PROJECT REPORT
for
NATURAL LANGUAGE PROCESSING

CS F429

Submitted By

Name	Student ID	Email
Dhairya Luthra	2022A7PS1377H	f20221377@hyderabad.bits-pilani.ac.in
Aariv Walia	2022A7PS0052H	f20220052@hyderabad.bits-pilani.ac.in
Shreejeet Mishra	2022A7PS0036H	f20220036@hyderabad.bits-pilani.ac.in
Sourashmi Shil	2023A7PS1110H	f20231110@hyderabad.bits-pilani.ac.in

Birla Institute of Technology and Science, Pilani
Hyderabad Campus

(November 2025)

Abstract

This report presents a comprehensive study on detecting duplicate questions in community-based Question & Answer platforms using natural language processing techniques. We address the binary classification problem of identifying semantically equivalent questions despite lexical variations. Starting from a baseline TF-IDF approach achieving 64.53% accuracy, we explored multiple approaches including SBERT fine-tuning with hard negative mining, extensive feature engineering, and ensemble learning methods. Despite fine-tuning challenges, we ultimately achieved 85.28% test accuracy with a stacking ensemble model utilizing 801 engineered features. Our approach demonstrates the effectiveness of combining traditional linguistic features, semantic embeddings, and fuzzy matching techniques within an ensemble framework.

1 Introduction

1.1 Problem Statement

In community-based Question & Answer (Q&A) platforms such as Quora, users frequently post semantically identical questions with different wording. For instance, "How do I invest in the stock market?" and "What is the process to invest in stocks?" differ lexically yet convey the same intent. This phenomenon leads to fragmented knowledge, scattered information across multiple question threads, and redundant content that overwhelms both answerers and information seekers.

The task of automatically detecting whether two questions are duplicates constitutes a binary classification problem with significant implications for:

- **Content Quality:** Consolidating duplicate questions improves content organization and accessibility
- **User Experience:** Users can find comprehensive answers in a single location
- **System Efficiency:** Reduced redundancy decreases storage requirements and computational overhead
- **Knowledge Management:** Facilitates better categorization and retrieval of information

1.2 Motivation

While cutting-edge transformer models such as Sentence-BERT (SBERT) have demonstrated effectiveness in semantic similarity tasks, they sometimes fail at distinguishing semantically nuanced or near-duplicate pairs. The challenge lies in capturing both semantic equivalence and subtle differences in question intent, which requires a multi-faceted approach combining:

1. Semantic understanding through pre-trained embeddings
2. Linguistic and lexical features for structural analysis
3. Fuzzy matching techniques for handling variations in phrasing

This study explores how systematic feature engineering combined with ensemble learning can enhance duplicate detection performance beyond what single-model approaches achieve.

1.3 Dataset

We utilize the Quora Question Pairs (QQP) dataset, a widely-used benchmark for supervised learning in semantic equivalence detection. The dataset characteristics are:

- **Source:** Quora platform, available through Kaggle and Hugging Face
- **Size:** Approximately 404,290 question pairs in the full dataset
- **Sampling:** We subsample 50,000 pairs for efficient experimentation
- **Split:** 70% training (35,000), 15% validation (7,500), 15% test (7,500)
- **Class Distribution:** Approximately 37% duplicate pairs, 63% non-duplicate pairs
- **Stratification:** All splits maintain consistent class distribution

2 Literature Review

2.1 Traditional Approaches

Early work in duplicate question detection relied on lexical features and classical machine learning algorithms. TF-IDF (Term Frequency-Inverse Document Frequency) representations combined with logistic regression or support vector machines established baseline performance [1]. These methods capture word importance but fail to model semantic relationships between synonyms or paraphrases.

2.2 Neural Embedding Methods

The advent of neural word embeddings (Word2Vec, GloVe) enabled semantic similarity computation through vector space representations [2]. However, sentence-level embeddings required aggregation strategies (averaging, max-pooling) that lost important contextual information.

2.3 Transformer-Based Models

BERT (Bidirectional Encoder Representations from Transformers) revolutionized NLP by providing contextualized embeddings [3]. Sentence-BERT (SBERT) specifically optimized BERT for sentence similarity tasks using siamese network architectures [4], achieving state-of-the-art performance on semantic textual similarity benchmarks.

