REPORT: Duplicate Question Pair Detection

Abstract

We designed a system for detecting duplicate question pairs on Q&A platforms using advanced NLP and deep learning techniques. We combined identified features with contextual embeddings from BERT to capture syntactic and semantic relationships working with the Quora Duplicate Questions dataset. The resulting system is accurate and efficient, making it suitable for real-world applications such as improving search results and user experiences on Q&A platforms.
we analysed and compared the results obtained from the model to determine their effectiveness in solving the problem at hand. our project contributes to the field by showcasing the improvements achieved by transitioning from Machine Learning algorithms to advanced transformer models**.**

Introduction

NLP has shown huge improvement in the recent years in the way the computer understands and processes the language. The user-generated contents on forums, Q&A sites, and social networking create an big problem for removing duplicate questions. It cluttered databases, brought in inefficiencies, and diluted user experiences due to the presence of semantically identical but different-phrased questions. For example, "How can I lose weight quickly?" and "What are the fastest ways to lose weight?" ask essentially the same question but use different wording. It is therefore important for detection of such duplicates, since the correct detection can smooth information retrieval, raise user satisfaction. This work investigates a approach towards solution through the use of mainstream traditional machine learning, and more advanced transformer-based methodologies as well as deep-learning-based approaches to address this question for a strong, effective solution.

In our project, we have used libraries like SpaCy, NLTK, TensorFlow, PyTorch, and Hugging Face Transformers for preparing data, developing models, and evaluation results. For the basic machine learning models, we used methods such as Logistic Regression, Support Vector Machines (SVM), Random Forest, XGBClassifier, which have been improved by detailed feature engineering. We also used advanced model as BERT for meaning understanding. We have applied cross-validation for a detailed evaluation, ensuring that the proposed solution is strong.

Prior Related Work

Measuring and classifying the similarity in texts is an essential thing in duplicate question detection. Initially, token-based measures such as cosine similarity, Jaccard Index performed well for text duplicates or even near-duplicate but did a poor job in cases where semantic questions were represented differently.
Later, models such as Logistic Regression and Random Forest used methods based on features that looked at word overlap and TF-IDF. These

methods made things better but did not understand the context very well.
Recently, the task changed with deep learning models such as Word2Vec, GloVe, and BERT, which understand the context.

However, they are generally more computationally expensive and harder to understand. Our system is designed to couple advanced features with BERT embeddings, ensuring there's a balance between computation efficiency and accuracy. This approach somehow fuses the strengths from both traditional and modern approaches.

Dataset

We used the Quora Duplicate Questions Dataset, which contains 404,290 question pairs this source is helpful to our research since it was already, publicly available, and we used it after following ethical guidelines. The dataset used in our research for duplicate Question detection covers virtually all types of question pairs generated from another dialogue of question-answering websites forms and online groups.
It is carefully chosen to have both duplicate and non-duplicate question pairs for a representation balance of the two classes. Later we did proper data cleaning and preprocessing to filter out repeated information and key Keep the text data consistent.
To resolve class Imbalance, we used oversampling and under sampling technology.
The dataset's diversity enables evaluation across different domains, ensuring the models' generalizability.

labeled as:

- 1 (Duplicate): Questions have the same meaning.
- 0 (Non-Duplicate): Questions are unrelated.

Challenges We Faced:

1. Imbalanced Data: The dataset had many more non-duplicate pairs than duplicate ones, which needed careful management during training.
2. Linguistic Variability:
Questions had variations in wording, and features needed to recognize grammatical differen ces from those involving meaning**.**

How We Prepared the Data:

In the early stages we did Exploratory data analysis that helps us understand data better. It sees the distribution of duplicate numbers and non-duplicate question pairs, along with class imbalances, and looking at how long the questions are. EDA also helped us to understand variations in question formulations, assess feature correlations, and visualize the data in lower-dimensional spaces using techniques like t-SNE. The findings during our data preprocessing was feature engineering, and fine-tuning methods

contributing to our research's success in improving the model performance for duplicate question detection.

1. Cleaned Text: Removed special characters, normalized whitespace, and converted all text to lowercase.
2. Handled Stopwords: Eliminated common words like “the” and “is” that don’t add meaning.
3. Tokenized Text: Split sentences into individual words for better analysis.
4. We normalized to standardize the features to
   a normal range, with each characteristic that helps equal during model training. Normalization is especially important for distance-based algorithm. The learning ability varied due to their dimensions.

These steps ensured the dataset was clean and consistent for feature extraction.

