**Quora Question Pair Similarity Detection using NLP and Machine Learning**

Sai Srinivas Lakkoju Vinay Vasetti

Data Science Data Science

[Saisrinivas.lakkoju@gwu.edu](mailto:Saisrinivas.lakkoju@gwu.edu) vinay.vasetti@gwmail.gwu.edu

**Abstract**

**Duplicate questions on platforms like Quora present challenges by cluttering search results and dispersing high-quality answers across multiple threads. This project focuses on identifying such duplicate questions using a combination of Natural Language Processing (NLP) techniques and machine learning models. By leveraging the Quora Question Pairs dataset, we preprocess, analyze, and extract meaningful features to quantify similarities between question pairs. Models such as Logistic Regression, XGBoost, and a Siamese BERT network are implemented to classify question pairs as duplicates or not. The project achieves scalable and robust performance by integrating advanced embedding techniques, fuzzy matching, and semantic similarity measures, ultimately improving user experience and reducing redundancy in knowledge sharing.**

1. **Introduction**

Duplicate question detection on Quora is a critical task aimed at identifying whether two questions convey the same meaning. This problem arises due to the platform's vast and growing repository of user-generated content, which often includes multiple ways of asking the same question. Duplicate questions can hinder users from accessing high-quality answers and increase the burden on contributors who may need to respond to similar queries repeatedly. Addressing this issue enhances user experience by improving answer discoverability and reducing redundancy in responses.

The task of duplicate question detection is framed as a binary classification problem, where the goal is to determine whether a given pair of questions is semantically equivalent. This requires effective representation of textual data as numerical inputs for machine learning models. Traditional approaches rely on hand-engineered features combined with tree-based algorithms like Random Forests or Gradient Boosted Trees. However, with advancements in deep learning, neural network-based methods have gained prominence, leveraging techniques such as sentence embeddings and transformer architectures like BERT to capture semantic nuances.

In this paper, we explore various machine learning and deep learning approaches for solving this problem using the Quora Question Pairs dataset. Starting with simple linear baselines, we experiment Gradient Boosted Trees, and advanced neural networks. Our study also draws inspiration from related work in Natural Language Inference (NLI), where models are trained to identify relationships like entailment or contradiction between sentence pairs. By leveraging these techniques, we aim to develop robust models that can effectively identify duplicate questions and contribute to enhancing Quora's knowledge-sharing ecosystem.

# Implementation

For this project, a variety of programming tools and libraries were utilized to implement and evaluate the duplicate question detection system effectively. These tools include:

* **Python**: The primary programming language used for data preprocessing, feature engineering, and building machine learning models.
* **NLTK (Natural Language Toolkit)**: Employed for natural language processing tasks such as tokenization, stopword removal, and lemmatization.
* **spaCy**: Used for generating word embeddings and extracting linguistic features to capture semantic relationships between question pairs.
* **FuzzyWuzzy**: Applied for fuzzy string matching to calculate similarity metrics like token set ratio and partial ratio.
* **TF-IDF Vectorizer**: Utilized for text vectorization to convert textual data into numerical representations based on term frequency-inverse document frequency.
* **XGBoost**: Implemented as a machine learning model to enhance prediction accuracy with optimized hyperparameters.
* **Logistic Regression**: Used as a baseline model for binary classification tasks.
* **Transformers (BERT)**: Integrated to develop a Siamese BERT model for advanced semantic similarity analysis between question pairs.
* **AWS EC2**: Leveraged for scalable computing resources, enabling efficient data processing and model training on large datasets.
* **Matplotlib and Seaborn**: Used for visualizing data distributions, feature correlations, and model performance metrics.

These tools collectively facilitated the development of a robust system for identifying duplicate questions in the Quora Question Pairs dataset.

# Data

## Data Exploration

The Quora Question Pairs dataset contains a training set of 404,290 question pairs and a test set of **2,345,795** question pairs, originally provided as part of a Kaggle competition. However, since the test set lacks labels for the question pairs, performance evaluation can only be conducted using accuracy through Kaggle's online submission system. To facilitate a more comprehensive analysis, including metrics beyond accuracy and detailed error analysis, we created our own test set by splitting the provided training set.

