Paraphrase Identification using Supervised Machine Learning Techniques

Mayanka Verma, Nanditha Sundararajan
{verma.m,sundararajan.n}@husky.neu.edu
CS6120 NLP Fall 2019,Northeastern University

1. Introduction

In our daily life, we can use different words and phrases to express the same meaning. This is related to our knowledge and cultural habits, that later is reflected in our verbal and written communications.

Platforms like Stack Overflow and Quora are one of the largest knowledge-based text repositories where people share information in question answer format. On such platforms, multiple questions with same intent can cause readers to spend more time finding the best answer to their questions, and make writers answer multiple versions of the same question.

Quora has around 300 million users across the globe. There is a high probability that people will ask similar questions worded differently. For example: “How do I read and find my YouTube comments?” and “How can I see all my YouTube comments?” are duplicate questions because they both have the same intent and therefore, should be answered only once.

In our project, we have built a computational model to capture semantic similarity between Quora question pairs to classify them as duplicates or non-duplicates to enhance the user experience by faster response retrieval.

2. Related Work

This project was motivated by the Quora-Question Pairs[1] competition hosted by Kaggle community. Earlier works related to this project involved extracting features from the data provided and constructing models like Logistic Regression, XGBoost etc.

In addition to the above mentioned work, we ran different versions of feature-based models using the subsets of features extracted from the dataset. Also, we constructed a deep neural model, ’Manhattan LSTM’ to capture the semantic relationships between the question pairs better.

3. Dataset

Quora Question Pairs[1] corpus was chosen from Kaggle. The dataset has 404351 Quora Question pairs. There are total 6 columns in the dataset:

- id - the distinct id of each question pair
- qid1, qid2 - distinct ids of each question
- question1, question2 - textual description of each question.
- is_duplicate - binary response variable which is set to 1 if the question pairs have same intent else set to 0.

The split of duplicate and non-duplicate question pairs is 37% and 63% respectively. The training dataset was manually labelled as duplicates or non-duplicates by humans therefore it is subjective and prone to noise.

4. Pre-processing

4.1 Text Pre-processing

We have implemented tokenization followed by conversion to lower case. We removed the frequently occurring stop words using NLTK. This ensured that our result is not affected by frequently occurring words and helped us save on computation time. We performed porter stemming to help our ML model to learn how to extract intent form a sequence of words with the same meaning but different forms. We removed repeating whitespace characters[2]. Also, we expanded the contracted words to standardize our text for example, don’t was expanded to do not.

4.2 Exploratory Analysis

We performed various exploratory data analysis to better understand our dataset and the extracted features to help us identify the important features for separation of the classes.

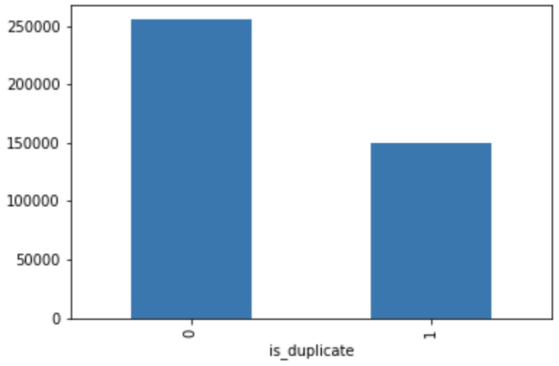

Fig:1:Data Distribution across labels

As we can see that the split of duplicate and non-duplicate classes in the dataset is ‘37:63’. None of the classes are under-represented and therefore, we will not have to account for oversampling.

