Duplicate Question Detection

Mohit Nihalani, Sushanth Samala, Rohit Saraf
{mn2643 , ss12852, rs6785}@nyu.edu

Abstract On Quora, multiple questions with the same intent can cause seekers to spend more time finding the best answer to their question and make writers feel they need to answer multiple versions of the same question. Quora values canonical questions because they provide a better experience to active seekers and writers, and offer more value to both of these groups in the long term. In this project, we want to tackle this natural language processing problem by applying advanced techniques to classify whether question pairs are duplicates or not. To summarize the problem we are trying to Identify which questions asked on Quora are duplicates of questions that have already been asked. In this project, we have experimented with various text preprocessing, feature engineering techniques like Fuzzy features and compared the performance of various algorithms Logistic regression, XGBoost, Siamese Network. We got the best results by using the Siamese adaptation of a Bidirectional GRU with Bert Embeddings.

Introduction

Quora is a platform to ask questions to get quality answers and valuable insights. Quora allows like minded people to connect and share ideas and to gain and share knowledge about anything. More than 100 million people are active on Quora every month, so it's natural that many questions tend to be similarly worded. Therefore, these multiple questions with the same core question cause seekers to spend a lot of time in finding the best answer to the question and make writers feel compelled to answer multiple versions of the same question. Quora likes these canonical questions because of the value it provides to both the groups and the better experience to answer seekers and writers.

Currently, Quora uses a model to identify these duplicate questions which is a Random Forest model which is an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees[1]. In our project, we applied advanced natural language processing techniques and models to check whether the questions are different versions of the same question or not.

Prior Work

Previous work related to similarity between sentences was carried out based on entailment and logical inference Rocktaschel et al. (2015)[2] focused on attention methods based on LSTM(Long Short-Term Memory). Entailment problem is different from our problem in the sense that we are trying to find out inherent symmetry between the questions. [3] Explores approaches based on LSTMs using a Siamese architecture with learned representations and another method of using two LSTMs with two sentences in the sequence.

[4] focuses on continuous bag of words neural network model which surprisingly out-performs much more complicated recurrent and attention based models. [5] proposes a new siamese model architecture utilizing two identical LSTM sub-networks and employs a 3 layer neural network architecture for classification. Siamese model has a drawback that there is no interaction between the questions during the training process, this issue may cause information loss, [6] proposed a compare-aggregate architecture to resolve this issue. Compare-aggregate makes use of word-level matching and aggregating the matching results for final classification.

In the recent days, more advanced models have been proposed such as attention based model by [7] which uses convolutional neural networks(CNNs) to capture the interdependence between two sentences and information at different levels of abstraction.

Methods

Preprocessing

The data consists of a massive collection of human generated text and contains many non-ASCII characters. We employed different preprocessing steps to eliminate these anomalies.

Starting by eliminating html tags, we removed punctuations, Not a number and stop-words. We used NLTK's stop-words to perform the removal. In addition to this we performed stemming which is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form[8]. In the end expanding contradictions to finish up the preprocessing steps.

Feature Extraction

Questions were preprocessed by removing non-ASCII characters and other techniques mentioned in the above section. Basic features like frequency of the words, common words, length of sentences etc were extracted prior to preprocessing and after the preprocessing step more advanced and fuzzy features were extracted such as fuzzy ratio (measurement of similarity between two sentences using

edit distance), fuzzy partial ratio, token sort ratio (sorting token in sentences and then scoring fuzzy ration), token set ratio, longest substring ratio where tokens are generated by directly splitting the sentence where-ever a space is found. We featurized text data with tfidf weighted word-vectors to obtain similarity between the question pair, the greater the tfidf values the greater the similarity.

