Detecting Duplicate Questions using Lexical, Syntactic and Transformer-based Features

Course: Natural Language Processing

Instructor: Papadopoulos Apostolos

Vasileios Kesopoulos

Laoutaris Nikolaos

Contents

1. Introduction

1.1 Question Similarity

Online Q&A platforms usually encounter a high volume of user-submitted questions that are either exact duplicates or just paraphrased variations of existing ones. Identifying these semantically similar questions is useful for organizing content which improves search relevance by minimizing redundant answers. The challenge of this problem lies in determining whether a given pair of questions express the same intent even though different words and structure may exist. This means that semantically equivalent questions can differ lexically and syntactically. Therefore, the solution to this problem should include linguistic and semantic understanding.

1.2 Objective

The objective of this project is to build a predictive model that can correctly determine if two questions are duplicates, using a combination of engineered textual features and advanced language representations. Our solution attempts to integrate classical similarity metrics as well as statistical embeddings (like TF-IDF), linguistic features and transformer embeddings to generate a diverse feature set. These features are then used to train a classifier, specifically an XGBoost model, to perform binary classification. The scope of this hands-on work includes data preprocessing, feature engineering, model training, evaluation, error analysis and hyperparameter optimization using Optuna.

1.3 Dataset Overview

The dataset used for duplicate question detection is sourced from Quora and consists of pairs of questions labeled to indicate whether they are duplicates. Each entry includes a unique identifier (id), question pair identifiers (qid1, qid2), the text of the two questions (question1, question2), and a binary label (is_duplicate) where 1 means that the questions are equivalent and 0, that they are not equivalent.

2. Data Preprocessing

The initial steps for building a duplicate question detection system were to create a data preprocessing pipeline for cleaning standardization of the input text. This preprocessing is important because it reduces noise and ensures that the representations of the questions are consistent. Additionally, to accommodate different modeling needs, three cleaning pipelines were implemented, varying in the degree of cleaning applied. The *squeaky* pipeline performs extensive normalization including stemming, while *light* cleaning omits stemming and *transformer* cleaning applies minimal normalization suited for transformer-based models that rely on subword tokenization. The individual preprocessing actions are provided in the following subsections.

2.1 HTML Tag Handling

We applied regular expressions to detect and replace HTML tags with meaningful token placeholders, preserving the tag type as a token. This step ensures that HTML markup does not distort semantic understanding while retaining potentially relevant contextual cues.

2.2 Contraction Expansion and Possessive Removal

English contractions were systematically expanded to their full forms using a contraction library. Additionally, possessive suffixes ('s) were removed to further normalize the text. This reduces vocabulary fragmentation caused by multiple forms of the same base word, aiding the model in better capturing semantic similarity.

2.3 Unicode Normalization

To handle diverse character encodings and special characters, accents and diacritics were stripped and typographic variations such as curly quotes and different dash characters were standardized to common ASCII equivalents. This harmonization ensures uniform representation of characters across all questions.

2.4 Currency Symbol Replacement

Currency symbols present in the text were identified and replaced with their three-letter ISO currency codes. This conversion prevents confusion caused by symbol variability and improves consistency in numerical and financial contexts.

2.5 Text Cleaning and Tokenization

Additional cleaning involved removing punctuation, extra whitespace and stopwords. Words were then stemmed to their root forms using the Porter Stemmer. This enhances the model’s ability to recognize semantically similar content despite lexical differences.

3. Feature Engineering

3.1 Lexical-Based Features

3.1.1 Common Word Count

The common word count feature simply counts the number of unique words shared between the two questions. This feature provides a straightforward indication of how much content is shared, which might be useful for identifying pairs that are likely to be duplicates due to similar wording.

3.1.2 Jaccard Similarity

Jaccard similarity measures the lexical overlap between two questions by comparing the sets of unique words in each. It is calculated as the size of the intersection divided by the size of the union of the word sets. This metric captures how much vocabulary is shared between the questions, providing a simple indicator of similarity.

3.1.3 Levenshtein Similarity Ratio

Levenshtein similarity quantifies the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one question into the other. By normalizing this distance, we obtain a similarity score that reflects how closely the questions match at the character level, making it sensitive to small textual differences.

