Quora Question Pairs

Mixture of Experts Architecture Approach

Code Implementation

Lazaros Panitsidis

Konstantinos Kravaritis

International Hellenic University

School of Science & Technology

MSc Data Science

Natural Language Processing and Text Mining

Associate Professor Apostolos N. Papadopoulos

12 June 2025

Abstract

This paper presents a modular and extensible pipeline for detecting semantically duplicate question pairs in the Quora Question Pairs (QQP) dataset. The proposed architecture combines traditional text preprocessing, diverse feature engineering, and a hybrid ensemble of classical machine learning and transformer-based experts integrated under a trainable Mixture-of-Experts (MoE) framework. Sentence representations are generated using SBERT (MiniLM and MPNet), and enriched through syntactic, lexical, and embedding-based features. Each expert is treated as a frozen module, and their outputs are aggregated using a softmax-based gating layer trained with fixed hyperparameter configurations (learning rates of 0.001, 0.01, and 0.05; epochs of 1, 2, and 10, respectively). The system achieves high predictive performance, with test log-loss as low as 0.038 and ROC-AUC exceeding 0.998, while maintaining strong inference efficiency. Lightweight CSV-based logging, deterministic data splits, and modular code structure ensure full reproducibility. This work offers a scalable foundation for semantic similarity tasks, highlighting the value of combining heterogeneous representations within a unified, trainable ensemble.

1. Introduction

Efficiently identifying duplicate questions is a critical task in community-driven platforms such as Quora, Stack Overflow, and Reddit. Duplicate detection improves user experience by reducing redundant content, consolidating answers, and optimizing search relevance. Despite the simplicity of the task definition, it involves subtle semantic reasoning, making it an ongoing challenge in natural language understanding (NLU).

Recent advances in transformer-based models have substantially improved semantic matching performance. However, many studies rely exclusively on end-to-end fine-tuning, often overlooking the benefits of combining symbolic and statistical representations. This paper adopts a hybrid strategy that leverages both classical machine learning techniques and modern pretrained transformer models. By combining these diverse paradigms under a Mixture-of-Experts (MoE) ensemble with a trainable gating layer, we demonstrate a performance improvement over any individual expert. Our method is designed to be fully modular, reproducible, and extensible to other pairwise classification tasks.

2. Problem Definition

Let **(q1, q2)** denote a pair of natural language questions. The objective is to predict whether they are semantically equivalent, meaning, duplicates. Formally, this is posed as a binary classification problem with label $\mathbf{y} \in \{\mathbf{0, 1}\}$, where $\mathbf{y = 1}$ denotes a duplicate pair.

The learning objective is to find a function $\mathbf{\hat{y} = f(q1, q2)}$ such that $\mathbf{\hat{y}} \in \mathbf{[0, 1]}$ approximates the probability of semantic duplication. The model is trained by minimizing the binary cross-entropy loss:

$$\mathbf{L = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})]}$$

Here, $\mathbf{\hat{y}}$ is the predicted probability and $\mathbf{y}$ is the ground-truth label. The performance is evaluated using standard classification metrics, as well as log-loss for probability calibration.

3. Dataset and Experimental Setup

3.1 Data Splitting Strategy

The dataset used in this study is the Quora Question Pairs (QQP) dataset, which contains over 400,000 annotated question pairs, each labeled as either duplicate or non-duplicate.

To ensure reproducibility and minimize information leakage, the data was partitioned deterministically into three mutually exclusive subsets:

- **Training set**: 80% of the data
- **Validation set**: 10% of the data
- **Test set**: 10% of the data