Analysis Step

The extracted features are stored in a Pandas Dataframe format so that it is easy to manipulate it. We generated word clouds of duplicate and non-duplicate question pairs to see the most frequent ones. We then looked at the features of each category and plotted some of them like 'csc min', 'token sort ratio' to compare duplicate and non duplicate questions. We observed that the first words of the questions may not have to be considered when determining the similarity between them because most of the questions have starting words such as "What", "Who", "Where", "Why" the number of question pairs with the same starting words for the duplicated and non duplicated questions are 62% and 42% respectively. Hence, it does not make much of a difference if we ignore the starting words.

Word Embeddings

We used GloVe [11] model pre-trained word vectors as our initial word embeddings. GloVe is a context-free model which generates a single word embedding for each individual word. We experimented with 50, 100, 300 dimension vectors and found 300 dimensions to produce the best results. We used Spacy which has in-built GloVe embeddings dataset called "en_core_web_sm" with 300 dimension vectors, this is the most used word to vector embeddings with a lot of common words used in human text, this makes it ideal for our dataset of questions asked by humans.

We also used BERT [12]: Deep Bidirectional Transformers for Language Understanding [13] which is also a contextual model which generates a representation of each word that is based on the other words in the sentence thus capturing the relationship in a bi-directional way. These embeddings are useful for semantic search and information retrieval, thus making it ideal for our task of determining duplicate questions. These vectors are high quality feature inputs to downstream models. BERT has several advantages over GloVe in the sense that BERT can capture polysemy, this context-informed word embeddings capture more accurate feature representations, which resulted in better model performance.

Data

The dataset consists of question pairs and the target value duplicate or non-duplicate. The data consists of 404290 question pairs with appropriate

id - the id of a question pair.
qid1, qid2 - unique ids of each question.
question1, question2 - the full text of each question.
is_duplicate - the target variable. It is 1 if duplicate, if not duplicate it is 0.

The topics in the data span over a large range of topics including technology, culture, entertainment, politics and many more. The target values is_duplicate is given by human experts at Quora, and thus it is not 100% accurate in the true meaning of the questions, the dataset is noisy and we need to take that into account.

Models

Baseline : Random Model

As a baseline, we used Dirichlet distribution technique to predict if the pair of questions are duplicate or not. Dirichlet distribution is a family of continuous multivariate probability distributions parameterized by a vector of positive reals[9], it essentially generates a distribution of numbers which sum to 1, thus ideal for our baseline model.

Logistic Regression

Linear regression models are famous for classification tasks, the relationships between parameters are modeled by linear predictor functions whose parameters are modeled using the data in the training phase. Linear regression focuses on the conditional probability distribution of the responses rather than the joint probability distribution given the value of the predictors.

We have experimented with the stochastic gradient descent (SGD) learning variant of the linear regression, in which the gradient of loss is estimated for each data point and the model learns the best parameter values for the optimal results. As we have a binary classification problem(duplicate or non-duplicate), binary class l_2 penalized logistic regression minimizes the following cost function[10]:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1). \quad (1)$$

Note that, in this notation, w are the coefficients of the model, X is the design matrix consisting of approximate linear dependence values, C is the regularization parameter of the model, the value of y_i is the target value which has two possibilities -1(non-duplicate), 1(duplicate).

XGBoost

XGBoost stands for Extreme Gradient Boosting, this model is famous for it's speed and performance. In the previous model we only used a single model to train and learn the parameters form the data, here we make new models each time to correct the errors of the previous models i.e, an ensemble of weak prediction models. Models are added sequentially until no further progress/optimization can be made. XGBoost also uses gradient descent algorithm to optimize the parameters of the model.

The model employs the following algorithm for optimization[11]. Given the training set $\{(x_i, y_i)\}_{i=1}^{n}$,, a differentiable loss function $L(y, F(x))$,. the algorithm initializes the model with a constant value and computes the pseudo-residuals for each iteration using the formula

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n. \quad (2)$$

Then the model fits a base learner h_m(x) to pseudo-residuals i.e, trains it using the training set and computes the multiplier γ_m by solving the following one-dimensional optimization problem

$$\gamma_m = arg_\gamma min \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)) \quad (3)$$

And updates the model using $F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$. to obtain a better model and this goes on for each iteration until the model can no longer be optimized.