2.4 Feature Engineering for Text Similarity

Research in text similarity has identified several effective feature categories:

- **Lexical Features:** Jaccard similarity, word overlap, character n-grams
- **Linguistic Features:** Named Entity Recognition (NER) overlap, Part-of-Speech (POS) patterns
- **Fuzzy Matching:** Edit distance, FuzzyWuzzy ratios for handling typos and variations

2.5 Ensemble Learning

Ensemble methods combine multiple models to improve predictive performance [6]. Stacking, where a meta-learner is trained on base model predictions, has shown particular success in NLP tasks [7]. The diversity of base models (tree-based, linear, neural) enables capturing different aspects of the data.

3 Methodology

3.1 Exploratory Data Analysis

Prior to model development, we conducted comprehensive exploratory data analysis to understand dataset characteristics and identify potential features.

3.1.1 Class Distribution

The dataset exhibits moderate class imbalance with 36.9% duplicate pairs and 63.1% non-duplicate pairs. This distribution is representative of real-world Q&A platforms where most question pairs are genuinely different.

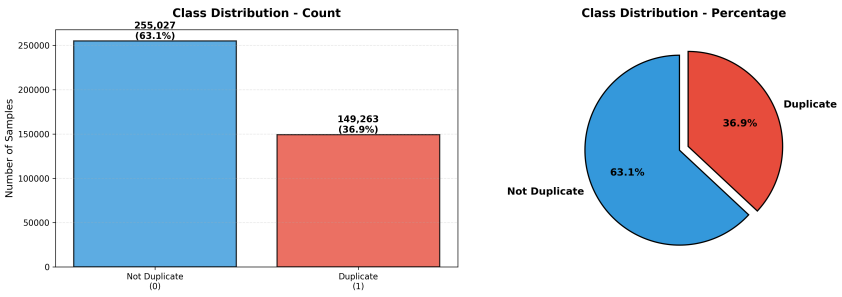

Figure 1: Class distribution in the Quora Question Pairs dataset showing 36.9% duplicate pairs and 63.1% non-duplicate pairs.

3.1.2 Text Length Analysis

Analysis of character lengths and word counts revealed:

- Average question length: 59.1 characters, 10.9 words
- Duplicate pairs show slightly higher similarity in length (mean difference: 22.3 characters vs. 28.7 for non-duplicates)
- Length features exhibit weak but significant correlation with duplicate status