**Dataset Overview**

Each sample in the dataset consists of the following fields:

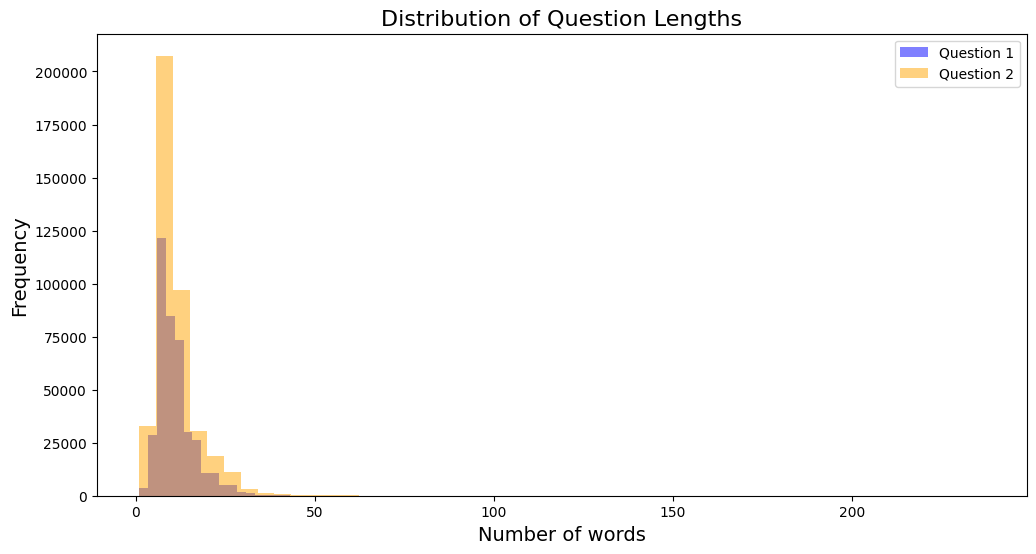
**id**: Unique identifier for each question pair, **qid1**: Identifier for the first question., **qid2**: Identifier for the second question, **question1**: Text of the first question,**question2**: Text of the second question., **is\_duplicate**: Binary label indicating whether the questions are duplicates (1 for duplicate, 0 otherwise).

**Key Observations**

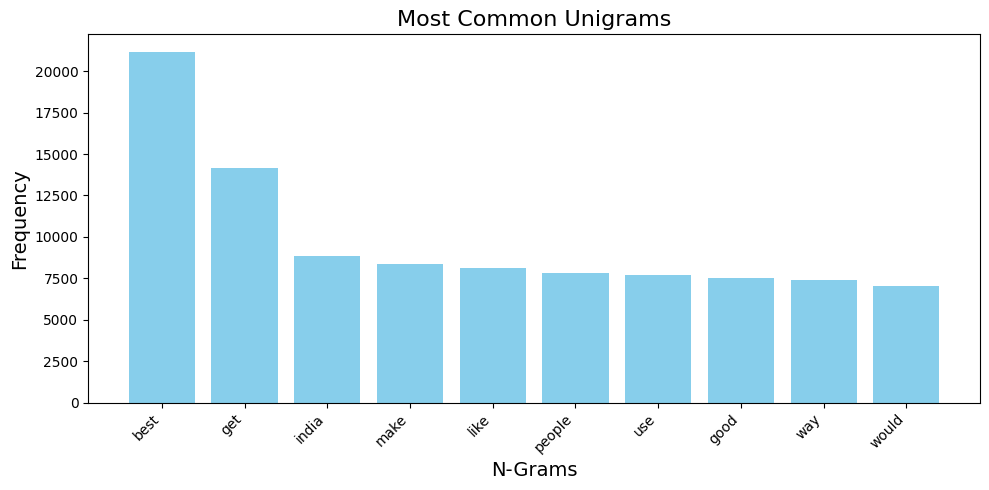
1. **Class Imbalance**:
   * Out of 404,290 question pairs:
     + 255,027 pairs (63.08%) are labeled as non-duplicates (0).
     + 149,263 pairs (36.92%) are labeled as duplicates (1).
   * This imbalance poses challenges for model training and evaluation.
2. **Question Uniqueness**:

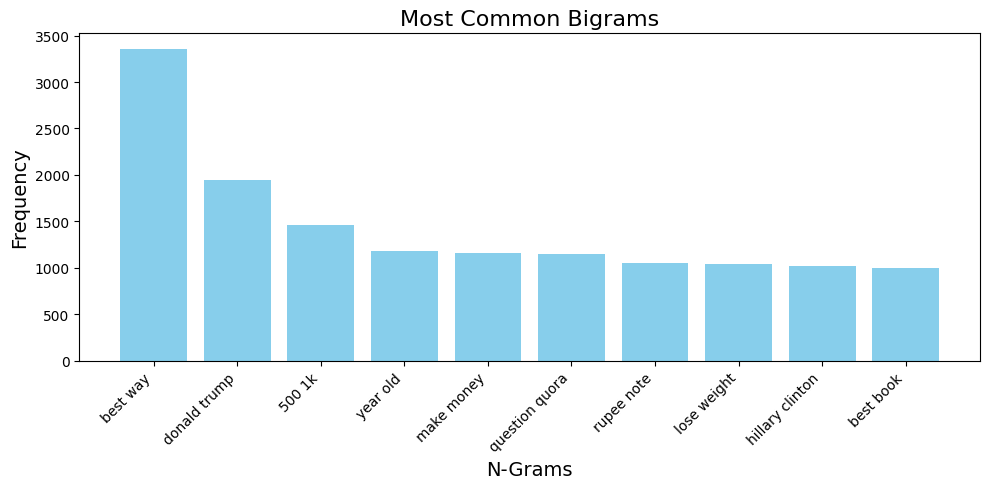
* Total number of Unique Questions: 537929
* Number of unique questions that appear more than one time: 111778 (20.78%)
* Maximum number of times a single question is repeated: 157

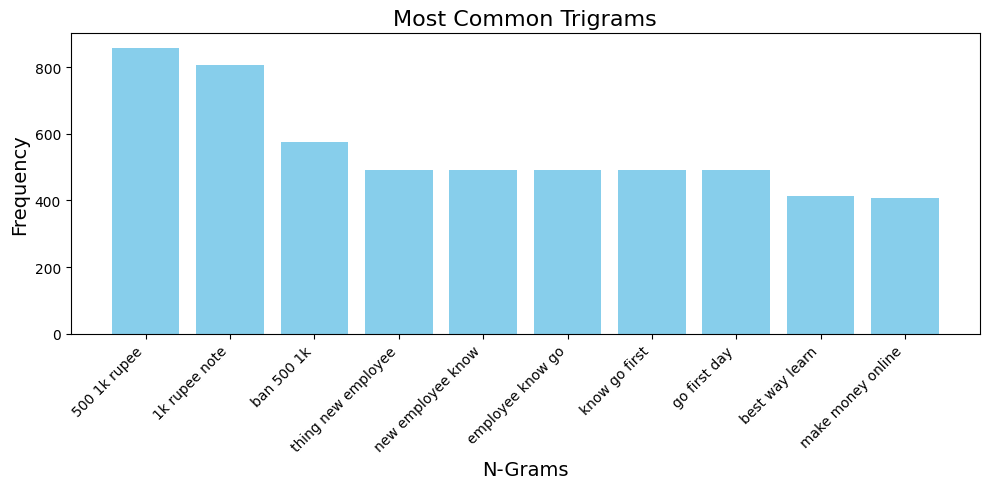
1. **Character Set and Missing Values**:
   * A total of 6,228 questions contain non-ASCII characters, spanning 8,744 question pairs.
   * Two pairs include an empty string in one of their questions. These rows were identified and removed during preprocessing.
2. **Question Lengths**:
   * The lengths of questions vary significantly across the dataset. Distribution analysis revealed that some questions are extremely short (e.g., one or two words), which may impact model performance.
   * Mostly the Length of the Questions are Below 50 words.



1. **N-Gram Analysis**:
   * To better understand linguistic patterns in the dataset, unigrams, bigrams, and trigrams were extracted from the text data after removing stopwords. This provided insights into common word combinations and their frequencies.







## Pre-Processing

The preprocessing pipeline was meticulously designed to clean and standardize the textual data in the Quora Question Pairs dataset. This process involved several key steps, each serving a specific purpose in enhancing the quality and consistency of the data for subsequent analysis.

**Lowercasing and Contraction Expansion**

The first step involved converting all text to lowercase, ensuring uniformity and preventing case-sensitive mismatches during analysis. This was followed by expanding common contractions to their full forms (e.g., "can't" to "cannot"), further standardizing the text and reducing variability.

