Question Pair Classification

Adarsh Puri apuri3@ncsu.edu
Emily Tracey etracey@ncsu.edu
Jonathan Nguyen jhnguye4@ncsu.edu
Koushik Shankar kshanka2@ncsu.edu

1 Background

Quora is a website with over 300 million monthly users, where they can ask or answer questions. It is a place where people can share or gain knowledge about anything they are curious about. With such a large community, similar questions are going to be asked which may cause users to spend more time to find their answers. Which leads us to our goal in "Quora Question Pairs Classification." Using a data set of sample Quora questions, we aim to present a model capable of determining whether a pair of questions are duplicates of each other. This can be extremely beneficial to users, allowing their questions to be answered much more quickly as well as reducing load on persons providing answers. This problem offers an opportunity to explore feature extraction, classification and deep learning techniques.

The ability to classify intent behind text has been used in varying contexts [3]. These consist of information extraction, detection of plagiarism, and question answering, which we will be focusing on in this paper. The ability to detect duplicates allows for simple question answering provided the duplicate detection is accurate. One example of previous work in this field is done by Zhang et al. (2018). They looked at duplicate detection within programming question communities. Comparison of candidate questions was accomplished with vector similarity metrics and association mining. Mueller et al. (2016) used word embedding vectors in combination with a modified LSTM to accurately model sentence similarity. Upon looking at trends within similar domains [3], the meshing of multiple feature extraction techniques with data mining models is common practice. Experimenting with sentence and word vectors with appropriate models appears to be paramount when structuring these problems.

Using outlines presented in previous work we separated our models to contain two distinct elements: feature extraction and classification. The full explanation of these elements can be found in Section 2. One limitation identified in similar work was the lack of direct comparison of techniques. Most papers identified a novel model to classify text, but it is difficult to directly compare these strategies presented because the scope and data used can differ widely. Our goal is to implement a larger collection of methods on a single dataset to allow for a direct comparison and analysis of the benefits of several feature extraction methods and classification models.

2 Method

A goal of our experiments is to determine appropriate strategies and models for discovering duplicate questions. In creating our models, we used standard data mining techniques consisting of preprocessing, feature extraction, and classification. Our goal is to combine different strategies to discover not only the best model, but also gain insight concerning the importance of preprocessing and benefits of different feature extraction techniques in relation to duplicate question detection.

2.1 Text Preprocessing

Preprocessing is an essential step when analysing data. We applied cleaning techniques including removal of excess punctuation, keeping case consistent and replacement of words with the same intent. We have also removed stop words when evaluating some models since some models seem to perform better with stop words. Attached below in Figure 1 is a sample of the first five questions of our dataset after text preprocessing has been applied.

```
what is the step by step guide to invest in share market in india
what is the step by step guide to invest in share market

what is the story of kohinoor kohinoor diamond
what would happen if the indian government stole the kohinoor kohinoor diamond back

how can i increase the speed of my internet connection while using a vpn
how can internet speed be increased by hacking through dns

why am i mentally very lonely how can i solve it
find the remainder when math2324math is divided by 2423

which one dissolve in water quikly sugar salt methane and carbon di oxide
which fish would survive in salt water

astrology i am a capricorn sun cap moon and cap risingwhat does that say about me
i am a triple capricorn sun moon and ascendant in capricorn what does this say about me
```

Figure 1:

2.2 Feature Extraction

Because strings of words can be difficult to inform models, we applied several algorithms to return vectors relating to each question. The algorithms we explored to perform our feature extraction are bag of words, Word2Vec, TF-IDF, and GloVe. Bag of words works by returning a vector relating to a grouping of text that contains the identification of the word used as well as frequency of each occurring word. Word2Vec on the other hand yields a vector consisting of the words within the given text and a numerical relation to other words in the space. TF-IDF analyzes the frequency of word to determine the significance of each word and assigns appropriate weights. GloVe focuses on the co-occurrences of the word over the whole document.