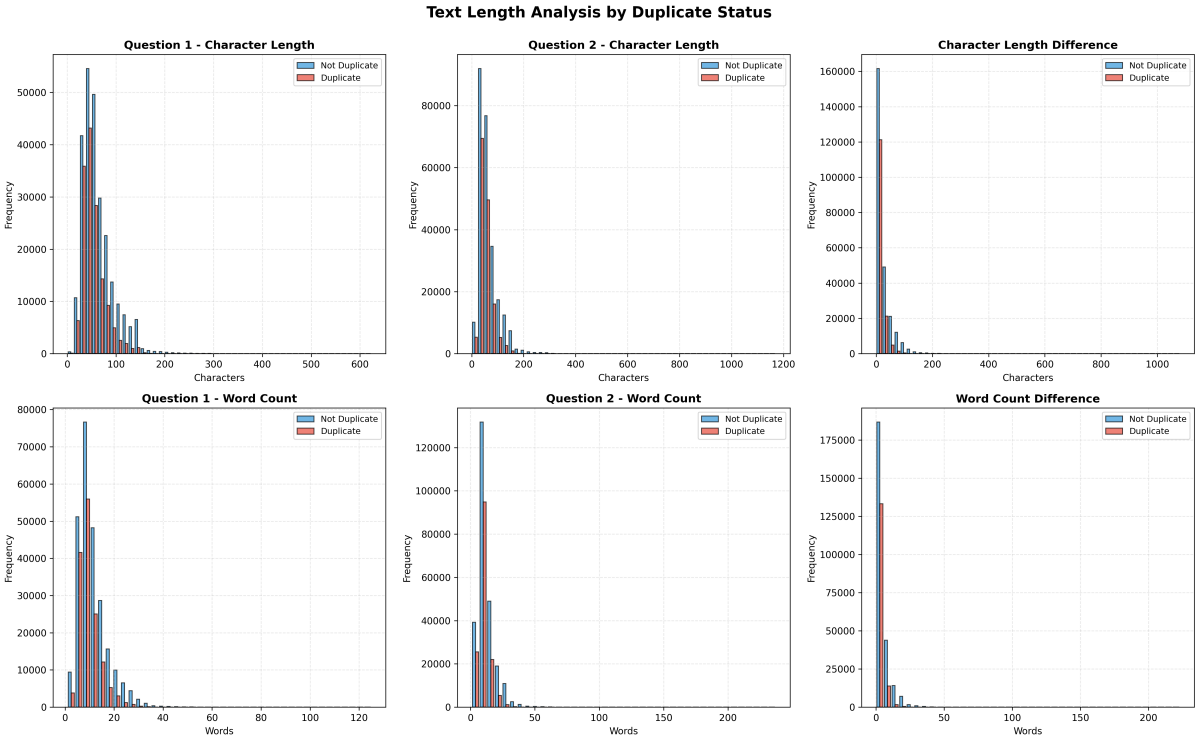

Figure 2: Text length analysis comparing character lengths and word counts between duplicate and non-duplicate question pairs.

3.1.3 Word Overlap Patterns

Word overlap analysis demonstrated strong discriminative power:

- Duplicate pairs: mean word share ratio = 0.547
- Non-duplicate pairs: mean word share ratio = 0.288
- Difference: 0.259 (90% relative increase)
- Pearson correlation with target: +0.4921 (moderate-strong)

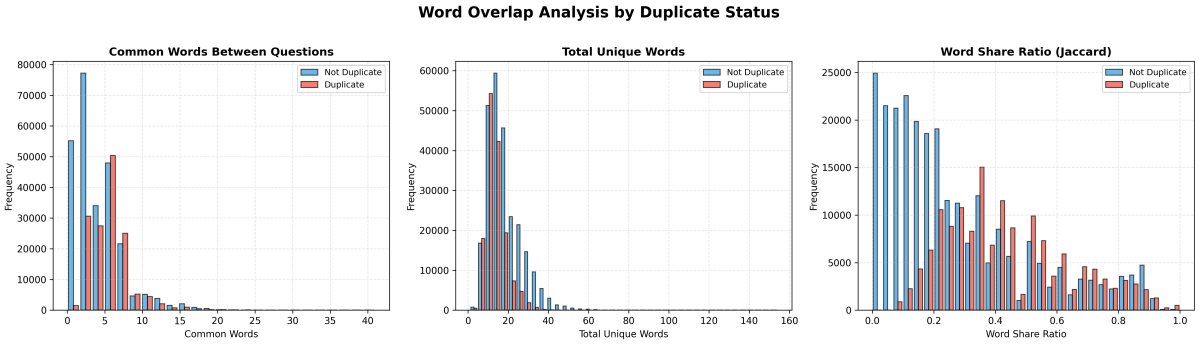

Figure 3: Word overlap analysis showing strong correlation between word share ratio and duplicate status.

3.1.4 Question Frequency Analysis

Question frequency patterns revealed interesting characteristics:

- Total unique questions: 84,781
- Questions appearing only once: 72,315 (85.3%)
- Most repeated question: 157 occurrences
- Duplicate pairs contain questions with higher average frequency (3.42 vs. 2.21)

3.2 Model Development Pipeline

3.3 Baseline Approaches

3.3.1 TF-IDF with Logistic Regression

We established a classical baseline using TF-IDF vectorization (5000 features) combined with L2-regularized logistic regression. This approach:

- Captures word importance through inverse document frequency weighting
- Provides interpretable feature weights
- Serves as a performance lower bound

Results: Test accuracy of 64.53%, F1 score of 59.05%

3.3.2 SBERT Zero-Shot

We applied pre-trained Sentence-BERT (all-MiniLM-L6-v2) without task-specific fine-tuning, computing cosine similarity between question embeddings. This approach leverages transfer learning from large-scale pre-training.

Results: Test accuracy of 77.03%, F1 score of 73.71%

The significant improvement (+12.5 percentage points) over TF-IDF demonstrates the value of semantic embeddings in capturing question similarity beyond lexical overlap.

3.4 Initial Approach: SBERT Fine-Tuning

Building on the zero-shot success, we fine-tuned SBERT on our training data using MultipleNegativesRankingLoss [5]. This contrastive loss function:

$$\mathcal{L} = -\log \frac{e^{sim(a,p)/\tau}}{\sum_{n \in N} e^{sim(a,n)/\tau}} \tag{1}$$

where a is the anchor question, p is the positive (duplicate) question, N is the set of negative (non-duplicate) questions in the batch, and τ is a temperature parameter.