**Character and Symbol Replacement**

Special characters and symbols were systematically replaced with meaningful equivalents to maintain semantic integrity. This included converting numerical representations (e.g., ",000,000" to "m" for million) and replacing currency symbols with their corresponding word forms. Such transformations preserved the original meaning while simplifying the text structure.

**Punctuation Removal and Tokenization**

Non-alphanumeric characters, including punctuation, were removed using regular expressions to clean the text. The cleaned text was then tokenized, splitting sentences into individual words. This step prepared the text for more granular analysis and processing.

**Lemmatization and Non-ASCII Character Handling**

Lemmatization was applied to reduce words to their base or root forms, utilizing WordNetLemmatizer from NLTK. This process was enhanced by incorporating part-of-speech tagging to improve accuracy. Additionally, non-ASCII characters were identified and either replaced or removed to ensure compatibility across all processing steps.

**Null Value Management**

The final preprocessing step involved handling null values in the dataset. Empty strings in the question columns were replaced with NaN, and rows containing null values in either question column were removed. This ensured that the dataset contained only complete and valid entries for analysis.

## Feature Extraction

Feature extraction plays a critical role in transforming raw textual data into meaningful numerical representations that can be utilized by machine learning models. For the Quora Question Pairs dataset, several features were engineered to capture lexical, semantic, and structural similarities between question pairs. Below is a detailed breakdown of the feature extraction steps:

**1. Basic Features**

The basic features form the foundation of question pair analysis, focusing on fundamental properties such as length and word count. These features include the character length of each question (q1len and q2len) and the number of words in each question (q1\_n\_words and q2\_n\_words). By capturing these basic attributes, we establish a baseline for comparing the structural similarities between question pairs, which can be indicative of their potential duplicity or relatedness.

**2. Frequency-Based Features**

Frequency-based features provide insights into the prevalence of questions within the dataset. By calculating the frequency of question IDs (freq\_qid1 and freq\_qid2) and their combinations (freq\_q1+q2 and freq\_q1-q2), we can identify patterns in question occurrence. These features help in understanding whether certain questions are more common or if there are notable differences in the frequency of paired questions, potentially revealing underlying relationships or trends in the data.

**3. Word Overlap Features**

Word overlap features quantify the lexical similarity between question pairs. By measuring the number of common unique words (word\_Common) and the total unique words (word\_Total), we can calculate the word share ratio. This ratio provides a normalized measure of lexical overlap, offering valuable insights into the semantic similarity of questions. Higher word overlap may indicate a greater likelihood of questions being duplicates or closely related.

**4. Token-Based Features**

Token-based features delve deeper into the structural and semantic similarities between questions. By analyzing common token ratios (ctc\_min, ctc\_max), stopword ratios (csc\_min, csc\_max), and first/last word matches (first\_word\_eq, last\_word\_eq), we capture nuanced relationships between question pairs. These features are particularly useful in identifying questions that may be phrased differently but convey similar meanings.

**5. Structural Features**

Structural features focus on the overall composition of question pairs. The absolute length difference (abs\_len\_diff) and mean length (mean\_len) provide insights into the structural similarity of questions. These features can help identify cases where questions may be rephrased versions of each other or where significant structural differences might indicate distinct questions.

**6. Fuzzy Matching Features**

Fuzzy matching features employ advanced string similarity algorithms to capture textual similarities that may not be apparent through exact matching. By utilizing various fuzzy ratios (fuzz\_ratio, fuzz\_partial\_ratio) and token-based ratios (token\_set\_ratio, token\_sort\_ratio), we can identify questions that are semantically similar despite minor variations in wording or structure.

**7. N-Gram Features**

N-gram features capture patterns across word combinations by analyzing unigrams, bigrams, and trigrams extracted from questions after removing stopwords. By calculating the frequencies of these n-grams, we can identify common patterns and phrases that may indicate similarity between questions. This approach is particularly effective in capturing local context and idiomatic expressions.

**8. Semantic Similarity Features**

Semantic similarity features aim to capture deeper relationships between question pairs beyond surface-level textual similarities. The Jaccard similarity measures the overlap between sets of words, while the longest common subsequence and substring ratio provide insights into shared sequential patterns. These features are crucial for identifying questions that may be semantically equivalent despite using different vocabularies or structures.

**9. Additional Features**

Additional features further enhance our understanding of question relationships. The ratio of question lengths (ratio\_q\_lengths) captures relative size differences, while common prefix and suffix lengths (common\_prefix, common\_suffix) identify shared beginnings or endings. Difference metrics in word and character counts (diff\_words, diff\_chars) provide additional granularity in comparing question structures. These features complement the core feature set, offering a more comprehensive analysis of question pair similarities.

