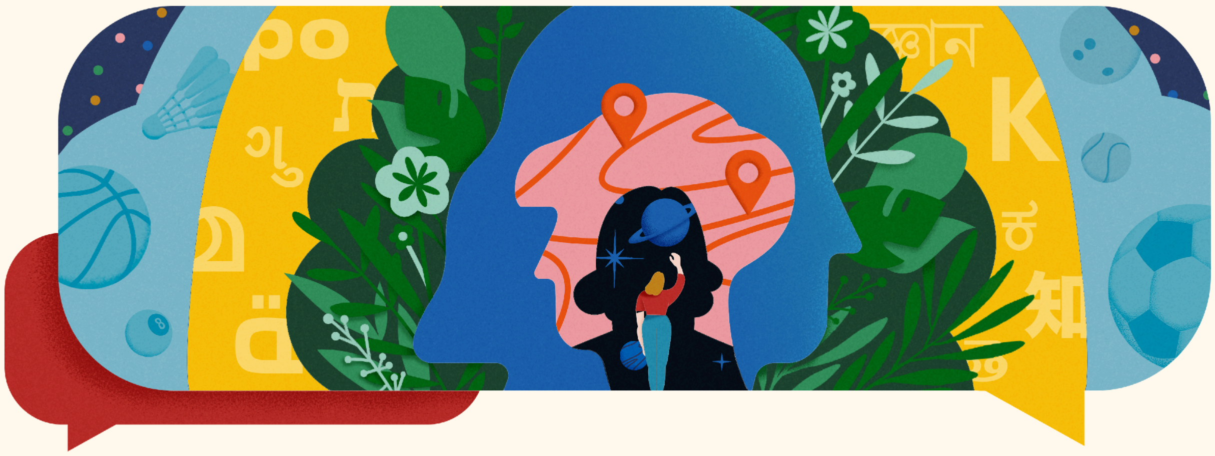
INTRODUCTION TO NATURAL LANGUAGE PROCESSING – SPRING 2024
PROF. AFRA MASHHADI

Quora Question Pairs
Predictive Modeling

By William Kluck & Mo Young



INTRODUCTION

Quora is a website where users can ask questions and receive answers on various topics. Being able to find answers quickly is essentially Quora's whole purpose. It would be helpful to know if the question someone is asking has been asked before, in one way or another. Our project sets out to solve this issue by training a model to identify when questions harbor the same intent.

Dataset Selection

There are many ways we could have approached this topic. The initial idea originates from Kaggle[1]. With a large dataset of roughly 400,000 question pairs, there is plenty of data to be confident that a model could be trained well without overfitting. The data is organized in a csv-file with the following columns: id, qid1, qid2, question1, question2, and is_duplicate. The

[1] DataCanary, hilfialkaff, Lili Jiang, Meg Risdal, Nikhil Dandekar, tomtung. (2017). Quora Question Pairs. Kaggle. https://kaggle.com/competitions/quora-question-pairs

column id represents the unique pairing of the 2 questions. Next, "qid1", or Question id 1, is the individual identifier for "question1", a single question from Quora.

Similarly, "qid2", or Question id 2, is the personal identifier for "question2", another question from Quora. The column "is_duplicate" is a boolean value representing whether the pair of questions had the same semantic meaning or intent. A value of 0 means that the questions are not matching, to be called non-duplicates. A value of 1 means that the questions are matching, to be called duplicates.

Preprocessing

Machine learning models depend on the quality of the data being fed into them. Depending on the model this may require some data cleaning, or augmentation