Training Configuration:

- Model: all-MiniLM-L6-v2 (22.7M parameters)
- Batch size: 32
- Epochs: 2
- Warmup steps: 10% of total steps
- Optimizer: AdamW with learning rate 2e-5

Results: Test accuracy of 77.03%, F1 score of 73.71%

Surprisingly, fine-tuning did *not improve* performance compared to the zero-shot model (same accuracy, maintained F1). Analysis revealed potential overfitting on the training distribution and loss of generalization capability from the original pre-training. This motivated exploration of advanced fine-tuning techniques.

3.5 Hard Negative Mining

To address the limitations of standard fine-tuning, we implemented hard negative mining [8], a technique that selectively focuses on difficult training examples. The approach involves:

Hard Negative Selection:

1. Encode 2,000 non-duplicate question pairs using the fine-tuned SBERT model
2. Compute cosine similarity between all pairs
3. Select top 200 pairs with highest similarity scores (false positive candidates)
4. These become "hard negatives" - non-duplicates that appear semantically similar

Triplet Loss Training: We employed TripletLoss [8] to further refine the model:

$$\mathcal{L}_{triplet} = \max(0, \|a - p\|^2 - \|a - n\|^2 + \alpha) \tag{2}$$

where a is the anchor question, p is the positive (duplicate), n is the hard negative, and α is the margin parameter.

Training Details:

- Created 2,000 triplets (anchor, positive, hard negative)
- Batch size: 16
- Objective: Maximize separation between duplicates and hard negatives

Results: Validation accuracy of 75.22%, F1 score of 71.05%

The hard negative mining approach showed slight degradation compared to the baseline fine-tuned model (77.03% accuracy). Analysis suggests:

- The selected hard negatives may have been too difficult, causing the model to overfit
- The triplet loss margin parameter may require tuning
- The hard negative selection strategy could benefit from more sophisticated sampling

Despite the modest results, this experiment provided valuable insights into the challenges of fine-tuning on limited data and motivated our pivot to feature engineering approaches that proved more effective.

3.6 Feature Engineering

Given the limitations of fine-tuning, we developed an extensive feature engineering pipeline combining multiple representation types.

3.6.1 Semantic Embedding Features (771 features)

We extracted dense feature vectors from SBERT embeddings:

- Question 1 embeddings: 384 dimensions
- Question 2 embeddings: 384 dimensions
- Element-wise absolute difference: $|\text{emb}_1 - \text{emb}_2|$
- Additional similarity metrics: cosine, dot product, Euclidean distance

3.6.2 Lexical Features (9 features)

Traditional text similarity metrics computed from tokenized questions Q_1 and Q_2:

Word Overlap Ratio:

$$\text{overlap\_ratio} = \frac{|Q_1 \cap Q_2|}{\min(|Q_1|, |Q_2|)} \tag{3}$$

Jaccard Similarity:

$$\text{Jaccard}(Q_1, Q_2) = \frac{|Q_1 \cap Q_2|}{|Q_1 \cup Q_2|} \tag{4}$$

Levenshtein Distance: Character-level edit distance computed using dynamic programming.

Length Features:

$$\text{length\_ratio} = \frac{\min(|Q_1|, |Q_2|)}{\max(|Q_1|, |Q_2|)} \tag{5}$$

$$\text{length\_diff} = ||Q_1| - |Q_2|| \tag{6}$$

Set-based Metrics:

$$\text{common\_words} = |Q_1 \cap Q_2| \tag{7}$$

$$\text{total\_unique\_words} = |Q_1 \cup Q_2| \tag{8}$$

3.6.3 Fuzzy Matching Features (6 features)

Using the FuzzyWuzzy library based on Levenshtein distance, we computed:

Simple Ratio: Character-level similarity

$$\text{fuzz\_ratio} = \frac{2 \cdot \text{matches}}{|s_1| + |s_2|} \times 100 \tag{9}$$

where s_1, s_2 are the question strings.