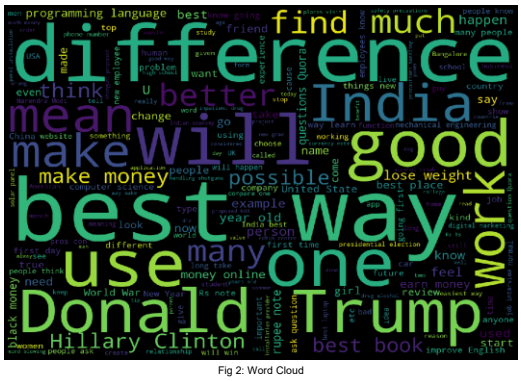

Fig 2: Word Cloud

We constructed the word cloud[3] based on the Quora question pairs to identify the most frequently used words on which the questions are asked on Quora. We can see that people generally post the questions to understand the difference between things or to find the best way.

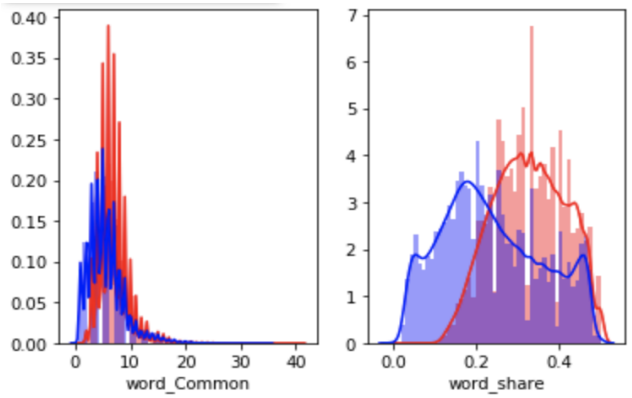

Fig:3:Distribution of word_Common and word_share features across labels

Word_common is the number of common unique words in Question 1 and Question 2. The distributions of the word_Common feature in duplicate and non-duplicate questions are highly overlapping and therefore does not seem as a good feature, contrary to our initial understanding.

On the other hand, word_share (common words/total number of words) will have a good predictive power, as the classes seem separable from the above distribution.

4.3 Feature Engineering

We have extracted 3 different sets of features used for feature-based models and GloVe Embeddings for feeding to our neural model.

4.3.1 Basic Text Features

We extracted basic text features like length information (character and word length) of q1 and q2. Also, absolute difference of their lengths were calculated. We got the common word intersection count between the question pairs as features. We calculated the 'word share' feature as common words/total number of words.

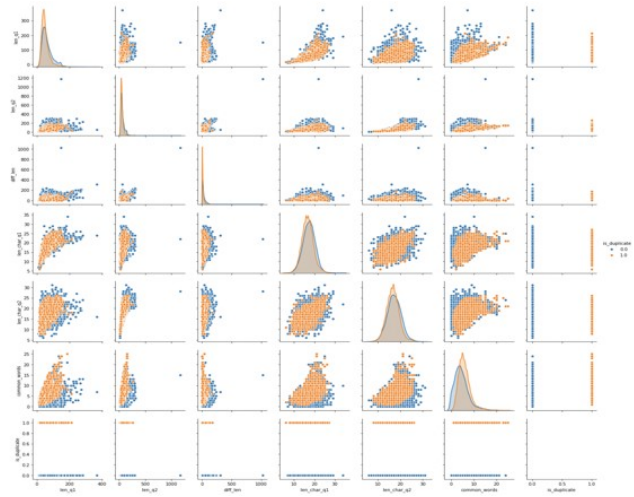

Fig: 4:Scatter pair-plot of basic features

We plotted a scatter pair-plot of the basic features extracted from the dataset and analysed the distribution of the data for both duplicate and non-duplicate question pairs. In the above image, blue points represent the duplicate pairs and orange as non-duplicates.

The duplicate question pairs seem to have a similar range of characters in them. Also there is a strong correlation in the number of words and the number of characters in a question.

Looking at the distribution of the basic features extracted, we can say that they do not linearly separate the classes and therefore, will not serve as good features. This can also be confirmed from the implementation of feature-based models using only basic features where we achieved an accuracy significantly lower than other versions of our models.