**10. TF-IDF Weighted Word Embeddings.**

In this step, we implement an advanced feature extraction process that combines TF-IDF vectorization with pre-trained word embeddings. The procedure begins by converting all questions to strings and creating a unified corpus. We then apply TF-IDF vectorization to this corpus, generating IDF scores for each word. A pre-trained spaCy model is loaded to provide high-quality word vectors. The core of the feature extraction Involves computing TF-IDF weighted word embeddings for each question. This is achieved by multiplying each word's vector from the spaCy model with its corresponding IDF score from the TF-IDF vectorization. The resulting vectors are averaged to create a single representative vector for each question. This process yields two new features, 'q1\_feats\_m' and 'q2\_feats\_m', which encapsulate the semantic content of each question in the pair as dense vector representations. By integrating TF-IDF scores with pre-trained word embeddings, this method captures both the relative importance of words within the specific corpus and their broader semantic meanings, providing a rich basis for subsequent analysis and modeling tasks.

## Models

After extensive data processing and feature engineering, our final dataset is now ready for modeling. We have created a rich set of **635 features** that capture various aspects of question pair similarity.

The core of our dataset includes **35 NLP features**. These features cover basic information like question lengths and word counts, as well as more advanced metrics that measure how similar the questions are in structure and meaning.

We've also added powerful word embedding features. For each question in a pair, we have a **300-dimensional vector** that represents its **semantic content**. These vectors help us capture the deeper meaning of each question, going beyond just the words used.

By combining traditional NLP features with these advanced word embeddings, we've created a **comprehensive dataset**. This approach allows us to analyze question similarity from multiple angles, considering both the surface-level text and the underlying meaning of the questions.

Our feature-rich dataset provides a strong foundation for building models that can effectively identify similar or duplicate questions. It balances simple, easy-to-understand features with sophisticated representations of language, setting the stage for accurate and insightful analysis.

For our modeling approach, we employed a comprehensive strategy to evaluate and compare different techniques for identifying duplicate questions. We began by splitting our dataset into training and test sets, using stratified sampling to ensure a balanced representation of classes. The split ratio was **70% for training and 30% for testing**, with a random state of 42 set for reproducibility.

Our **baseline model** was a Logistic Regression classifier, chosen for its simplicity and interpretability. This model provided a reasonable starting point for accuracy and served as a benchmark for more advanced techniques. We then progressed to an **XGBoost model**, a powerful gradient boosting algorithm known for its high performance across various machine learning tasks. To optimize the XGBoost model, we conducted hyperparameter tuning using RandomizedSearchCV, exploring a range of parameter combinations to find the most effective configuration.

For our most advanced approach, we implemented a **Deep Learning model** using a **Siamese BERT** (Bidirectional Encoder Representations from Transformers) architecture. This model leveraged the power of pre-trained BERT embeddings, combined with custom fully connected layers for feature extraction. The Siamese structure allowed for effective comparison of question pairs, capturing nuanced semantic relationships.

**This multi-model approach** enabled us to compare the performance of traditional machine learning techniques with state-of-the-art deep learning methods, providing a comprehensive evaluation of different strategies for duplicate question detection.

## Logistic Regression Model

**Logistic Regression** is a widely used statistical model known for its simplicity and interpretability, making it an excellent choice for establishing a baseline in classification tasks. In our analysis, we applied Logistic Regression to predict duplicate questions.

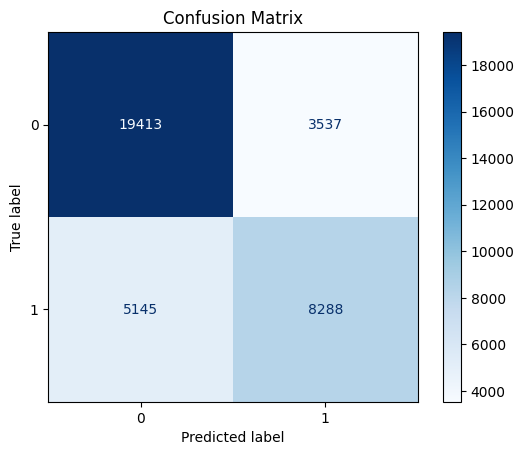
Upon training, the model's performance was evaluated using accuracy metrics and a confusion matrix. The confusion matrix, as shown in the image, provides insights into the model's classification capabilities, highlighting both correct predictions and areas for improvement. This initial analysis with Logistic Regression set the stage for exploring more complex models and refining our approach to better handle question duplication detection.