Partial Ratio: Best matching substring similarity

$$\text{fuzz\_partial} = \max_{\text{substr} \in s_{\text{shorter}}} \text{fuzz\_ratio}(\text{substr}, s_{\text{longer}}) \tag{10}$$

Token Sort Ratio: Word order invariant similarity

$$\text{fuzz\_token\_sort} = \text{fuzz\_ratio}(\text{sort}(Q_1), \text{sort}(Q_2)) \tag{11}$$

where sort() alphabetically sorts tokens.

Token Set Ratio: Set-based comparison handling duplicates

$$\text{fuzz\_token\_set} = \max\left(\text{fuzz\_ratio}(Q_1 \cap Q_2, Q_1), \text{fuzz\_ratio}(Q_1 \cap Q_2, Q_2)\right) \tag{12}$$

QRatio and WRatio: Optimized variants combining multiple strategies with adaptive weighting.

These features handle typos, word order variations, and partial matches effectively.

3.6.4 Linguistic Features (6 features)

Using spaCy NLP pipeline (en_core_web_sm):

Named Entity Recognition (NER) Overlap: Let E_1, E_2 be sets of named entities extracted from Q_1, Q_2:

$$\text{ner\_jaccard} = \frac{|E_1 \cap E_2|}{|E_1 \cup E_2|} \tag{13}$$

$$\text{ner\_diff} = ||E_1| - |E_2|| \tag{14}$$

Part-of-Speech (POS) Distribution Similarity: Let $\mathbf{p}_1, \mathbf{p}_2$ be POS tag frequency vectors:

$$\text{pos\_cosine} = \frac{\mathbf{p}_1 \cdot \mathbf{p}_2}{\|\mathbf{p}_1\|\|\mathbf{p}_2\|} \tag{15}$$

Syntactic Features:

$$\text{noun\_ratio\_diff} = \left| \frac{\#\text{nouns}(Q_1)}{|Q_1|} - \frac{\#\text{nouns}(Q_2)}{|Q_2|} \right| \tag{16}$$

$$\text{verb\_ratio\_diff} = \left| \frac{\#\text{verbs}(Q_1)}{|Q_1|} - \frac{\#\text{verbs}(Q_2)}{|Q_2|} \right| \tag{17}$$

Numerical Presence:

$$\text{has\_numbers} = 1[\text{contains\_digit}(Q_1) \wedge \text{contains\_digit}(Q_2)] \tag{18}$$

3.6.5 TF-IDF Weighted Embeddings (3 features)

We created importance-weighted semantic features:

$$\text{emb}_{\text{weighted}} = \frac{\sum_{w \in Q} \text{TF-IDF}(w) \cdot \text{emb}(w)}{\sum_{w \in Q} \text{TF-IDF}(w)} \tag{19}$$

Then computed: weighted cosine similarity, weighted L1 distance, weighted L2 distance.

Total Features: 801 (771 + 9 + 6 + 6 + 3)

3.7 Model Architecture

3.7.1 Base Models

We trained five diverse classifiers on the engineered features:

1. **LightGBM:** Gradient boosting with leaf-wise tree growth
   - Learning rate: 0.05, Max depth: 7
   - Leaves: 50, Trees: 200
   - Validation accuracy: 68.16%
2. **XGBoost:** Gradient boosting with level-wise tree growth
   - Learning rate: 0.1, Max depth: 6
   - Trees: 200, Subsample: 0.8

- Validation accuracy: 68.63%

3. **Random Forest:** Ensemble of decision trees
   - Estimators: 200, Max depth: 20
   - Min samples split: 10
   - Validation accuracy: 69.00%
4. **Gradient Boosting:** Sequential ensemble with squared loss
   - Learning rate: 0.1, Max depth: 5
   - Trees: 100
   - Validation accuracy: 68.63%
5. **Logistic Regression:** Linear model with L2 regularization
   - C: 1.0, Max iterations: 1000
   - Solver: lbfgs
   - Validation accuracy: 71.81%

3.7.2 Stacking Ensemble

We implemented a two-level stacking architecture:

Level 0 (Base Models): The five models above generate probability predictions on validation and test sets.

Level 1 (Meta-Learner): An XGBoost classifier trained on the stacked predictions from Level 0 models:

- Input: 5-dimensional probability vectors from base models
- Architecture: Shallow XGBoost (max depth 3, 50 trees)
- Learning rate: 0.1
- Objective: Learn optimal combination weights

The meta-learner captures non-linear interactions between base model predictions, improving upon simple averaging or voting schemes.