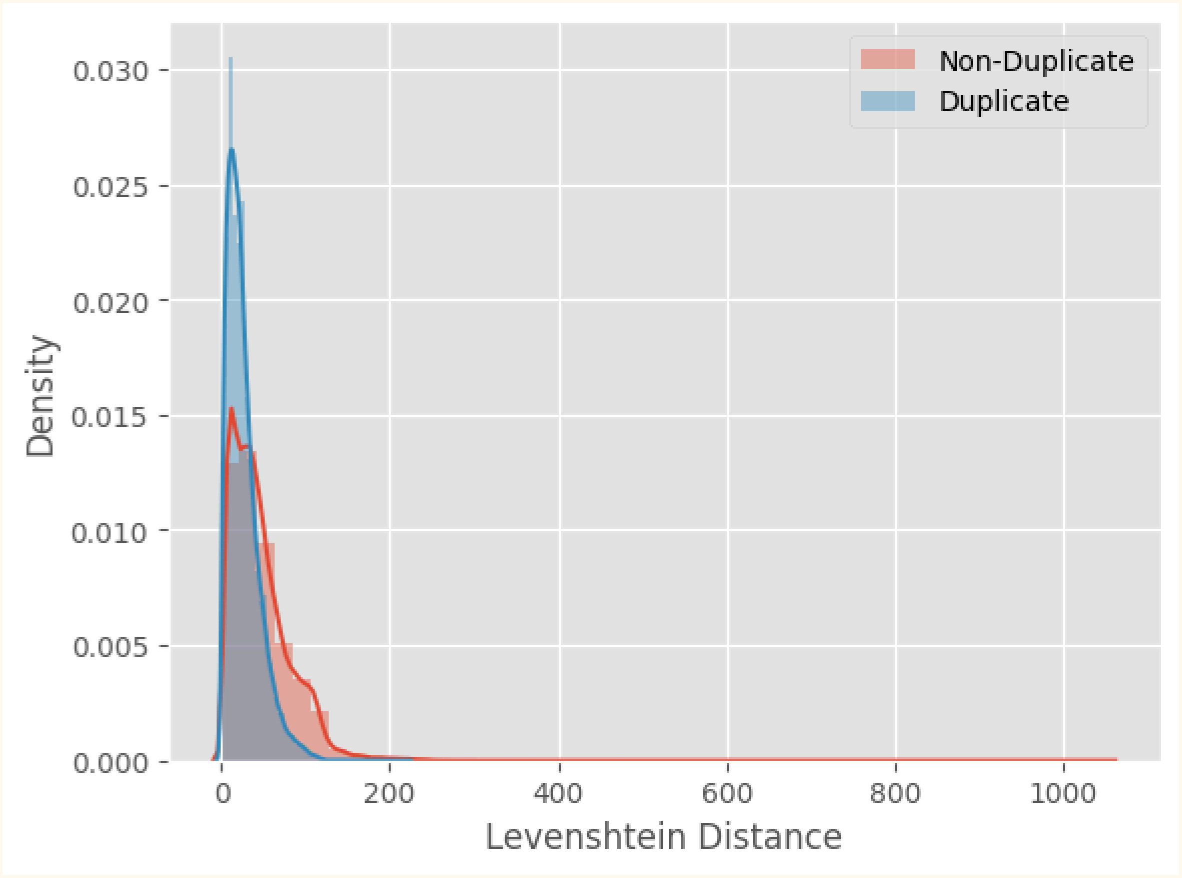
1. Removing Null Values

Not having null values in our data set was crucial. If a question is missing, then we will not know if it is similar to the other question. In our code, we only had 3 null values, so the best course of action was to remove all rows that had null values.

2. Adding/Removing Columns

Depending on the model, having unnecessary data can be a distraction. We wanted to ensure that we had only the data we needed when training our models. Removal of id, qid1, and qid2 are methods that we use. Another way that we attempt to add useful information to our dataset is to take the Jaccard similarity value and the Levenshtein distance value. The Jaccard similarity uses cosine similarity to show how far away vectors are from each other. The Levenshtein distance method uses swapping, deletions, and additions to count the steps to turn one sentence into another. The Jaccard Similarity method on our dataset yields data that is not very useful, while the Levenshtein distance metric yields tangible results (See Image 1).

Image 1: Levenshtein Distance for Duplicates and Non-Duplicates



The Levenshtein Distance for non-duplicates is higher on average when compared to duplicates. This means that non-duplicates have larger sentence structural differences than duplicates. Although, the difference is very low. In addition, we can see the mode values for both duplicates and non-duplicates are around 25. This shows that the Levenshtein distance while an interesting representation of the data may not be the most helpful in building our model. Because of this, we did not consider it when training our models.

Feature Extraction

In machine learning, feature extraction is a critical step in turning language into vectors and numerical representations. This allows models to perform calculations, and transformations to come to a definitive conclusion.

Word2Vec

Word2Vec turns each word in a sentence into an individual vector inside a vector space. We utilized the bag-of-words architecture which encodes each word and utilizes the word vectors to predict a target.

TF-IDF

TF-IDF stands for “Term-Frequency-Inverse-Document-Frequency”. Similar to Huffman encoding, this extraction method turns each sentence into a vector based on the frequency of each term in the overall dataset. It then uses the smallest encoding for the most frequent terms and the largest for the most infrequent ones. This saves critical space when utilizing the numerical vectors. Our Logistic Regression model uses this encoding (See Model Selection).