4.3.2 Word Embeddings based features using Word2Vev

Word2Vec pretrained on Google News[4] was used to obtain word embeddings for q1 and q2. Using the word embeddings, we calculated WMD[5] (Word to Mover distance) and Normalized WMD. WMD measures the minimum distance that the Word2Vec embeddings of one question need to travel to reach the embedded words in the other question.

We also computed pairwise distance metrics like cosine, Manhattan, Jaccard, Euclidean and Minkowski using the word2vec embeddings as similarity features[6].

Also, skewness (feature to measure lack of symmetry in word vectors) and kurtosis (feature to measure whether the data are heavy-tailed or light-tailed relative to a normal distribution) were extracted from word embeddings as features.

4.3.3 Fuzzy Features

Features based on fuzzy string matching were extracted[7]. Fuzzy string matching is used for string pattern matching. It is based on number of edits required to convert one sequence into another. Python's Fuzzywuzzy library was used for this.

Fuzzy features like Simple and Partial Ratios, Token Set and Sort Ratio, Partial Set and Sort Ratio. As observed, partial ratio features allows easier discrimination between the binary classes.

4.3.4 GloVe for deep neural models

GloVe[8] stands for Global vectors for word representation. It is an unsupervised learning method for generating word embeddings by aggregating global word-word co-occurrence matrix from corpus.

GloVe not only incorporates the local context information of words but also considers global statistics such as the word co-occurrences which in turn results in capturing the sentence semantics in an accurate manner. This will be used while constructing the inputs for our deep neural model.

5. Experiments

We split the data (feature matrix and response label) into 70:10:20 training, validation and test sets respectively.

5.1 Feature-based Classifier

As an initial step, we implemented Logistic Regression and Random Forest. For each of the classifier, we used "RandomizedSearchCV" for 5 folds for hyperparameter tuning.

We implemented multiple versions of both Logistic Regression and Random Forest models by taking subsets of the extracted features. For example, Logistic Regression was implemented using basic features, fuzzy features, Word2Vec based word embedding features and also their numerous combinations. This was done to perform feature selection.

Although, XGBoost might give a better result in comparison to the above mentioned models, we are not implementing the same as it has been frequently used in related works.

5.2 Deep Neural model-based Classifier

Given the large number of data entries, we can go for neural networks to achieve our goal of identifying duplicate question pairs. Since we have sequential data , we can use LSTMs[9] for sequence modelling than Recurrent Neural Network (RNN) as the latter is weak in learning long-term dependencies.

For measuring the 'similarity' between the two questions, we have implemented MaLSTM (Manhattan LSTM)- a Siamese deep network model. Siamese neural network[10] is an architecture that contains two identical sub-networks joined at the outputs and is widely used in tasks that involve finding the similarity between two comparable patterns. The Manhattan LSTM[11] model utilizes the Siamese architecture, where there are two identical LSTM sub-networks with tied weights (each processing one of the questions) that learn the mappings from the variable-length input sequences to the fixed-length vector representations.

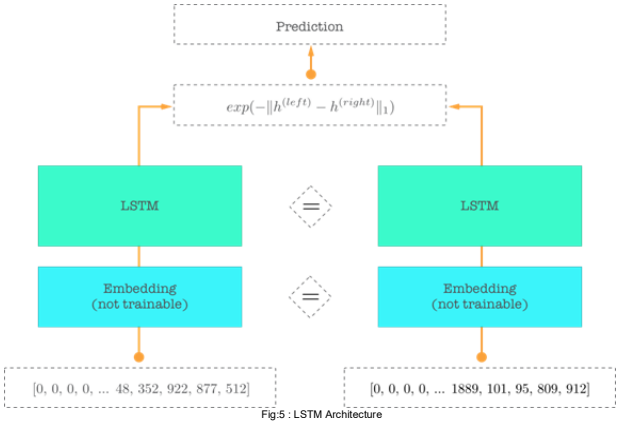

Fig:5 : LSTM Architecture

The vector representation of a question in the input question pair is passed to the embedding layer from each of the input layer.

