Machine Learning Group Project

Text Similarity and Duplicate Detection

By Group-8

Mohtashim Butt (24100238)
Ushna Saeed (24100098)
Bisma Nawaz (24100277)
Shahzaib Ali (24100066)
Hadia Shehzad (25020331)
Waleed Nadeem (24100282)

Problem setup

The project aims to detect the similarity and duplication among different text prompts. It also aims to detect the context of the prompts and checks how similar they are.

Methodology

The training dataset we're using is from quora question pairs that can be found here. We are training a model on the sentence embedding for each prompt in our dataset and then we're evaluating our models' performances with our testing data. We are handling this problem following three different approaches:

1. Naive Bayes.
2. Pretrained Encoder-Transformers (BERT).
3. Large Language Sequence Model.

Pre-Processing

Since we're calculating the text similarity, we refrained from removing the stop-words. In that way, we won't be losing any context of the sentences. We did follow the following steps for pre-processing:

- Special character removal
- Numeric digits removal
- URLs removal
- Conversion to the lower case.

This ensures that the model receives a standardized and clean input, contributing to improved generalization.

Naive Bayes

Approach

We used the Multinomial Naive Bayes model to predict whether a pair of questions is duplicate or not, based on the probabilities of how frequently they occur. The TF-IDF vectorisation was used to create vectors, and it worked by capturing the importance of words in the question relative to the entire dataset. This model is quite effective for such natural language processing tasks in that it efficiently handles sparse data generated by TF-IDF vectorization and requires fewer parameters to estimate.

Training

After preprocessing the data we trained the model on 90,000 questions from the dataset. The model estimated the probabilities during the training stage by counting the occurrences of each term in each class. It then calculated the probability of a question being a duplicate or not based on the observed word frequencies.

Results

For the predictions, it used the test data (30,000 questions) to assign a predicted class label based on the level of similarity. Below are the evaluation metrics for the model's performance:

Matrix	Numbers
Accuracy	0.7257
Precision	0.76
Recall	0.39
F1	0.52

An accuracy of 0.726 is reasonable, but it might even be misleading as there was a substantial class imbalance in the dataset (with 75,351 non-duplicates and 44,649 duplicates). Even if it predicted a lot of false negatives, it would still achieve a high accuracy because of the class imbalance.

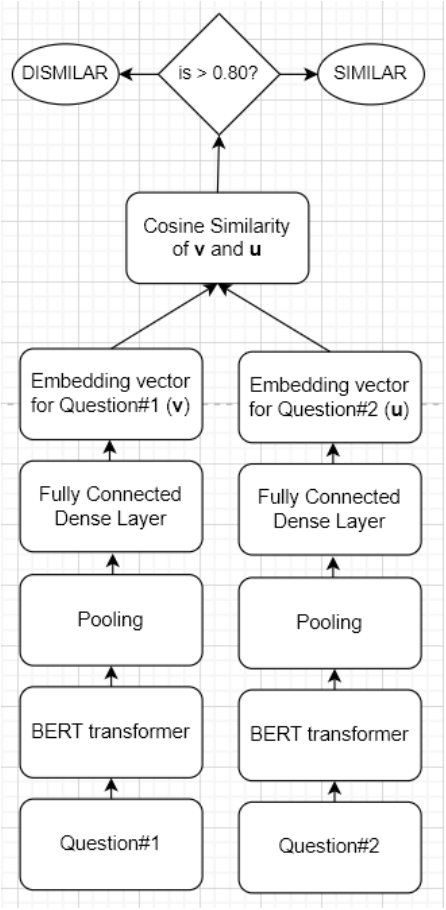
A precision of 0.76 looks promising, but the recall and F1 score are quite low. Given the class imbalance, the F1 score gives an accurate evaluation of the model's overall performance. The F1 score of 0.52 suggests that the model was not really able to capture the complexities in the data instances that would help it make accurate predictions.

Discussion
Some possible reasons for the subpar performance could be imbalanced dataset, feature representation, or model complexity. The dataset had a higher number of non-duplicates (75,351) than duplicates (44,649), which was causing a high accuracy (0.726) but low recall (0.39). A better distribution of positives and negatives could have resulted in better performance. The usage of TF-IDF vectorization instead of more sophisticated or pre-trained embeddings might also be causing the model to not capture the semantic meanings of the questions which would help in better predictions.

Our model is also very simplistic, and is not engineered to capture the intricate relationships within the data. It makes simplifying assumptions like feature independence (all words in the question sentences being conditionally independent of the class they belong to), which might not be true in real-world data. This assumption makes the computation quick and easy, but potentially compromises on capturing the dependencies between the words. This encouraged us to try other, more complex, models for text classification, like Pretrained Encoder-Transformers (BERT), and Large Language Models.

Pretrained Encoder Transformers

Approach
The notebook for this approach can be found here. We used Sentence-BERT (SBERT) model – a modification of the pretrained BERT network that uses siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity – to calculate our text embeddings. Our pipeline for pre-trained model was as follows:



We then calculated the confusion matrix after comparing the results from the predicted labels with the actual labels given in the data. That confusion matrix was used to calculate the evaluation metrics mentioned below.