We used GloVe embeddings for the Logistic Regression model and XGBoost model. Let us now look at the BERT model.

Siamese Network

Recently neural network has gain lot of popularity as are being used as a feature extractor for various NLP task specially due to advancement in Recurrent Neural Network architecture such as LSTM and GRU which helps to preserve information for longer sequences of data. For this experiment we will be using a special architecture of neural network known as Siamese Network (sometimes called a twin neural network) that uses same weight while working in tandem on two different input vectors to compute comparable output vectors.

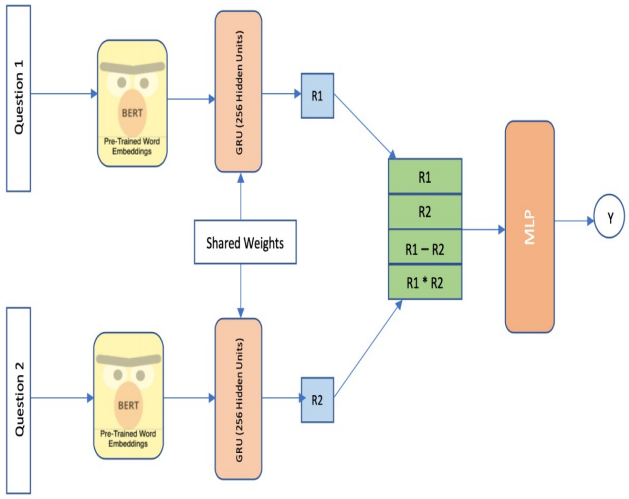

Figure 1: Siamese Network Model used for experiments.

Each sentence is first tokenized and mapped to its numeric index so each question is represented as word vectors that is padded to be 'c' word long. Then it is fed into model for learning which learns high level representations in the subsequent layers and makes use of those higher level features to perform the final classification (duplicate or not).

$$question = [a_1, a_2, ..., a_c] \in \mathbb{R}^c \quad (4)$$

As shown in figure 1 neural network architecture primarily consists of 4 layers:

Embedding Layer: We tried two different type of embedding first the Glove Embedding and for this we used [14] to easily load the embedding, convert words to index, and sentence's as vector of indexes and also create a Embedding matrix to be loaded in Neural Network. Secondly we use Bert Embeddings from huggingface repository(12) which provides pretrained Bert Model and api's to tokenize sentence and convert token to ids according to the input of the Bert pretrained model. We pool the outputs of Bert last hidden layer to create a representation which is then sent to a next layer. Size of the maximum sentence is set at 128 words, and the sentences which are smaller are padded with 0. So input to bert embedding layer, is the vector representation and attention mask for each question. Hidden unit size of Bert is 768 while for GLOVE is 300, so each word in the sentence is represented by high quality features depending upon the embedding used.

$$x_{embed} = EMBEDDING(comment) \in \mathbb{R}^{c*e} \quad (5)$$

Here e represents the dimension of word embeddings

Bidirectional GRU Layer: Emdedding representation for each sentence is then forwarded to bidirectional Gated Recurrent Unit layer[15] as they handle the vanishing and exploding gradient problem. We used the Bidirectional version as forward GRU captures unit from past and backward captures from the future. We implemented this using [16] and found one GRU layer with 512 hidden units was sufficient. We concatenated last hidden states of both forward and backward GRU to create a final feature vector for each question.

$$R_i = GRU(x_{embed}), i \in \{1, 2\} \quad (6)$$

Feed Forward Layer: Output of GRU layer for both the question are concatenated using a form proposed by [17].