This utilises the GloVe to generate sentence embeddings for the inputs and is fed into the LSTMs. The fixed-length vectors generated by each of the LSTMs, which contain the semantic meaning of the questions, are fed into a similarity function that uses the Manhattan distance metric (or L1 norm) between them as shown below. The Manhattan distance is used to evaluate the similarity as it forces the model to entirely capture the semantic differences between the two input questions.

$$exp(-\|h^{(left)} - h^{(right)}\|_1)$$

Since we have an exponent function applied to the negative of the distance metric calculated, the output/prediction will have values 0 or 1.
While training the model on the training set, we get:
Total Parameters: 58,379,701 parameters
Trainable Parameters:87,532 parameters
Non-Trainable Parameters: 58,292,169 parameters
This is the advantage of deep learning models over feature-based models. The deep learning model performs end-to-end model building and has more parameters to train on.

6. Result and Evaluation

The hyperparameter tuned feature-based models is first trained on the training set and then evaluated on the validation set. We are using Precision, Recall, F1 Score, ROC along with accuracy to evaluate our implemented models.

Precision is an important metric for the given scenario as we do not want questions to get tagged to incorrect answers. This requires a model with low false positives (high precision).

For performing feature selection, we implemented our feature-based models using the various subset combinations of the extracted features. Accuracy was taken as the evaluation metric for performing feature selection.

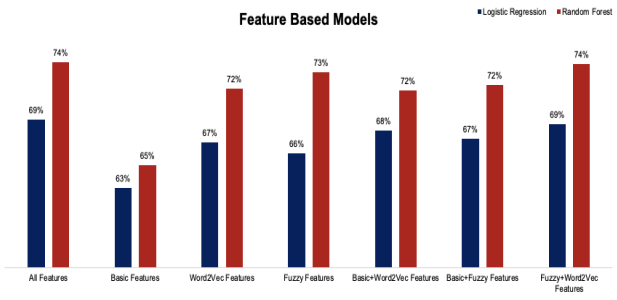

Fig: 6: Distribution of accuracies for models across different subsets of features

As seen from the above image, the accuracies achieved by implementing models using only basic features based on word counts and common word counts (basic features) were significantly lower in comparison to the accuracies achieved using other feature sets. This confirms that basic features

are not good for classifying question pairs as duplicates or non-duplicates. Also, the accuracies achieved using all features and just fuzzy and Word2Vec features were same. This tells us that addition of basic features was not improving the model performance.

For performing dimensionality reduction and to ensure simplicity of the model, our selected feature-based models includes only fuzzy and word2vec features as we get the same results as we get with all the features in terms of the evaluation metrics selected.

6.1 Performance Evaluation on Validation Set

For the model evaluation of feature-based models we selected the versions with only Fuzzy and Word2Vec word embedding features, as we achieved similar accuracies and F1 Score as that of the models with all the features and it was the highest amongst all other versions. However, the performance of the feature-based classifiers were not high enough and to achieve a better performance we implemented Siamese Manhattan LSTM to better capture the semantic similarity. The above 3 models were validated on validation set. The best values for hyperparameters such as number of hidden layers, loss and optimizer were as mentioned below:
Number of hidden layers: 100
Loss function: binary_crossentropy
Optimiser: adam.

	Accuracy	Precision	Recall	F1-Score	ROC
Logistic Regression using Fuzzy+Word2Vec Embeddings	69%	58%	58%	58%	77%
Random Forest using Fuzzy+Word2Vec Embeddings	74%	62%	72%	68%	83%
Siamese Manhattan LSTM	82%	71%	76%	74%	88%

Fig 7: :Evaluation Metris for Model Evaluation on Validation Set

Amongst feature-based classifiers, we can see that Random Forest gave better predictability in comparison to Logistic. The linear separability of the model was significantly higher along with accuracy and other evaluation metrics reported above. However, Siamese MaLSTM gave the highest performance amongst all the models implemented with a ROC of 0.88 i.e. the neural model was more capable of identifying questions with same intent.