Word2Vec Features This algorithm uses a neural network model to learn word associations from a large corpus of text. Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence. As the name implies, word2vec represents each distinct word with a particular list of numbers called a vector. The vectors are chosen carefully such that a simple mathematical function (the cosine similarity between the vectors) indicates the level of semantic similarity between the words represented by those vectors. This feature extraction will be used on our models that require deep learning.

TF–IDF Term frequency–inverse document frequency, is a numerical statistic which reflects how important a word is to a document in the collection or corpus. This method is often used as a weighting factor in information retrieval and text mining [10]. This means that as a word appears more frequently in the document then the importance or weight of the word will also increase proportionally but will be counterbalanced by how frequent the word is over the whole corpus. The purpose of TF-IDF is to determine which words are more common than others. This feature extraction will be used on our models that do not require deep learning.

Bag of Words is a method to extract features from text documents and is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. It is the simplest way to represent text as numbers, as each sentence will represent a bag of words vector. Drawbacks of using Bag of Words is that as more sentences contain new words, this would lead to a larger vocabulary or vector. Which will then result in having many vectors that will be filled with mostly zeros. Bag of Words also holds no information which means that it will not know the context of the sentence. This feature extraction will be used on our models that do not require deep learning.

GloVe which represents Global Vectors, is an unsupervised learning scheme for obtaining vector representations for words. These are trained on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. In simpler terms, it focuses on words co-occurrences over the whole corpus. The embedding vector relate to the probabilities that two words appear together[9]. This can be viewed as the multiplication of the word features matrix and feature context matrix. This feature extraction will be used on our models that require deep learning.

2.3 Classification Models

Finding well preforming classification methods is vital to the success of determining duplicate intent. The ones we have chosen to analyze are various Bayesian classifications, XGBoost, Logistic Regression, long short-term memory (LSTM), and various other neural networks (including CNN). These were selected because of their promise in NLP tasks [2][4].

Bayesian classification addresses the classification problem by learning the distribution of instances given different class values[5]. It uses joint probability to find an estimate, uses this estimate on new instances, and classifies the instance on which is most probable. For our model we will use Naive Bayesian classification, where the attributes are conditionally independent from the given class.

XGBoost is a scalable machine learning system for tree boosting that has been widely recognized in a number of machine learning and data mining challenges. The system runs more than ten times faster than existing popular solutions on a single machine and scales to billions of examples in distributed or memory-limited settings. The system is able to run much more faster than other solutions as it has systems and optimization algorithms such as novel tree learning algorithm for sparse data, and procedures to handle instance weights to approximate tree learning[4].

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. One of the problems of Logistic regression is that it is prone to over-fitting if there are too many features hence we need to fine tune the regularization parameters so that the model is able to generalize.

Long Short-Term Memory (LSTM), is a recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behavior required in complex problem domains like machine translation, speech recognition, and more. LSTM's are widely used for such tasks because they are able to overcome the problems of vanishing gradients and exploding gradients. Figure 2 shows the basic LSTM structure.

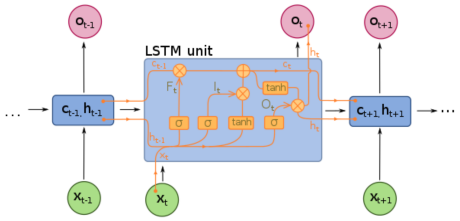

Figure 2: Basic LSTM Structure

Convolutional Neural Network (CNN), is a neural network that applies convolutional layers to local features. The inputs are words and each word can be represented by a vector, By appying different filters on the word vectors to create convolutional feature map and further proceed as per figure 3.

Time Distributed CNN (TD-CNN), is a a type of CNN that applies the same layer to several inputs to produce on output per input in time.