$$R := (R_1, R_2, R_1 - R_2, R_1 * R_2) \quad (7)$$

$$logits = RW + b \in \mathbb{R}^2 \quad (8)$$

$$\hat{y} = Sigmoid(logits) \in \mathbb{R}^2 \quad (9)$$

Where R is the concatenated feature vector and R_1 and R_2 are the output feature vectors from the first question and the second question respectively. Finally the concatenated output

is forward to a single feed forward layer with 1024 hidden units and dropout of 0.2 and then finally to the output layer with Sigmoid activation function to obtain probability score $\hat{y}$ for the question pair being duplicate. Both the questions go through different network which share weights.
Usually For Siamese network contrasive loss function is used, but as we need probability of duplicate rather than similarity distance, we used binary cross loss function and Adam optimizer.

Experiments and Results

We ran experiments on the models discussed so far, we have used multiple initial parameter configurations of the models and displaying only the best results here.

Baseline: Dirichlet's distribution randomly produces the probability of the questions being duplicate. The baseline random model obtains a log loss value of 0.8888, all the models that we used should have a loss value lower than this to be an acceptable model.

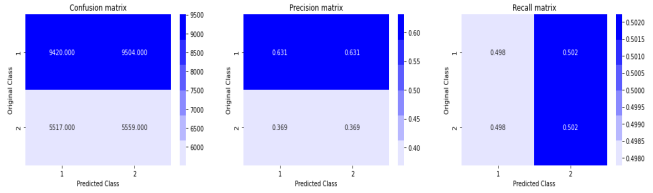

Fig 1. Random Model Confusion matrix, Precision matrix, Recall matrix.

Logistic Regression: The best learning rate for Logistic regression model is determined by initially picking a random alpha value and calculating the error value and then trying new values of alpha which reduces the error. We tuned the alpha value using 30,000 data points(as shown in Fig 2) and found the best value of alpha = 0.000862, we used this alpha value to perform training and testing.

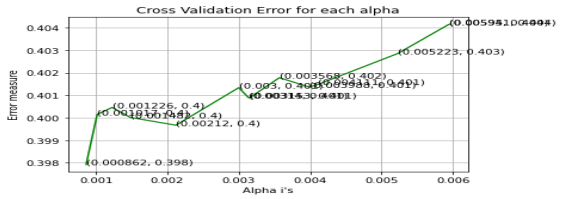

Fig 2. Logistic Regression hyper-parameter tuning.

In Fig 3 we can see the confusion matrix, precision matrix, recall matrix for the Logistic Regression model.

The log loss values for train and test set turned out to be, Train log loss = 0.39164 and Test log loss = 0.3979. The log loss values are considerably better when compared to the random model.

XGBoost: We ensembled the best performing model, this gave us a slightly better performance than the Logistic regression model. XGBoost model achieved a log loss value of 0.3482 on the test data. In Fig 4 we can see the confusion matrix, precision matrix, recall matrix values.

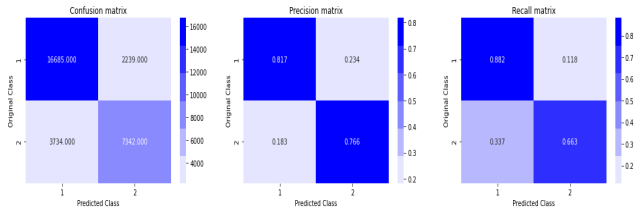

Fig 3. Logistic Regression Confusion matrix, Precision matrix, Recall matrix.

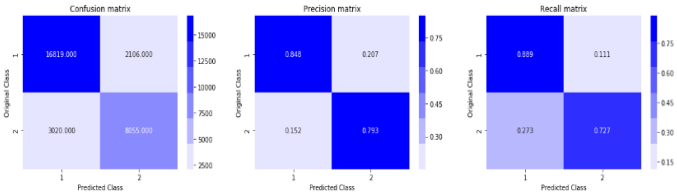

Fig 4. XGBoost Confusion matrix, Precision matrix, Recall matrix.

