding the stop words
4. **LW** : 1 if last word is same otherwise 0.
5. **CAP** : Common CAPITALIZED words.
6. **PRE** : Common prefixes among words in each question pair of lengths 3-6.
7. **MISCELLANEOUS** : Some features such as whether both questions contain ‘not’ in them, both questions contain some common digits.

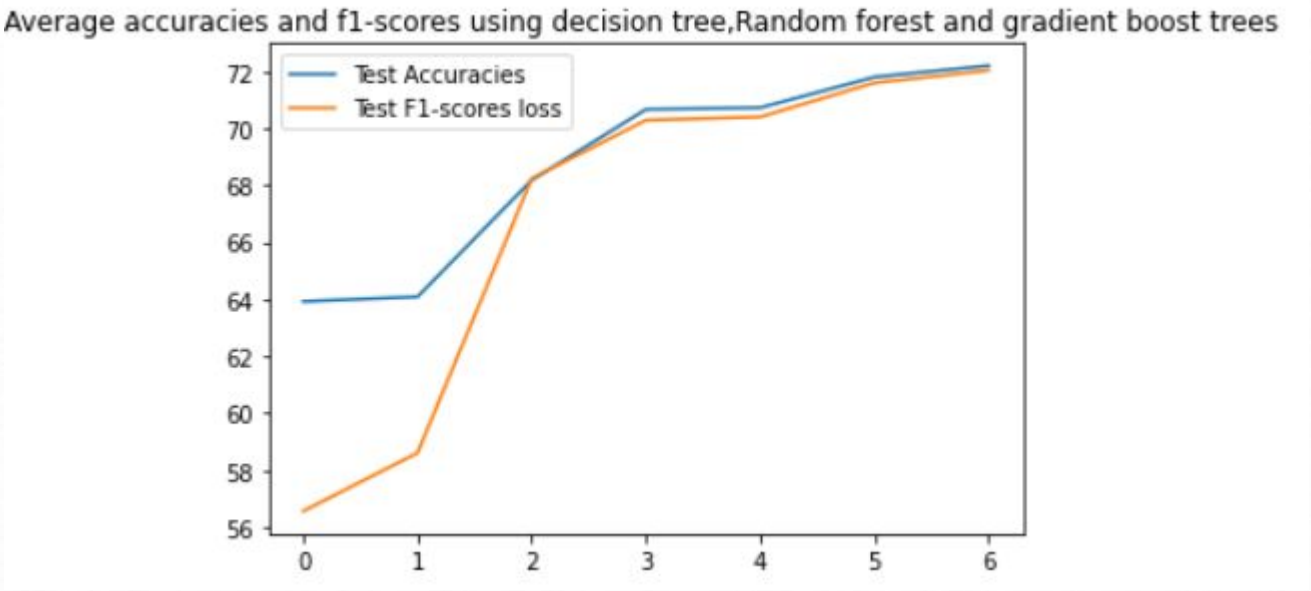
Feature Engineering Cont'd..

We incremented number of features by including new features for every set as shown below.

Feature set		Number of Features
• L 4	-	4
• L, LC	-	6
• L, LC, LCXS	-	8
• L, LC, LCXS, LW	-	9
• L, LC, LCXS, LW, CAP	-	11
• L, LC, LCXS, LW, CAP, PRE	-	19
• L, LC, LCXS, LW, CAP, PRE, M	-	21

Observations:

Below Figure shows the accuracies and f1-scores of the different Feature sets:



Comparison:

Our Model on different feature sets

Feature Set	Average Accuracy%	Average F-Scores
L	63.919999999999995	56.556666666666665
L,LC	64.08333333333333	58.60333333333333
L,LC,LCXS	68.17666666666666	68.23666666666666
L,LC,LCXS,LW	70.66333333333333	70.28333333333335
L,LC,LCXS,LW,CAP	70.72	70.39666666666666
L,LC,LCXS,LW,CAP,PRE	71.79666666666667	71.59333333333335
L,LC,LCXS,LW,CAP,PRE,Misc	72.19666666666667	72.03

Paper's model

		Validation Set Results		
Features	Num features	Acc (%)	Δ acc	F-score
Majority classL	0	63.1	-	-
L	4	63.7	0.6	30.7
L, LC	6	68.5	**4.8**	59.8
L, LC, LCXS	8	70.7	**2.2**	63.3
L, LC, LCXS, LW	9	72.7	2.0	63.6
L, LC, LCXS, LW, CAP	11	72.8	0.1	64.5
L, LC, LCXS, LW, CAP, PRE	19	73.2	0.4	64.8
L, LC, LCXS, LW, CAP, PRE, M	25	74.6	**1.5**	66.3

From the above comparison, we can conclude that the accuracies are almost similar to that in the paper and f1-scores are higher in our experiments as compared to the models given in the paper.

Neural Network Models

We experimented with the following models that are given in the Paper.

Most of the Models given in the paper are slight variations of LSTM and the accuracies are almost similar.

So, we included Continuous Bag Of Words Model and LSTM model.

CBOW Model is simple 3 Layer Dense MLP model yet it gave the best accuracy both in the paper and in our experiment with it.

For the above models we need to use Glove 840B 300 dimensional embeddings to obtains the numerical representations of the questions.