Evaluation For preliminary evaluations, we will look at interesting solutions we have discovered and analyze by dividing training and testing data. We aim to use our initial results to shape our final evaluations. While, the primary metric we will use to compare models is accuracy, other metrics including recall, precision and f-measure will not be ignored.

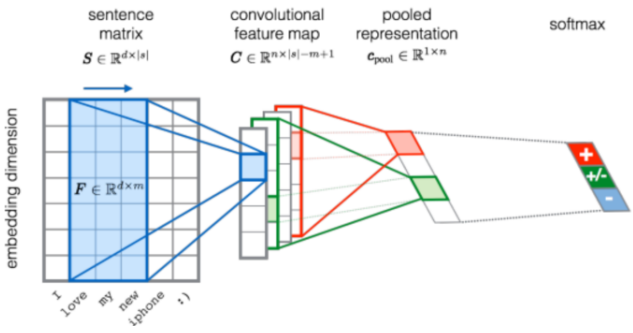

Figure 3: Basic CNN Structure

3 Plan and Experiment

In our experiment we hoped to gain insight not only into the best preforming model, but also into several hypotheses. These hypotheses are as follows: 1. Deep learning models will have better performance when comparing question intent, 2. Combinations of methods that leverage strengths of each leads to higher accuracy in determining duplicates, and 3. Impact of preprocessing and feature extraction will be significant to overall performance. To test these statements we proposed to split our models into two categories, simple machine learning models with no neural networks and complex deep learning models. Both types of models are tested against several feature extraction methods listed above to determine the impact they have on the accuracy of the classification model.

3.1 Dataset

Quora released its first ever dataset publicly on 24th Jan, 2017. This dataset consists of question pairs which are either duplicate or not. Duplicate questions mean both question have same meaning.

The dataset for this project will be taken from the open dataset provided by Kaggle, at "Quora Question Pairs Classification." The dataset consists of pairs of questions with following data labels id, qid1, qid2 (unique ids of each question), question1, question2, and isduplicate (set to 1 if question1 and question2 have essentially the same meaning, and 0 otherwise). The ground truth labels on this dataset is taken to be 'informed' but not 100% accurate and may include incorrect labeling. We have displayed a histogram in Figure 4 to visualize this dataset and to show that out of the 404290 training labels, there are 255027 non-duplicates and 149263 duplicates. The testing data, on the other hand, has 2345796 question pairs. We preprocessed the entire dataset as described in the method section above and split our data as 80% train, 10% validation and the rest 10% to test our models.

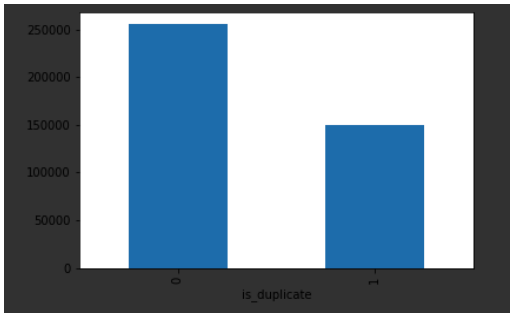

Figure 4:

3.2 Model Combinations

For our simple models, we tested various combinations involving feature extraction methods of bag of words, TFIDF, and n-grams with the classification models XGBoost, multinomial naive

Bayes and, logistic regression. The nature of these models limited feature extract methods to those yielding a single vector for the whole question, but we still maintained used several techniques on a single classifier to test the impact of feature extraction methods. Figure 5 shows the framework for the Machine Learning simple models. For our complex deep learning models, we created different