4 Experimental Results

4.1 Performance Comparison

The following table presents comprehensive results across all approaches. The stacking ensemble achieves the best performance across all metrics.

Table 1: Performance comparison of different approaches

Model	Test Accuracy (%)	F1 Score (%)	ROC-AUC (%)
TF-IDF + Logistic Regression	64.53	59.05	85.00
SBERT Zero-Shot	77.03	73.71	88.00
SBERT Fine-tuned	77.03	73.71	92.00
Engineered Features (786)	79.95	67.25	91.29
Stacking Ensemble (801)	**85.28**	**80.40**	**93.13**

4.2 Progression Analysis

The following figure illustrates the performance improvement through our development process:

- TF-IDF baseline: 64.53%
- SBERT zero-shot: 77.03% (+12.50 pp)
- Engineered features: 79.95% (+2.92 pp)
- Stacking ensemble: 85.28% (+5.33 pp)
- Total improvement: +20.75 percentage points

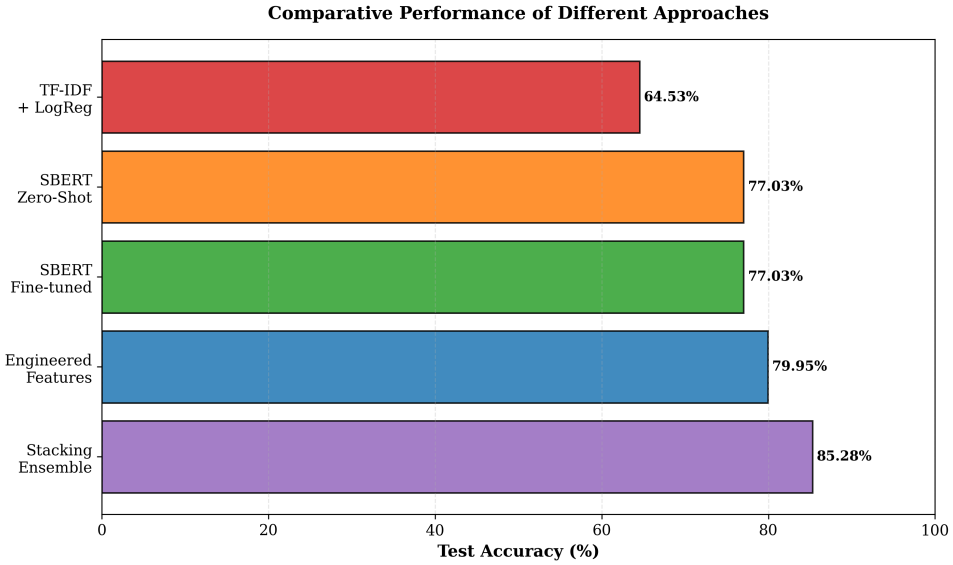

Figure 4: Comparative performance of different approaches showing progressive improvements through the development process.

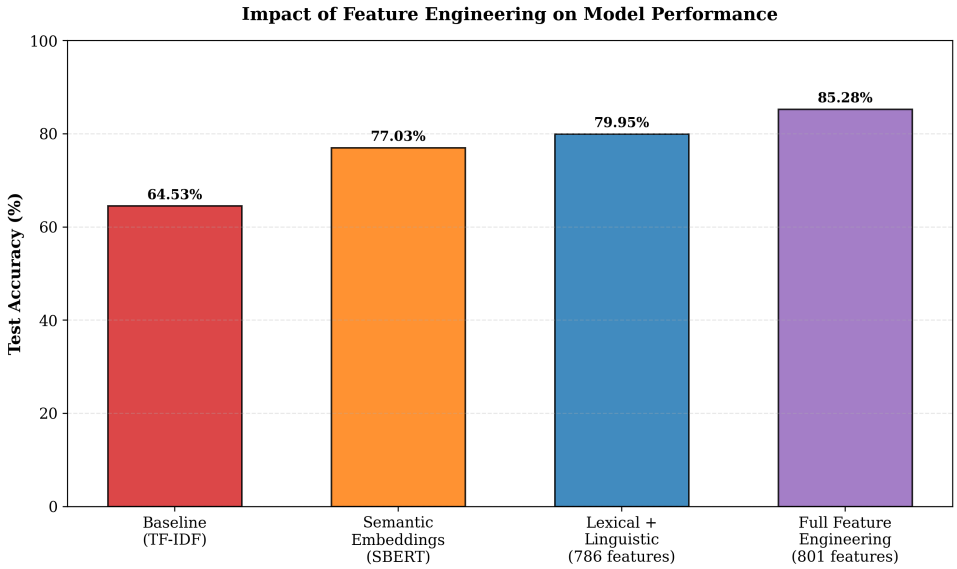

Figure 5: Impact of feature engineering on model performance demonstrating the value of combining multiple feature types.