Model Selection

Logistic Regression

Logistic regression uses probabilities to predict binary outcomes. As the results of our data were in the form of 0 (non-duplicates) and 1 (duplicates), this model seemed to perfectly align with our project’s goal.

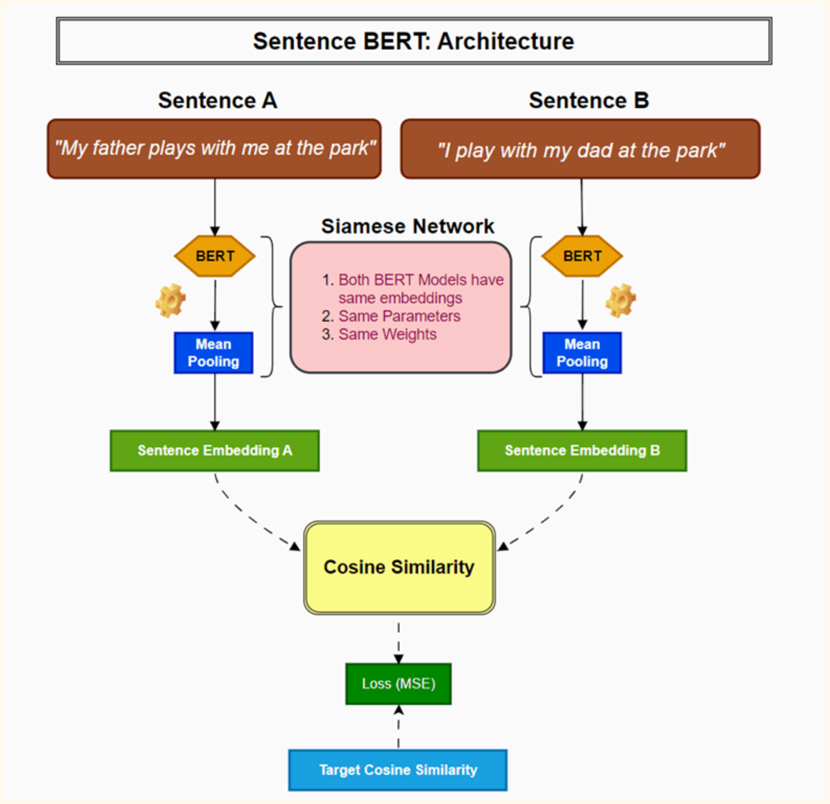
Naive Bayes

The Naive Bayes Classifier algorithm stems from the principles of the Bayes theorem and the “naive” assumption that every feature is conditionally independent given the target class. Given that Naive Bayes is simple to implement and not time-consuming to run (SVM took hours to run), we decided to give Naive Bayes a try in our project.

Siamese BERT

BERT stands for “Bidirectional Encoder Representations from Transformers” and is a great method for deciphering semantic meaning from text. Siamese BERT also known as Sentence BERT (SBERT) improves on this by comparing a pair of items by conducting separate BERT analyses’, this is called Bi-Encoding. After conducting the BERT, it tailors the results to contain an equal number of parameters and weights. This allows the predictions to be more accurate in the later steps. Then the model converts the semantic results into vector embeddings (See Image 2).

Image 2: Sentence BERT Methodology



The vector embeddings are compared by cosine similarity and output a loss value based on the Binary Cross Entropy Loss Function. We chose this loss function because it is good at choosing between 2 results (in our case duplicate or non-duplicate).

Evaluation Metrics

Accuracy

For each model, we produced an accuracy score based on how many correct predictions our model picked divided by the total number of predictions it made. The reason this metric is considered incomplete is that it ignores class imbalance. For example, our Quora question data is 60% non-duplicate. If our model predicted everything as non-duplicate, its accuracy would be 60% but it wouldn't be very useful at predicting duplicates.

Precision

Another metric we used is precision. Precision takes our model's total ratio of true positives and divides the total number of positives predicted. This serves to check how many duplicates predictions are duplicates. The reason precision by itself is incomplete is that it biases the positive class.

Recall

Similar to precision, recall is used to calculate a metric in terms of the positive class. Recall is the ratio of correct positive predictions to the total number of positive values in the data. The reason that recall is incomplete by itself is that it biases the positive class.

F1 Score

F1 is a combination of precision and recall. This is good in our case because it considers false negatives and positives. F1 does this by computing an average of precision and recall.