Word Embeddings

— — —

Glove Pretrained:

- 2.2 Million words
- 840 Billion Tokens’
- 300 dimensional embeddings

Feature Engineering

We cannot use the skipgrams since they are very high dimensional, the time taken to train neural network models on such high dimensions is very high. So, we need to engineer some features by making use of GLove embeddings.

Initially the an embedding index dictionary is developed from the text file of glove embeddings which contains each word as the key and the 300 dimensional vectors representation of each words.

As mentioned in the paper, vector representation of each question is sum of embeddings of each word in it.

q1_feats and q2_feats contain the vector representation of question 1 and question 2.

We concatenated the q1_feats, q2_feats, difference and product of q1_feats, q2_feats to obtain the final features.

Continuous Bag of words (CBOW)

The features were split in 80:20 ratio for train and test data.

The vector representations thus obtained from above feature engineering is fed into 3 Layer dense MLP followed by softmax classifier.

Model was compiled with Adam optimizer with sparse_categorical_cross entropy and other default parameters.

```
model = Sequential()
model.add(Dense(200,input_dim=1200))
model.add(Activation('relu'))
model.add(Dropout(0.1))
model.add(Dense(100))
model.add(Activation('relu'))
model.add(Dropout(0.1))
model.add(Dense(50))
model.add(Activation('relu'))
model.add(Dropout(0.1))
model.add(Dense(2))
model.add(Activation('softmax'))
```

Observations:

CBOW model produced good accuracy on test data which is 80.616 which is very close to the accuracy given in the paper which is 83.4

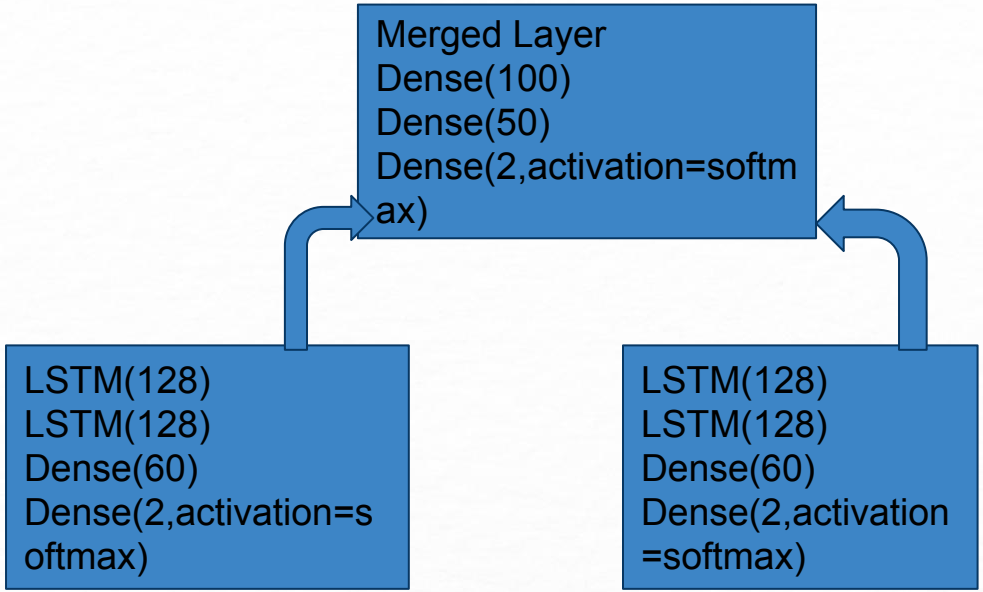
F1-score was higher compared to the one given in the paper :

F1-score from our model = 80.64

F1-score given was 77.8

LSTM

Below is the architecture of LSTM:



LSTM cont'd

Differences between CBOW and LSTM are:

- We used LSTM with RNN based Neural network followed by some Dense layers.
- In the CBOW the vector representation were not passed through tanh layer as in LSTM.
- In LSTM first layer contains the tanh activation function as mentioned in the paper.
- Each question is trained on separate LSTM model and the both the models are merged as shown in the architecture.

Last layer was softmax with out_dimensions as 2 for all neural net based models.

Observations

Our model accuracy = 75.47

Given accuracy in paper = 81.4

F1-score of our model = 75.52

F1-score given = 75.4

F1-scores are almost same for both the models.

Accuracy was bit low compared to the model given in the paper.

Contributions

Individual Contributions:

Roll Number	Name	Contribution
2020201087	Ayush Kumar Rai	SVM, Logistic regression, Vector representation of text.
2020201012	P.Chaitanya Kumar	Tree Models,CBOW, Feature engineering for Tree based Models.
2020201095	Neha Agarwal	LSTM, Data Preprocessing, Feature engineering for Neural network Models

Conclusion

As observed in all the above experiments the accuracies are very close to the ones given in the paper and in most of the cases the f1-scores are higher in our Models.

Linear Model(Logistic Regression) with optimal hyper parameters performed well compared to the tree models.

SVM takes more time to train the model which is approximately around 20 hours.

Logistic regression and tree models with separate features are fast.

CBOW takes less time to train compared to LSTM and produced high accuracy.

Thank you!