6.2 Scoring of MaLSTM on Test Set

As per our expectations, Manhattan LSTM (MaLSTM) gave the best performance on the validation set. Therefore, MaLSTM was chosen as our final model and was scored on the test set. Time taken for the model to train was 6 hours.

Accuracy	Precision	Recall	F1-Score	ROC
81%	71%	80%	75%	89%

Fig 8: :Evaluation Metris for MaLSTM on Test

We achieved an accuracy of 81% on the test set. We were able to attain sensitivity/recall of 80%. However, precision achieved was 71%. The linear separability of the model is good as we have achieved an ROC of 89%.

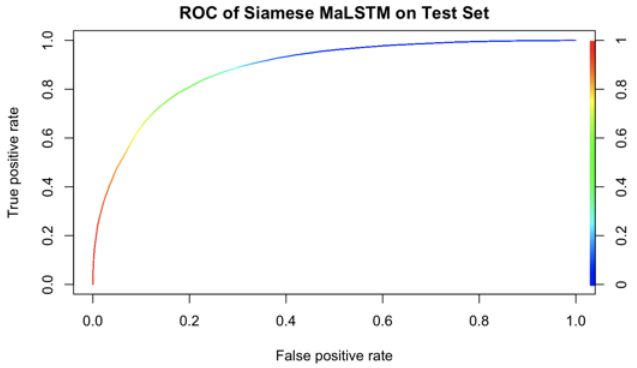

Fig 9: : ROC curve for MaLSTM classifier on Test Set

7. Conclusions

We have introduced the Paraphrase Identification problem as well as the properties of the Quora Question Pair dataset. We have elaborated on the feature engineering process of extracting sentence level and word level features. We ran various versions of feature-based models on different subsets of features and identified that Fuzzy features and Word2Vec had strong predictive powers for identifying duplicate question pairs. We discussed the use of neural models and its advantages over traditional feature-based classifiers. We implemented Siamese Manhattan LSTM, a deep neural model, that performed significantly better than the other models and achieved an accuracy of 81%.

Future effort is required in terms of exploring other methods that could contribute in improving the Siamese MaLSTM network. Experimenting with state-of-art models in paraphrase identification such as, ABCNN[12] model and BiMPM[13] model could help in achieving higher accuracy.

8. References

1."Quora Question Pairs | Kaggle". Kaggle.com. N.p., 2017. Web. 23 Apr. 2017.
2. Techniques for preprocessing- https://www.analyticsvidhya.com/blog/tag/text-preprocessing/.
3.Word cloud- https://www.datacamp.com/community/tutorials/wordcloud-python
4.Google News- https://code.google.com/archive/p/word2vec/
5. WMD- https://towardsdatascience.com/word-distance-between-word-embeddings-cc3e9cf1d632
6. Importance of distance measures - https://towardsdatascience.com/importance-of-distance-metrics-in-machine-learning-modelling-e51395ffe60d
7. Dhakal Ashwin,Poudel Arpan,Pandey Sagar, Gaire Sagar. Exploring Deep Learning in Semantic Question Matching.10.1109/CCCS.2018.8586832
8.GloVe Embeddings - https://nlp.stanford.edu/projects/glove/
9. "Understanding LSTM Networks" Colah's blog. Aug 27, 2015. Web. 24 Apr. 2017.

10. Pond Premtoon, anwers to Quora question "What are Siamese neural networks, what applications are they good for, and why?" quora.com. Jun 1, 2016. Web. 24 Apr. 2017.
11. Jonas Mueller,Aditya Thyagarajan.”Siamese Recurrent Architectures for Learning Sentence Similarity”.2016
12. Wenpeng Yin, Hinrich Schutze , Bing Xiang, Bowen Zhou , “ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs”.2016
13. Wang,Hamza,Florian,“Bilateral Multi-Perspective Matching for Natural Language Sentences”.2017