**Results:**

The Logistic Regression model demonstrated consistent performance across both training and test sets, achieving a training accuracy of 76.87% and a test accuracy of 76.14%. The confusion matrix reveals the detailed classification results:

* **True Negatives** (0,0): 19,413 cases were correctly identified as non-duplicate questions
* **False Positives** (0,1): 3,537 non-duplicate questions were incorrectly classified as duplicates
* **False Negatives** (1,0): 5,145 duplicate questions were incorrectly classified as non-duplicates
* **True Positives** (1,1): 8,288 cases were correctly identified as duplicate questions

The similar training and test accuracies suggest that the model is not overfitting, providing a reliable baseline for comparison with more advanced models



## XG-Boost

**XGBoost (eXtreme Gradient Boosting)** is an advanced implementation of gradient boosting machines known for its speed and performance. In our implementation, we utilized XGBoost with GPU acceleration to enhance computational efficiency. The model was configured with RandomizedSearchCV for hyperparameter optimization, exploring various combinations of parameters including number of estimators, learning rate, maximum depth, and sampling rates.The hyperparameter tuning process explored the following key parameters:

* Number of estimators: [50, 100, 150, 200]
* Learning rate: [0.01, 0.1, 0.2, 0.3]
* Maximum depth: [3, 5, 7, 9]
* Subsample and column sample ratios: [0.5, 0.7, 0.9, 1.0]

The model was trained using 5-fold cross-validation with 10 random parameter combinations. GPU acceleration was enabled through the **'gpu\_hist'** tree method to improve training speed. The confusion matrix visualization provides a detailed breakdown of the model's classification performance, enabling comparison with our baseline logistic regression model. This advanced modeling approach with XGBoost demonstrates our effort to leverage modern machine learning techniques while maintaining efficient computation through GPU acceleration.

**Results:**

The model achieved impressive results with a **training accuracy** of 99.89% and a **test accuracy** of 82.48%. The confusion matrix shows:

* True Negatives (0,0): 20,222 correctly identified non-duplicate questions
* False Positives (0,1): 2,728 non-duplicate questions incorrectly classified as duplicates
* False Negatives (1,0): 3,646 duplicate questions incorrectly classified as non-duplicates
* True Positives (1,1): 9,787 correctly identified duplicate questions

This represents a significant improvement over the Logistic Regression model, with better classification performance across all metrics. The high training accuracy suggests some overfitting, but the model maintains strong performance on the test set.

## Siamese BERT Model

**Siamese BERT Architecture**

Our most sophisticated approach employs a Siamese BERT architecture, leveraging the power of pre-trained BERT (Bidirectional Encoder Representations from Transformers) for question similarity detection. The model utilizes the 'bert-base-uncased' variant, which provides contextual embeddings through bidirectional training. The Siamese structure consists of two identical BERT networks sharing weights, followed by custom fully connected layers that reduce the 768-dimensional BERT embeddings to more compact 128-dimensional representations.

**Data Processing and Augmentation**

The data preparation pipeline incorporates advanced tokenization and augmentation techniques. Questions are tokenized using BERT's tokenizer with a maximum sequence length of 128 tokens. To enhance model robustness and prevent overfitting, we implement data augmentation using WordNet-based synonym replacement. This augmentation strategy expands the training dataset while maintaining semantic consistency, helping the model learn more generalizable features.

**Model Training and Optimization**

The training process employs a contrastive loss function with a margin of 0.5, specifically designed for Siamese networks. The model is optimized using Adam optimizer with a learning rate of 2e-5, complemented by a ReduceLROnPlateau scheduler for adaptive learning rate adjustment. To prevent gradient explosion, we implement gradient clipping with a maximum norm of 1.0. The training spans 10 epochs, with model checkpointing based on validation performance using Spearman's rank correlation as the metric.

**Performance Evaluation**

The model's performance is evaluated using both loss metrics and Spearman's rank correlation coefficient, providing insights into both the absolute prediction accuracy and the relative ranking capability. This sophisticated deep learning approach demonstrates the potential of transformer-based architectures in capturing subtle semantic relationships between question pairs, offering a more nuanced understanding of question similarity compared to traditional machine learning approaches.