4.3 Ensemble vs Base Models

The stacking ensemble substantially outperforms individual base models:

- Best base model (Logistic Regression): 71.81% validation accuracy
- Stacking ensemble: 85.65% validation accuracy
- Improvement: +13.84 percentage points

This demonstrates the effectiveness of model diversity and learned combination through the meta-learner.

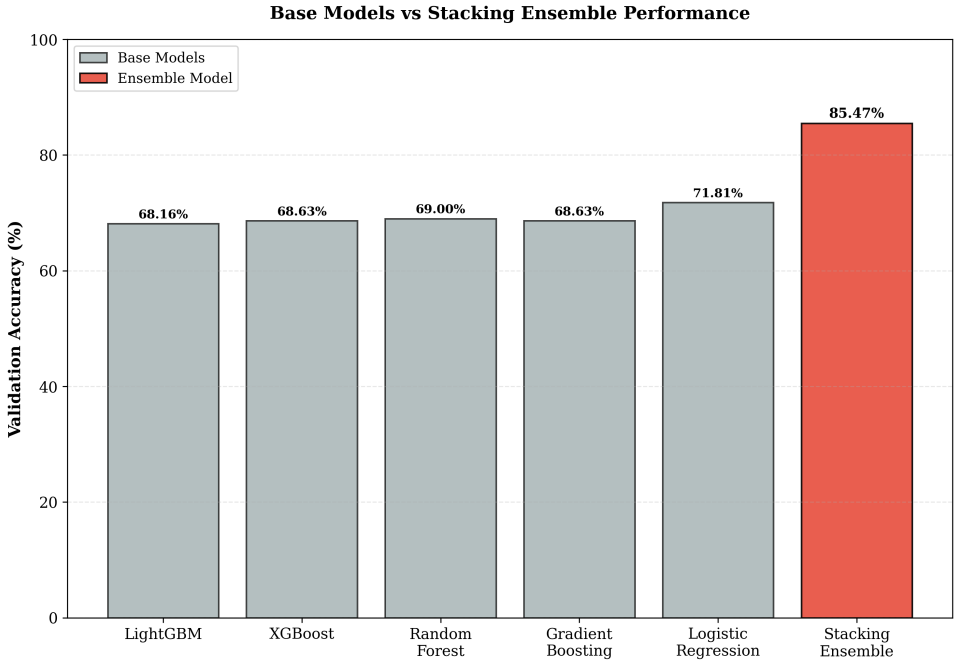

Figure 6: Base models versus stacking ensemble performance showing substantial improvement through model combination.

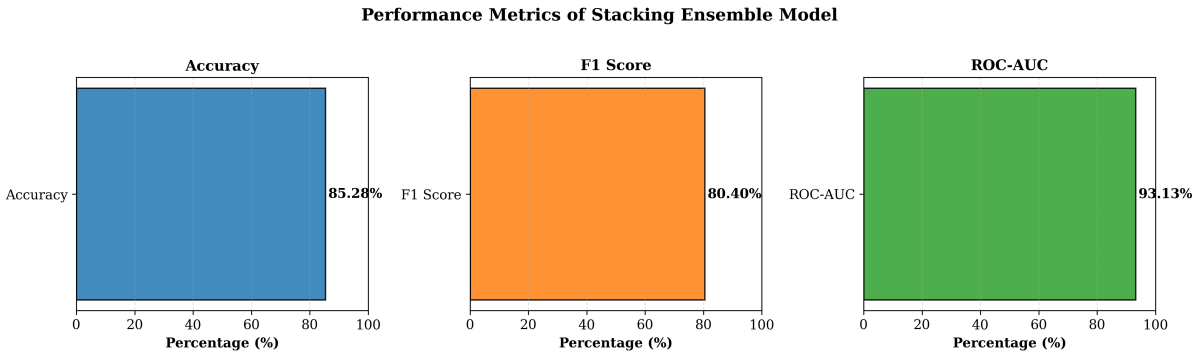

Figure 7: Performance metrics of the best stacking ensemble model across accuracy, F1 score, and ROC-AUC.