3.1.4 TF-IDF and Dimensionality Reduction

Term Frequency-Inverse Document Frequency (TF-IDF) is a widely used technique for representing text data as numerical vectors. In this project, each question was converted into a TF-IDF vector, where each dimension reflects the importance of a word in the question relative to its frequency across the entire dataset.

Because TF-IDF vectors can be high-dimensional and sparse, applying dimensionality reduction techniques such as Singular Value Decomposition (SVD) can help capture the most salient patterns in the data. By projecting the original TF-IDF vectors into a lower-dimensional space using SVD, we obtained dense representations that summarize the main topics or themes present in the questions. This approach helps to mitigate noise and redundancy in the original TF-IDF features, potentially improving the model's ability to use underlying semantic structure.

To compare the vector representations of question pairs, we employ several similarity and distance metrics. Each metric captures a different aspect of the relationship between the two questions:

- Cosine Similarity: Measures the cosine of the angle between two vectors, indicating how similar their directions are. High cosine similarity suggests that the questions share similar semantic content, regardless of their magnitude.
- Absolute Difference (Mean): Computes the mean of the absolute differences between corresponding elements of the two vectors. This metric reflects the overall distance between the vectors in feature space, with lower values indicating greater similarity.
- Dot Product: Calculates the sum of the products of corresponding vector elements. The dot product captures both the magnitude and alignment of the vectors and is often higher when both vectors are large and point in similar directions.
- Element-wise Multiplication Mean: Averages the element-wise products of the two vectors. This metric highlights features that are simultaneously strong in both questions, emphasizing shared signal strength.

By combining these metrics, we obtain a richer assessment of similarity between question pairs, which might improve the effectiveness of our model.

3.2 Syntactic Features

3.2.1 Part-of-Speech (POS) Overlap Ratio

The POS (Part-of-Speech) overlap ratio compares the grammatical structure of two questions by examining their sets of POS tags. It is computed as the ratio of shared POS tags to the total unique POS tags present in both questions. This feature helps capture syntactic similarity, which can be useful for identifying paraphrased questions with similar grammatical patterns.

3.3 Transformer-Based Embedding Features

Transformer-based models, such as those from the Sentence Transformers library, provide powerful contextual embeddings for text. We used several pre-trained transformer models to encode each question into a dense vector that captures its semantic meaning. For each pair of questions, we compute the similarity between their embeddings using the same metrics discussed previously. These metrics allow us to quantify the semantic closeness of the questions beyond simple lexical overlap, enabling the detection of duplicates even when the wording differs significantly but the underlying meaning is similar. Using multiple transformer models further enriches the feature set by capturing diverse aspects of semantic similarity.

3.4 Text Cleaning Strategies for Feature Extraction

Our pipeline was designed to maximize predictive performance by using different levels of cleaning per feature.

The "squeaky-clean" approach aggressively removed special characters, stopwords, numbers and punctuation, which was especially effective for producing stable input to token-based features such as Jaccard similarity and common word count, where noise reduction improves overlap detection.

The "light-clean" approach applied smoother normalization (for example lowercasing and punctuation removal) which preserved more of the original context and syntax, making it suitable for TF-IDF-based features and bag-of-words comparisons, where retaining lexical structure is beneficial.

Finally, the "transformer-clean" approach focused on minimal intervention, ensuring that semantic-rich sentences were preserved for sentence embedding models like MiniLM and DistilBERT, which are sensitive to sentence structure and rely on contextual cues.

This modular cleaning-to-feature mapping allowed each feature to be generated under optimal text conditions, ultimately enhancing the diversity and effectiveness of the set of features used in the final predictive model.

4. Experimental setup and Results

4.1 Modeling

Because the XGBoost model performs well on structured data and can handle both linear and non-linear patterns, it was chosen to build our classifier. Five-fold cross-validation was performed, and the model was fitted using a simple setup. Evaluation metrics were accuracy (to measure overall performance) and log loss (to evaluate the model's probabilistic calibration). These metrics give a brief overview of performance and serve as a standard against which subsequent tuning attempts can be evaluated.

The evaluation results demonstrate the model's satisfactory performance, with a 5-fold cross-validation accuracy of 85.13% and a log loss of 0.3183. These metrics show that the classifier has a strong discriminative power and is well-calibrated for determining if Quora question pairs are duplicates.