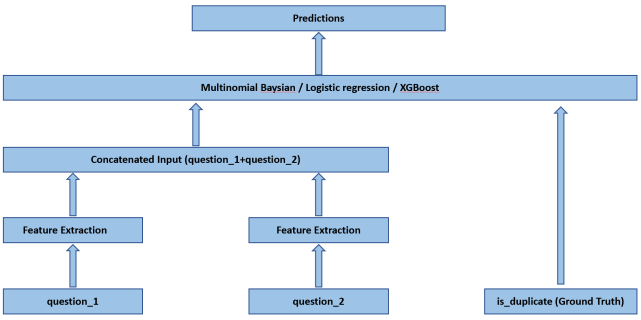

Figure 5: Machine Learning Models

combination of models that used feature extraction methods of Word2vec and GloVe embeddings with classification models of LSTM, TD-CNN, and 1D-CNN. Word2vec and GloVe embedding feature extraction methods were better suited for our complex models as they took into account context of the word to create vectors and matrices and again allow for the significance of feature extraction techniques to be illustrated. In this form of transfer learning, we trained our word embedding on google corpus(Word2vec) and wikipedia corpus[11] (GloVe), these are the pre-trained word vectors. We froze this embedding layer parameters and fed it to our Models. We used the Keras Library to bult the LSTM, TD-CNN, and 1D-CNN classifiers individually and in combinations. As the word embeddings were pre-learned we added additional fully connected layers to train the parameters while fitting the neural network. The layer takes 20000 as the first argument, which is the size of our vocabulary corpus, and 300 as the second input parameter, which is the dimension of the embedding. The third parameter is the input length of 40, which is the length of each comment sequence. In Figure 6, there is a visualization of the steps a model takes to predict if a question is similar or not. In addition to this, we trained a more complex merged deep learning model to test various combinations of classification techniques yielding a better performance. Figure 7 shows the framework for the merged model.

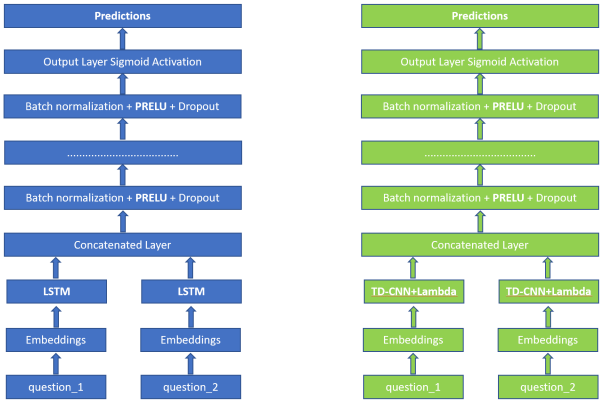

Figure 6: LSTM Model TD-CNN Model

4 Results

Our best preforming model among the simpler models was logistic regression with bag of words and n-grams yielding 0.8029 accuracy on the test data and 0.9681 on the training. The model with the second best performance was multinomial Bayes with bag of words and n-grams (0.7832 testing accuracy and 0.9079 training accuracy). Interestingly, our worst preforming combination was