Siamese network: With the help of deep learning and word embedding techniques we were able to achieve promising result from both Glove and Bert embeddings. We experimented with different number of of GRU layers, and hidden units but there was no increase in performance, so out test results are based on 1 GRU layer with 512 hidden units. Siamese Network with BERT embeddings was able to outperform all other models achieving 91.7% accuracy and 0.277 log loss.

Table 1: Summary of main results on test data

Model	Precision	Recall	Log Loss	Accuracy
Random	49.77	63.06	0.8888	49.94
Logistic Regression	88.17	81.71	0.3979	80.09
XGBoost	88.87	84.78	0.3482	82.91
Glove Network	88.32	89.3	0.29568	86.4
Bert Network	88.6	86.4	0.27761	91.7

Conclusion

In this paper we demonstrated different approaches of identifying semantic similarity between questions with the help of various classical feature engineering techniques and also by learning high level features using deep learning. Our results show that with the help of semantic features such as Fuzzy features combined with machine learning models we can get classification accuracy. Our results also show that Bert embeddings are high quality contextual representation of words and when combined with Siamese Network can outperform classical models by a wide margin.

In future we would like to combine Siamese network with machine learning models as an Ensemble Learning model, where predictions of Siamese network will be used as one

of the features. We would also like to experiment with various document distance metrics such as Word Movers Distance, Minkowski distance, Canberra distance and use them as features. Perhaps main idea is to move away from siamese network towards the multi-perspective matching as described in [17] and it would be interesting to see how they perform in this task when combined with other deep learning models such as CNN.

References

[1] Wikipedia Random Forest Page; https://en.wikipedia.org/wiki/Random_forest

[2] Tim Rocktaschel, Tomas Kocisky, Phil Blunsom. Reasoning About Entailment With Neural Attention; https://arxiv.org/pdf/1509.06664.pdf

[3] Elkhan Dadashov, Sukolsak Sakshuwong, Katherine Yu. Quora Question Duplication

[4] Lakshay Sharma, Laura Graesser, Nikita Nangia, Utku Evci. Natural Language Understanding with the Quora Question Pairs Dataset; https://arxiv.org/pdf/1907.01041.pdf

[5] Zihan Chen, Hongbo Zhang, Xiaoji Zhang, Leqi Zhao. Quora Question Pairs; http://static.hongbozhang.me/doc/Quora.pdf

[6] Shuohang Wang, Jing Jiang. A Compare-Aggregate Model For Matching Text Sequences; https://arxiv.org/pdf/1611.01747.pdf

[7] WenpengYin, HinrichSchutze, BingXiang,BowenZhou. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs; https://arxiv.org/pdf/1512.05193.pdf

[8] Wikipedia Stemming page; https://en.wikipedia.org/wiki/Stemming

[9] Wikipedia Dirichlet distribution page; https://en.wikipedia.org/wiki/Dirichlet_distribution

[10] Scikit learn official logistic regression page; https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

[11] Jeffrey Pennington, Richard Socher, Christopher D. Manning (2014). Glove: Global vectors for word representation. In In EMNLP.

[12] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, Jamie Brew (2019). HuggingFace's Transformers: State-of-the-art Natural Language ProcessingArXiv, abs/1910.03771.

[13] Jacob Devlin and Ming-Wei Chang and Kenton Lee and Kristina Toutanova (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language UnderstandingCoRR, abs/1810.04805. [14] Radim Rehurek, Petr Sojka (2010). Software Framework for Topic Modelling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks (pp. 45–50). ELRA.

[15] Kyunghyun Cho and Bart van Merrienboer and Güland Fethi Bougares and Holger Schwenk and Yoshua Bengio (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine TranslationCoRR, abs/1406.1078.

[16] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A. (2017). Automatic differentiation in PyTorch

[17] Zhiguo Wang and Wael Hamza and Radu Florian (2017). Bilateral Multi-Perspective Matching for Natural Language SentencesCoRR, abs/1702.03814.