Transformer-based embeddings proved to be particularly effective, according to feature importance analysis. Paraphrase MiniLM L12 mean absolute difference of embeddings was the most significant feature, accounting for more than 26% of the total importance. Conventional NLP features such as Levenshtein similarity and Jaccard similarity also had significant effects, indicating that the best outcomes are obtained by a hybrid strategy that combines deep and conventional semantic features. Interestingly, the model's comprehension of semantic similarity was improved by a number of cosine similarity and dot product features from different transformer models (such as MiniLM and Quora DistilBERT).

On the other hand, characteristics like TF-IDF multiplication mean and POS overlap ratio made very little contribution, suggesting that they may be redundant or weakly correlated with the target. Multiplication mean particularly did not seem to be a powerful vector similarity metric, so we omitted its associated features from the project, keeping only one for reference.

The specific values for feature importances measured through a permutation importance scheme can be viewed in the following table.

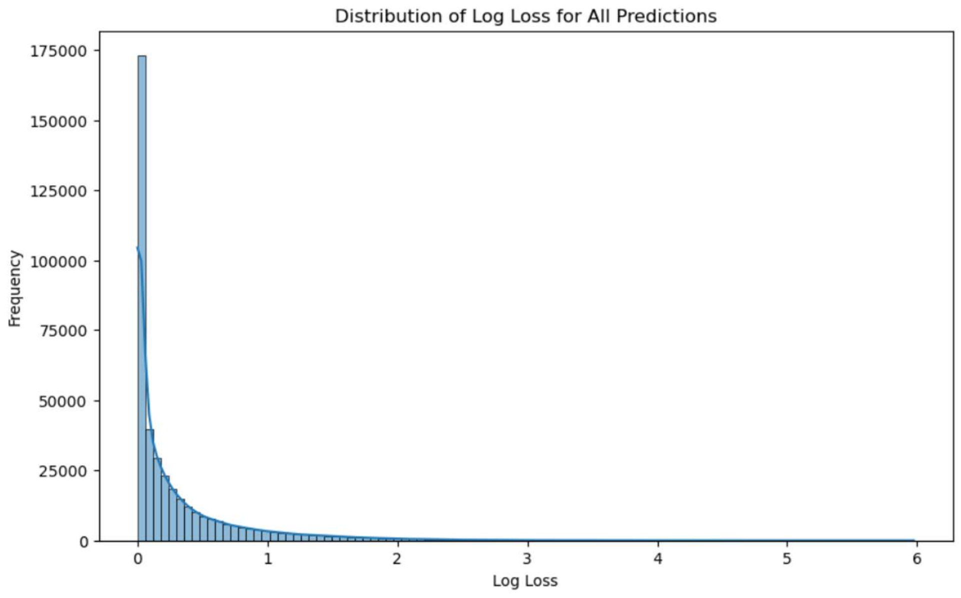
Feature	**Feature Importance**
Paraphrase MiniLM L12 mean absolute difference	0.2611
Jaccard similarity	0.1429
DistilBERT mean absolute difference	0.0561
Paraphrase MiniLM L12 cosine similarity	0.0492
TF-IDF mean absolute difference	0.0451
TF-IDF cosine similarity	0.0360
All MiniLM L6 cosine similarity	0.0334
Paraphrase MiniLM L6 cosine similarity	0.0285
Levenshtein similarity	0.0280
All MiniLM L6 dot product	0.0275
All MiniLM L6 mean absolute difference	0.0214
Common word count	0.0208
TF-IDF with SVD cosine similarity	0.0187
DistilBERT dot product	0.0157
Paraphrase MiniLM L6 dot product	0.0137
Paraphrase MiniLM L6 mean absolute difference	0.0133
Paraphrase MiniLM L12 dot product	0.0130
DistilBERT cosine similarity	0.0125
POS overlap ratio	0.0021
TF-IDF multiplication mean	0.0000