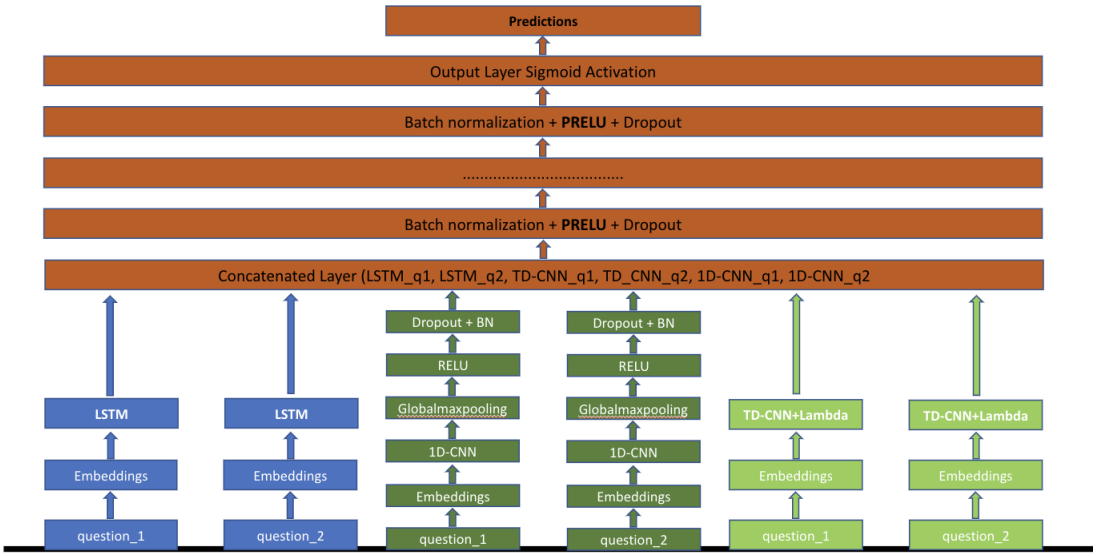

Figure 7: Merged Model

multinomial Bayes with bag of words without n-grams, suggesting significant value of including n-grams. Other combinations of extraction and classification methods were XGBoost with bag of words and logistic regression with TFIDF and n-grams. Results of all of our simple models can be found in Figure 8.

Model	Training Accuracy	Testing Accuracy	Training F1 Score	Testing F1 Score
Logistic Regression with Bag of Words and n-grams	0.9681	0.8029	0.9655	0.7778
Multinomial Naïve Bayes with Bag of Words and n-grams	0.9079	0.7832	0.8989	0.7574
XGBoost with Bag of Words	0.8467	0.7881	0.8279	0.7587
Logistic Regression with TFIDF and n-grams	0.8353	0.7823	0.8082	0.7418
Multinomial Naïve Bayes with Bag of Words	0.7666	0.7323	0.7456	0.7093

Figure 8: Accuracy and F1 scores from models described in the leftmost column sorted by Training Accuracy

After running our deep learning models, we came to conclusion that the complex model that used Word2vec with a merged layer of LSTM, TD-CNN, 1D-CNN resulted in the best testing accuracy of 0.83 and training accuracy of 0.98. The other models Word2vec with LSTM, Word2vec with TD-CNN and 1D-CNN, and GloVe with TD-CNN and 1D-CNN all have very similar testing accuracy's of 0.82. Surprisingly,the models that used GloVe embeddings and LSTM model classifications produced the worst results, which showed more care should be taken while handling LSTM. Figure 9 is a table that shows the Testing accuracy of all of our complex deep learning models.

Before starting this project, we used Quora's model LSTM with concatenation as a baseline for our results which was an 0.87 testing accuracy. After conclusion of our experiments, we were unable to create a model that would outperform their results. Through testing different combination of

Model	Test Accuracy
Word2vec Embeddings + LSTM	0.82
GloVe Embeddings + LSTM	0.63
Word2vec Embeddings + (TD-CNN +1D-CNN)	0.82
GloVe Embeddings + (TD-CNN +1D-CNN)	0.82
Word2vec Embeddings + (LSTM+TD-CNN +1D-CNN)	0.83
GloVe Embeddings + (LSTM+TD-CNN +1D-CNN)	0.69

Figure 9: Complex Deep Learning Models Results

models we did come up with a model that came fairly close, which was our Word2vec Embedding with LSTM, TD-CNN, and 1D-CNN that had a testing accuracy of 0.83. In direct relation to our hypotheses, we also inferred that combining layers to models generally would lead to better accuracy, however this increase in accuracy was limited and only marginally better than Word2Vec Embeddings with just LSTM. One surprise stemming from the comparison of our simpler models and the complex deep learning models is that simple models can preform reasonably well. While our highest accuracy model is fairly complex and takes a long time to train, bag of words with n-grams and logistic regression preformed almost on par with it. Another key point we can conclude from this experiment is the importance of appropriate feature extraction. LSTM works better on Word2vec Embeddings compared to GloVe, as the models that involved GloVe scored accuracy's that were significantly lower than all the other models. Another example of this appears from the with the addition of n-grams to simpler models. If data preprocessing and feature extractions are done properly and added to a simple model, it can have far more effective results. Overall we were unable to achieve the goal of finding a model that would have a accuracy greater than 0.87 but we were able to recognize which models were more effective than others.