Methodology:

During the training phase, the training function is responsible for updating the model’s parameters to minimize the chosen loss function. We explored different models for the task of duplicate question detection: Logistic regression, SVM to tune hyperparameters for optimal performance.

LOGISTIC REGRESSION was one of the building models in our approach for the duplication of questions. The choice of this algorithm was driven by simplicity, and the effectiveness of its use in binary classification.

For this model:
Feature Engineering:
We extracted textual features like TF-IDF vectors, cosine similarity scores to measure the similarity between question pairs. These features were used as input to the Logistic Regression classifier to predict whether the questions were duplicates or not.
Hyperparameter Tuning:
All of these important parameters of the model would manage overfitting with regularizatio n parameter as well as the type of regularization, 1 or 2, which improved via grid search and cross-validation. This ensured there was a good balance in the bias-variance trade-off.

SVM (Support Vector Machine):

Was used as a tool since it is efficient for modelling high-dimensional data
and strong in dealing with nonlinear separations using kernels.
Hyperparameter Optimization:
Important parameters like the regularization parameter (C) and kernel parameters were set using grid search. It was hypothesized that making these changes would increase the distance between similar and dissimilar question

pairs, thus reducing classification errors.

BERT:

Bidirectional Encoder Representations from Transformers, BERT, is smart and pre-trained language with which the deep meaning about the question pairs was understood.

Model Input:

We tokenized the question pairs with BERT's tokenizer, splitting the text into word pieces, and then passed these tokenized inputs to a fine-tuned BERT model. The [CLS] token representation from BERT was used as the feature vector for predicting semantic similarity.

Fine-Tuning:

We used our labelled dataset for fine-tuning BERT and modifying its pre-trained weights for the task of duplicate question finding. At the time of fine-tuning:
The optimizer (AdamW) and learning rate scheduler (linear warm-up) had been carefully modified in order to not overfit. Dropout layers were used during training in order to reduce overfitting and improve generalization.

Semantic Similarity Score:

The model took question pairs and provided similarity scores after fine-tuning with a threshold for classification as duplicates or non-duplicates.

Evaluation:

BERT outperformed traditional models using contextual embeddings to effectively identify nuanced semantic relationships between question pairs.

Visualization Tools We Used:

1. Pair Plots: Showed relationships between features and duplicate labels.
2. t-SNE: Created 2D and 3D visualizations to reveal clusters captured by BERT embeddings.

Metrics for Evaluation:

We evaluated the model's performance using a separate test dataset to ensure generalization on unseen data.

During training, k-fold cross-validation was employed to enhance robustness and avoid overfitting by averaging performance across k folds. Fine-tuning pre-trained models and employing data analysis, feature engineering, and optimized loss functions further improved duplicate question detection.

Experiments and Results

Results when we tested on test dataset.

Model Comparisons:

1. Baseline (Random Forest with handcrafted features):
    - Accuracy: 86%
2. BERT as Feature Extractor (with Random Forest):
    - Accuracy: 91%
3. Fine-Tuned BERT:
    - Accuracy: 94%

Key Takeaways:

Fuzzy Matching and Token Overlap:
These features were very important in measuring word similarity between question pairs and served as inputs to traditional machine learning models like Logistic Regression and SVM.

Length-Based Metrics:
Differences in word counts, character lengths, and sentence lengths helped. This was really helpful to make it easier for the machine learning models to tell apart the duplicate pairs from the non-duplicate pairs.

Contextual Embeddings (BERT):
Advanced transformer-based embeddings finally understood deeper meanings among pairs of questions, solving issues which were not possible to solve by just looking at words used, improving performance significantly--find rephrased duplicates.

Cross-Validation for Testing Models:
Careful cross-validation made sure that each model was well-tested using different datasets. This helped us adjust the hyperparameters for the best results and to choose the best approach.

Analysis

Our results confirmed that combining handcrafted features with BERT embeddings strikes the right balance between computational efficiency and accuracy. By leveraging shallow and deep text representations, we significantly improved performance.

What Worked Well:

- Interpretability: The model remained understandable, with clear insights into how features contributed to predictions.
- Transparency: Features like token overlap and fuzzy matching highlighted the linguistic properties giving the results.

Conclusion

We successfully built a system to detect duplicate questions by integrating traditional feature engineering with modern NLP techniques. BERT embeddings captured semantic relationships, leading to superior performance. Fine-tuning BERT further enhanced accuracy, making the system highly effective for real-world applications.

This project demonstrates how combining modern NLP methods with traditional approaches can produce scalable solutions. Our model can improve Q&A platforms by identifying duplicate questions and enhancing user experience.