4.2 Hyperparameter Tuning

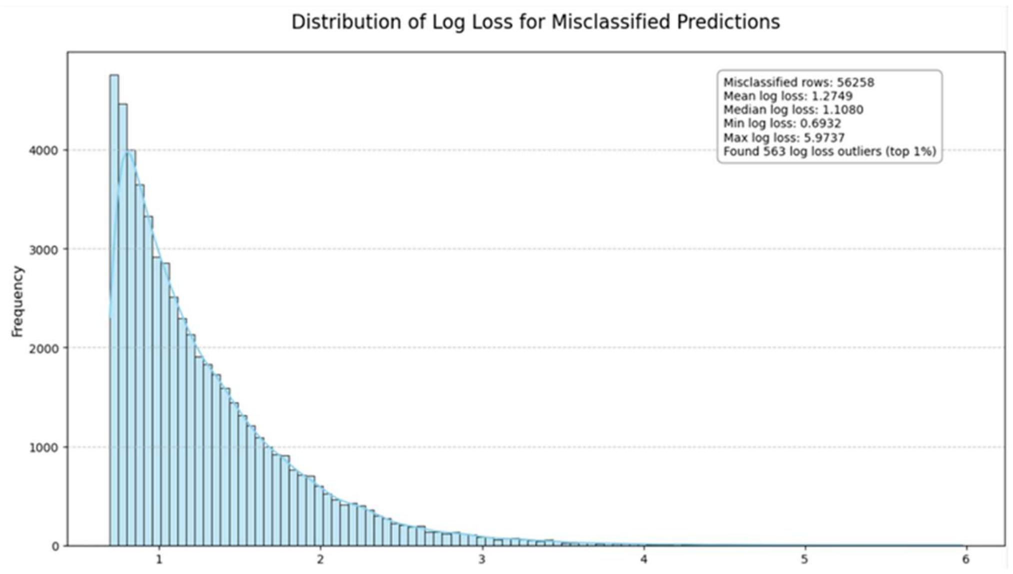
To enhance model efficiency, Optuna, a cutting-edge hyperparameter optimization framework, was utilized. The objective function established the search area for XGBoost parameters including the number of estimators, learning rate, depth and regularization factors. A Bayesian optimization method was utilized to smartly investigate favorable parameter combinations across 50 trials, aiming to maximize negative log loss, thereby enhancing both classification accuracy and probabilistic forecasts. The highest cross-validated score reached was a negative log loss of -0.3102, which shows some improvement compared to the final model's log loss of 0.3183 on the test set, suggesting effective generalization and successful adjustments.

5. Prediction Analysis and Error Inspection

Most predictions have very low log loss values, indicating confident and accurate predictions, according to the first histogram, which depicts the distribution of log loss for all predictions. This is in line with the low average log loss of 0.3183 and the overall accuracy of 85.13%.



With the majority of log losses clustered just above the 0.69 threshold (which corresponds to a 50% prediction confidence) and a tail of more difficult-to-classify events with greater loss values, the second histogram isolates misclassified predictions and displays a right-skewed distribution.



Even among errors, the model is not typically producing wildly overconfident incorrect predictions, as seen by the mean log loss for misclassified rows of 1.2749 and median of 1.1080. Also, we observe that there is a subset of predictions in which the model is very confident yet incorrect, as evidenced by the presence of 563 high-loss outliers (top 1%). However, further exploration highlighted ambiguity and data quality issues in the original input data. Despite the model's good overall performance, our experimentation indicates that certain improvements might be made by examining particular outliers.

6. Conclusion

This project addressed the Quora Question Pairs Similarity problem by developing a machine learning pipeline that integrates semantic textual similarity methods with a gradient boosting model. We began with careful preprocessing and feature engineering featuring lexical overlaps, TF-IDF similarity and advanced sentence embeddings, continued with modelling utilized the XGBoost algorithm and with hyperparameter adjustments directed by Optuna to enhance log loss optimization. The ultimate model attained a log loss of 0.3183, a negative log loss of -0.3102 using Bayesian optimization, and an accuracy rate of 85.13%, showing satisfactory performance.

An analysis of feature importance demonstrated that deep semantic features like those derived from transformer-based sentence embeddings such as paraphrase-MiniLM-L12-v2, were vital for the model's effectiveness, greatly surpassing traditional features like word overlap or Levenshtein distance. The assessment metrics were further illuminated by examining log loss distributions, which showed that most predictions were both accurate and confident. Nonetheless, an error analysis revealed that the model exhibited overconfidence in a certain group of incorrect predictions, frequently related to mislabeled data, highlighting opportunities for future improvements.

To sum up, the pipeline successfully identifies the semantic connections between pairs of questions by utilizing a balanced mix of manually designed features and embeddings from pretrained language models. Enhancements might be achieved by tackling miscalibration and data quality issues, as well as improving the model's capability to generalize across subtle paraphrasing through methods such as data augmentation, ensemble learning, or the integration of contextual transformers in a complete architecture.