5 Conclusion

In this paper, we have explored methods to classify question pairs as duplicates. Various frameworks and methods for comparison are presented and aim to be a guide for future research with classifying intent within text. The generalizability of the models presented remains unclear and interesting directions for future research could involve using multiple similar datasets with the models presented. Testing against multiple question pairs from various sources and topics would likely reveal any relationships between the models and ability to detect duplicates, since our results are specific to the Kaggle Quora dataset. Moreover, adding batch normalization improved accuracy. The use of the leaky Relu activation function for hidden layers, i.e. PRELU instead of Relu function did not change the accuracy. With the addition of any amount of dropout decreased accuracy. A total of 5 hidden layers in the fully-connected component had the best accuracy, with between zero and 7 hidden layers evaluated. From the number of tested dimension of 50 ,100, 150 and 300, using 300 dimensions for the layers in the fully-connected component showed the best accuracy. We also observed that the merged model with although the GloVe embedding has more trainable parameters in merged model but it performed the worst. Figure 10 shows the number of parameters in 4 of those deep learning models. Additionally, further exploration into the affect of preprocessing and feature extraction methods could involve trying more combinations of the algorithms listed above. With our limited resources and time of the project, we were unable to test all possibilities, but extending our simple models to work with Word2vec embedding might provide further insight into the difference in performance of deep learning models. Analyzing a full matrix of performance would create an image of the effectiveness of each model and future avenues for increasing performance.

MODEL	TRAINABLE PARAMETERS	Non-Trainable Parameters	Total parameters
Word2vec Embeddings + (TD-CNN+ 1D CNN)	1,182,717	114,723,000	115,905,717
GloVe Embeddings +TD-CNN	1,178,817	114,720,600	115,899,417
Word2vec Embeddings + (LSTM+TD-CNN+CNN)	2,806,317	172,082,400	174,888,717
GloVe Embeddings + (LSTM+TD-CNN+CNN)	60,159,417	114,720,600	174,880,017

Figure 10: Number of Parameters in Deep learning Models

References

[1] Wei Emma Zhang, Quan Z. Sheng, Jey Han Lau, Ermyas Abebe, and Wenjie Ruan. 2018. Duplicate Detection in Programming Question Answering Communities. ACM Trans. Internet Technol. 18, 3, Article 37 (May 2017), 21 pages.

[2] Samuel R Bowman & Gabor Angeli & Christopher Potts (2015) A large annotated corpus of natural language inference

[3] Mamdouh Farouk (2019) Measuring Sentences Similarity: A Survey

[4] Tianqi Chen, Carlos Guestrin (2016) XGBoost: A Scalable Tree Boosting System

[5] Nir Friedman & Ron Kohavi(1999) Bayesian Classification

[6] Stephan Dreiseitl & Lucila Ohno-Machado(2003) Logistic regression and artificial neural network classification models: a methodology review

[7] Ralf C. Staudemeyer & Eric Rothstein Morris(2019) Understanding LSTM a tutorial into Long-Short-Term Memory Recurrent Neural Network

[8] Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16). AAAI Press, 2786–2792.

[9] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation," in Proceedings of the 2014 Conference on Empirical Methods In Natural Language Processing (EMNLP 2014), October 2014.

[10] Hans Christian, Mikhael Pramodana Agus, Derwin Suhartono. "Single Document Automatic Text Summarization Using Term Frequency-Inverse Document Frequency (TF-IDf)", December 2016 [11] https://nlp.stanford.edu/projects/glove/

Appendix

GitHub `https://github.ncsu.edu/engr-csc522-fall2020/P16`