ROC-AUC

ROC stands for "Receiver-Operating-Characteristic" and is a type of graph showing the performance of a class. The curve compares true positives and false positives. AUC stands for "area under the curve". The AUC measures the quality of the model's predictions and how well the predictions are ranked. When the other metrics are high, it is important to look at ROC-AUC to see if the model performs well in all other cases.

Experimental Setup

Stratified K-Fold

We used stratified k-fold cross-validation with a fold of 10 on every classic ML model to make the environment as consistent as possible and to ensure that all the ML model's results are reliable. Just a side note, we were unable to figure out a way to implement Stratified K-Fold on the S-Bert model.

Results Analysis

Naive Bayes with Word2Vec

The results for Naive Bayes with Word2Vec showed an accuracy of 0.69, a precision of 0.59, a recall of 0.56, and an F1-score of 0.57. This model performed the worst in nearly all categories compared to the other models. When compared to the Logistic Regression with Word2Vec, both models had an equal F1 score. However, Logistic Regression outperformed in terms of accuracy and precision, while Naive Bayes demonstrated a superior recall.

Logistic Regression with Word2Vec

The results for Logistic Regression with Word2Vec yielded an accuracy of 0.72, a precision of 0.67, a recall of 0.49, and an F1-score of 0.57. In comparison, the Logistic Regression model using TF-IDF performed significantly better across all metrics. Notably, the recall and F1-score for the Naive Bayes model hovered around 50%, indicating poor performance. We attribute this to Word2Vec's reliance on contextual information from surrounding words, which is challenging to capture given the average question length of around 60 characters. This limited context likely hindered Word2Vec's ability to accurately grasp the semantic meaning.

Logistic Regression with TF-IDF

The results for Logistic Regression with Word2Vec demonstrated an accuracy of 0.76, a precision of 0.75, a recall of 0.76, and an F1-score of 0.75. Compared to other classic machine learning models, this model performed notably well. With all scores hovering around 75%, it exhibited no significant weaknesses, delivering consistently average performance across all metrics.

Siamese BERT with Word2Vec

The predictions for SBERT achieved an accuracy of 0.87, a precision of 0.84, a recall of 0.81, an F1 score of 0.82, and an impressive ROC-AUC of .94. This model outperformed all the other models we tested, exceeding our expectations. Given the complexity of our dataset, achieving accuracy in the high 80s is notably strong. Furthermore, it's worth highlighting that all the performance metrics remained above 80%, underscoring the model's consistent and robust performance.

Discussion

The implications of our results shed light on the strengths and limitations of various NLP models for our specific project.

Transformer Models

The Siamese BERT (SBERT) model significantly outperformed traditional machine learning models. This shows the power of transformer models in understanding the semantic meaning of text.

TF-IDF over Word2Vec

The Logistic Regression model using the TF-IDF demonstrated better performance across all metrics compared to the same model using Word2Vec. This suggests that TF-IDF, which emphasizes the importance of rare words, is more effective at distinguishing between duplicates and non-duplicates in our context.

Class Imbalance

The inherent imbalance in our dataset 60% non-duplicates and 40% duplicates had more of a hindrance on models that are sensitive to class distribution. The relatively balanced performance of TF-IDF shows that it has relative robustness to class imbalance. The SBERT's performance shows that it is better at handling imbalanced data.

Conclusion

Key Findings

In conclusion, this project highlighted the significant power and importance of transformers in NLP. The BERT model, in particular, demonstrated exceptional performance, consistently surpassing the results of all the classical machine learning models we employed. This underscores the transformative potential of advanced transformer models in achieving superior outcomes in NLP tasks. Notably, BERT excelled even with extremely similar question pairs that were not considered duplicates, exceeding our expectations. With further fine-tuning, the performance of the BERT model can be enhanced even more, promising even better results in future applications.

Reflections

To improve our results, we could have made several adjustments. Firstly, while using 5 epochs for SBERT yielded strong performance, additional epochs might have further enhanced the model's effectiveness. Additionally, incorporating more cross-validation for SBERT would have provided a more robust evaluation. Our dataset was approximately 60/40 non-duplicate/duplicate; balancing this dataset before training could have potentially led to better outcomes.

In terms of challenges, we faced limitations with Google Colab's free computational units. This restriction highlighted the need for a more powerful computer or the use of an online server to handle similar work in the future, ensuring smoother and more efficient processing.