Results:

The Siamese BERT architecture demonstrated progressive improvement over training:

* Final Test Loss: 0.0548
* Test Correlation: 0.6199
* Training Loss decreased from 0.0682 to 0.0177 over 10 epochs
* Validation Correlation improved from 0.4695 to 0.6195

The model showed consistent improvement in validation correlation up to epoch 10, with the best performance achieved in the final epoch.

**Conclusion:**

In conclusion, our comprehensive analysis of question pair similarity detection using three distinct models - Logistic Regression, XGBoost, and Siamese BERT - demonstrates the progressive improvement in performance as we move from simple to more sophisticated approaches.The baseline Logistic Regression model provided a solid foundation with a test accuracy of 76.14%, offering a balanced performance in identifying both similar and dissimilar question pairs. The XGBoost model, leveraging gradient boosting and optimized hyperparameters, significantly enhanced the predictive power, achieving a test accuracy of 82.48%. This improvement is evident in the confusion matrix, which shows a notable increase in correctly identified true positives and true negatives.The Siamese BERT model, while not directly comparable in terms of accuracy due to its different evaluation metric, showed consistent improvement throughout training, culminating in a test correlation of 0.6199. This advanced deep learning approach demonstrates the potential for capturing nuanced semantic relationships between questions, going beyond surface-level textual similarities. Each model offers unique strengths: Logistic Regression provides interpretability, XGBoost delivers high accuracy with efficient computation, and Siamese BERT excels in understanding complex language patterns. This multi-model approach not only highlights the advancements in question similarity detection but also provides a robust framework for choosing the most appropriate model based on specific application requirements, balancing between simplicity, accuracy, and depth of language understanding.

**Contributions:**

**Sai Srinivas Lakkoju's Contribution**

I took charge of the initial stages, focusing on data exploration, preprocessing, and feature engineering. I began by thoroughly analyzing the Quora Question Pairs dataset, identifying key characteristics such as class imbalance and question uniqueness. This analysis was crucial for understanding the challenges we would face in our modeling efforts.

I implemented a comprehensive preprocessing pipeline that included lowercasing, contraction expansion, and special character handling. This standardization was essential for ensuring consistency across our dataset. I also managed null values and conducted exploratory data analysis to gain deeper insights into the dataset's structure.

In the feature extraction phase, I developed a wide array of features to capture various aspects of question similarity. These included basic features like question length and word count, frequency-based features, and word overlap features. I utilized NLTK and spaCy libraries to extract linguistic features and generate word embeddings. One of my key contributions was implementing TF-IDF weighted word embeddings, which combined TF-IDF vectorization with pre-trained word vectors to create rich semantic representations of questions.

For the modeling phase, I took responsibility for implementing and fine-tuning the Logistic Regression and XGBoost models. I conducted hyperparameter optimization for XGBoost using RandomizedSearchCV, which significantly improved its performance. I also analyzed the performance of both models using confusion matrices and accuracy metrics, providing a solid baseline for comparison with more advanced techniques.

**Vinay Vasetti's Contribution**

I focused on advanced feature engineering and the development of our deep learning model. Building upon Sai's initial feature set, I implemented fuzzy matching features, n-gram analysis, and semantic similarity measures. I utilized the FuzzyWuzzy library for string matching and developed custom functions to calculate Jaccard similarity and other advanced metrics, which proved crucial in capturing nuanced relationships between question pairs.

The centerpiece of my contribution was designing and implementing the Siamese BERT architecture. I leveraged the 'bert-base-uncased' model and developed a custom neural network structure with shared weights. This approach allowed us to capture deep semantic relationships between questions that simpler models might miss.

To enhance the robustness of our model, I implemented data augmentation techniques using WordNet-based synonym replacement. This strategy helped prevent overfitting and improved the model's ability to generalize across various question formulations.

I also optimized the training process for the Siamese BERT model, implementing contrastive loss, gradient clipping, and learning rate scheduling. These techniques were essential in managing the complexity of the model and ensuring stable training. To evaluate the model's performance, I used Spearman's rank correlation coefficient and conducted detailed analysis of the model's learning progression over multiple epochs.

My work demonstrated the potential of transformer-based architectures in capturing subtle semantic relationships between question pairs, pushing the boundaries of what's possible in question similarity detection.