Training

Since the model we used is pre-trained, it did not require any training beforehand. At first, we used the "`distiluse-base-multilingual-cased-v1`" version of SBERT. The evaluation metrics for this model on the Quora Question Pair dataset were these (based on a batch of 120000 question-pairs):

Matrix	Numbers
Accuracy	0.7397111111111111
Precision	0.635419469407523
Recall	0.7051236116087424
F1	0.6684593393529394

Discussion

The low readings above might be because of using the pre-trained model that might've not been trained over the dataset we used. Hence to improve on the results, we trained a logistic regression model via the embeddings generated by the SBERT and then re-evaluated. Also,

if we decrease the threshold for the cosine similarity, the accuracy and precision decreases along but the recall and F1 increases. This means that with the threshold value increase, the number of false positives increases (i.e., the number of similar marks increases despite being dissimilar). This trend is followed till the 0.90 threshold. After that, it showed abnormal behavior.

Sequential Models

Approach:
The chosen methodology involves the utilization of Sequence Models, with a specific focus on Recurrent Neural Networks (RNNs) augmented with embeddings. We utilized the GPT-4All library to generate embeddings, providing a contextualized representation of input sentences that manages to capture semantic meaning and context. (The notebook can be found here)

Model Architecture:
The selected model architecture is based around the use of Long Short-Term Memory (LSTM) networks. LSTMs are employed due to their ability to capture long-range dependencies in sequential data. The model structure comprises an LSTM layer followed by a fully connected layer designed for binary classification.

Training Methodology:
The training process involves the use of binary cross-entropy loss and the Adam optimizer. The training loop is executed over 100 epochs, allowing the model to learn the intricacies of the training data. The model's performance is continuously monitored through the tracking of both loss and accuracy metrics.

Findings:
The training accuracy exhibits steady improvement over epochs, reaching an impressive value of approximately 99.90% after 100 epochs. The model undergoes thorough assessment on a separate validation set, yielding an accuracy of 81.04%.

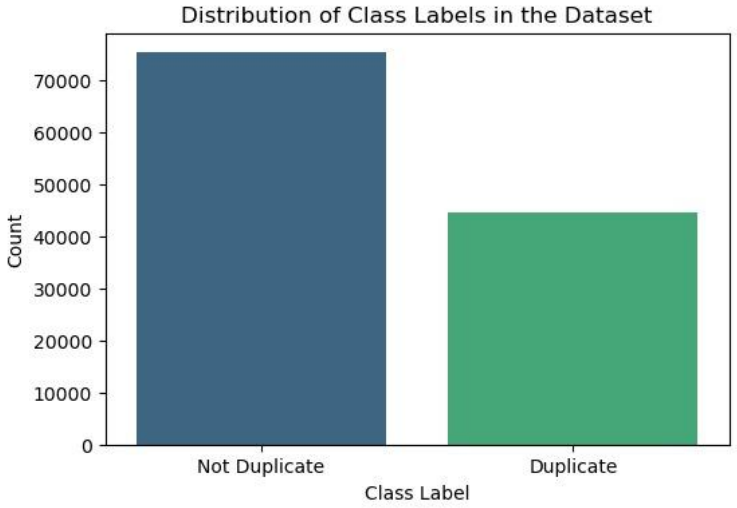
Matrix	Numbers
Accuracy	0.8104
Precision	0.7408
Recall	0.7610
F1	0.7508

Discussion:
The high training accuracy suggests that the model has learned the training data well. However, the lower validation accuracy indicates a potential overfitting issue or a need for further model tuning. Experimentation with hyperparameters and model architecture may be beneficial to improve generalization to unseen data.

Evaluation

The reasons behind evaluation scores being low can be because of:

- **Bias**: The dataset was marked manually by human annotators. Hence, it does contain the bias in it. The question pairs being marked similar/non-similar are actually in the human annotators' perspective which might not be reflected in some model's embeddings' perspective.
- **Class Imbalance:** The dataset had a higher number of non-duplicates (75,351) than duplicates (44,649), which might be causing the models to get biased towards non-duplicates. We can also see from the evaluation metrics that all the three models had high accuracy but a lower recall, which could be because of the class imbalance.



- **False Positive:** We believe that each of our models detected a relatively large amount of False Positives because of which, precisions and recalls were relatively low (among all three models).

Best Model: The best model out of all these was LSTM since it has the ability of capturing long-range dependencies in sequential data. The LSTM model also has a memory that stores information from previous steps. It then uses it to influence the output of the cell at the current time step. The output of each LSTM cell is then passed to the next cell in the network, allowing the LSTM to process and analyze sequential data over multiple time steps.So, this is why using the LSTM produced very high accuracy.

	Naive Bayes	SBERT	LSTM
Accuracy	0.7257	0.74	0.8104
Precision	0.76	0.635	0.7408
Recall	0.39	0.705	0.7610
F1	0.52	0.67	0.7508

As we can see from the comparison above, LSTM performed significantly well even with the huge class imbalance in the dataset.