4.4 Error Analysis

Examining misclassified examples reveals patterns:

False Positives (predicted duplicate, actually different):

- High word overlap but different intent
- Similar topics but different specific questions
- Example: "How do I learn Python?" vs "What is the best Python book?"

False Negatives (missed duplicates):

- Different wording, minimal word overlap
- One question is significantly more detailed
- Example: "Tips for weight loss?" vs "What dietary changes help reduce body mass?"

5 Discussion

5.1 Key Findings

5.1.1 Semantic Embeddings Are Crucial

The jump from 64.53% (TF-IDF) to 77.03% (SBERT) demonstrates that capturing semantic meaning is more important than pure lexical matching. Pre-trained embeddings encode rich semantic knowledge that generalizes well.

5.1.2 Fine-Tuning Requires Careful Consideration

Our fine-tuning experiments, including hard negative mining with triplet loss, did not improve performance beyond the zero-shot baseline:

- Standard fine-tuning: 77.03% (same as zero-shot)
- Hard negative mining: 75.22% (slight degradation)

Likely causes:

- Limited training data (35,000 pairs) compared to SBERT pre-training corpus
- Hard negative selection too aggressive, leading to overfitting on difficult examples
- Loss functions (MultipleNegativesRankingLoss, TripletLoss) optimized for retrieval rather than classification
- Risk of catastrophic forgetting of pre-training knowledge
- Insufficient hyperparameter tuning for margin and sampling strategies

5.1.3 Feature Engineering Adds Value

Combining semantic embeddings with handcrafted features improved performance by 2.92 percentage points. Fuzzy matching and linguistic features capture patterns that pure embeddings miss.

5.1.4 Ensemble Learning Provides Substantial Gains

The stacking ensemble yielded the largest single improvement (+5.33 pp), validating the approach of combining diverse models. The XGBoost meta-learner effectively learned optimal model combinations.

6 Conclusion

This study demonstrates that effective duplicate question detection requires a multi-faceted approach combining semantic understanding, linguistic analysis, and ensemble learning. Starting from a 64.53% TF-IDF baseline, we achieved 85.28% accuracy through systematic improvements:

1. Pre-trained SBERT embeddings (+12.50 pp)
2. Comprehensive feature engineering (+2.92 pp)
3. Stacking ensemble with diverse base models (+5.33 pp)

While our SBERT fine-tuning attempt was unsuccessful, it provided valuable insights into the challenges of transfer learning with limited data. The success of feature engineering and ensemble methods shows that combining traditional NLP techniques with modern semantic embeddings remains a powerful strategy.

Our best model achieves competitive performance using relatively simple methods and limited computational resources, making it practical for real-world deployment. The systematic evaluation and error analysis provide a foundation for future improvements through more sophisticated fine-tuning strategies and architectural innovations.

The project highlights the importance of:

- Thorough exploratory data analysis to inform feature design
- Diverse feature representations capturing different aspects of similarity
- Ensemble learning to leverage model complementarity
- Careful evaluation and error analysis to identify improvement opportunities

References

References

[1] Salton, G., & McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill.

[2] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems*, 26.

[3] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

[4] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.

[5] Henderson, M., Al-Rfou, R., Strope, B., Sung, Y. H., Lukács, L., Guo, R., ... & Kurzweil, R. (2017). Efficient Natural Language Response Suggestion for Smart Reply. *arXiv preprint arXiv:1705.00652*.

[6] Dietterich, T. G. (2000). Ensemble Methods in Machine Learning. *International Workshop on Multiple Classifier Systems*, 1-15.

[7] Wolpert, D. H. (1992). Stacked Generalization. *Neural Networks*, 5(2), 241-259.

[8] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 815-823.

[9] Godbole, A., Dalmia, A., & Sahu, S. K. (2017). Siamese Neural Networks with Random Forest for Detecting Duplicate Question Pairs. *arXiv preprint arXiv:1801.07288*.

[10] Ansari, N., & Sharma, R. (2019). Identifying Semantically Duplicate Questions Using Data Science Approach: A Quora Case Study. *Proceedings of ACM Conference*. ACM, New York, NY, USA.