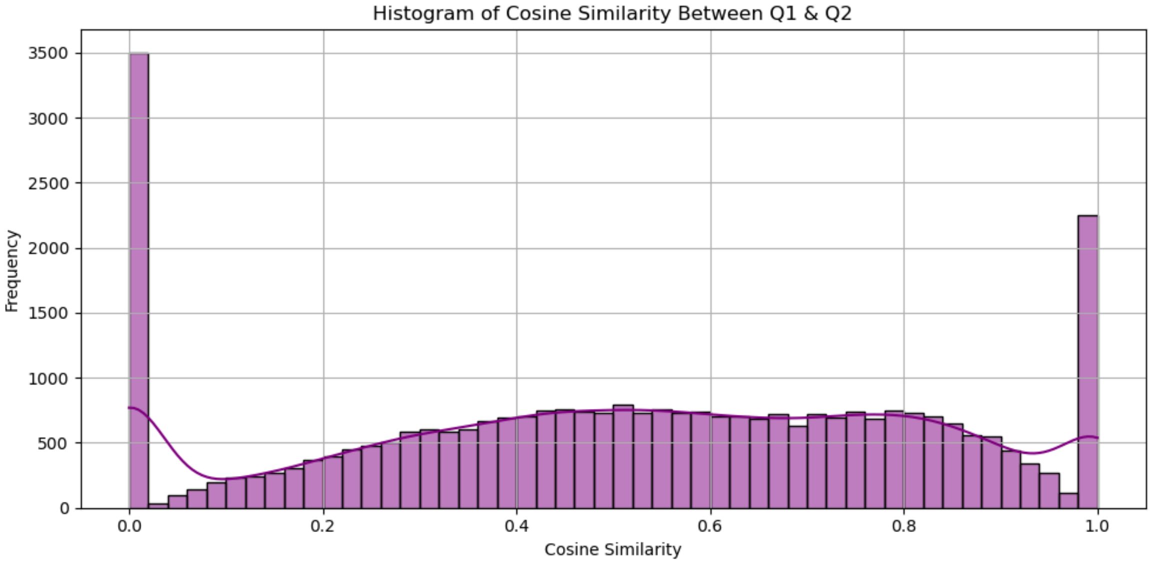
A **GroupShuffleSplit** strategy was applied, grouping by the minimum of `qid1` and `qid2`. This ensured that no semantically linked question appeared in more than one split, thus maintaining strict isolation between train, validation, and test sets. Additionally, class distribution was preserved across splits using stratified sampling to mitigate any performance bias introduced by label imbalance.

3.2 Exploratory Data Analysis (EDA)

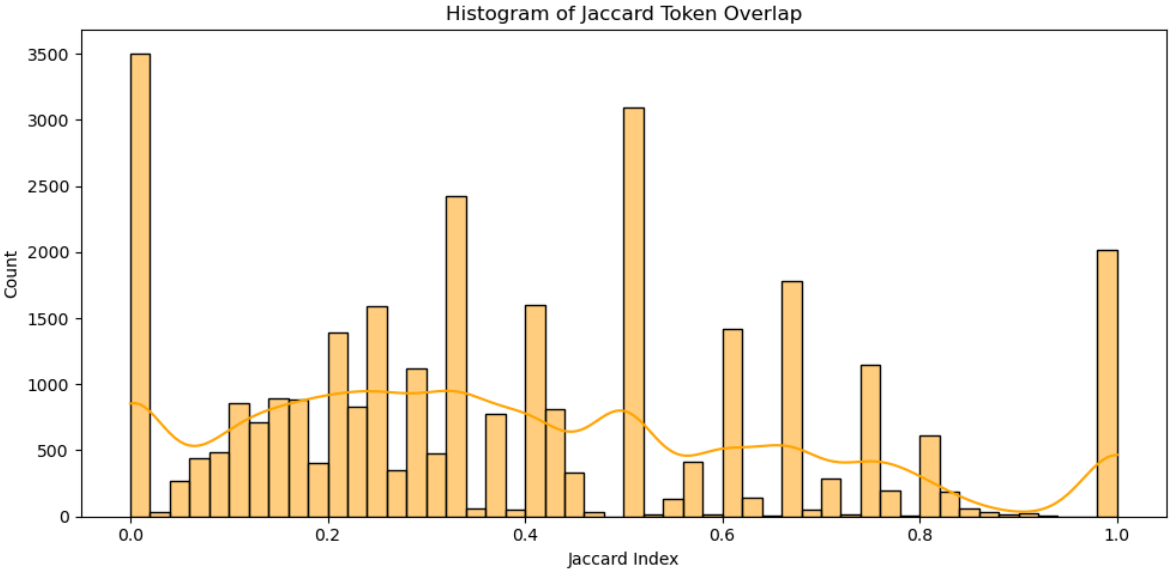
A preliminary analysis of the dataset revealed a moderate class imbalance, with approximately 37% of question pairs labeled as duplicates. This skewed distribution underlines the importance of balanced evaluation metrics such as F1 score and AUC in addition to accuracy.

Descriptive statistics were computed for both question fields, including distributions of sentence length (measured in characters and tokens). These analyses provided insight into the linguistic variability across questions and helped guide preprocessing and embedding choices.

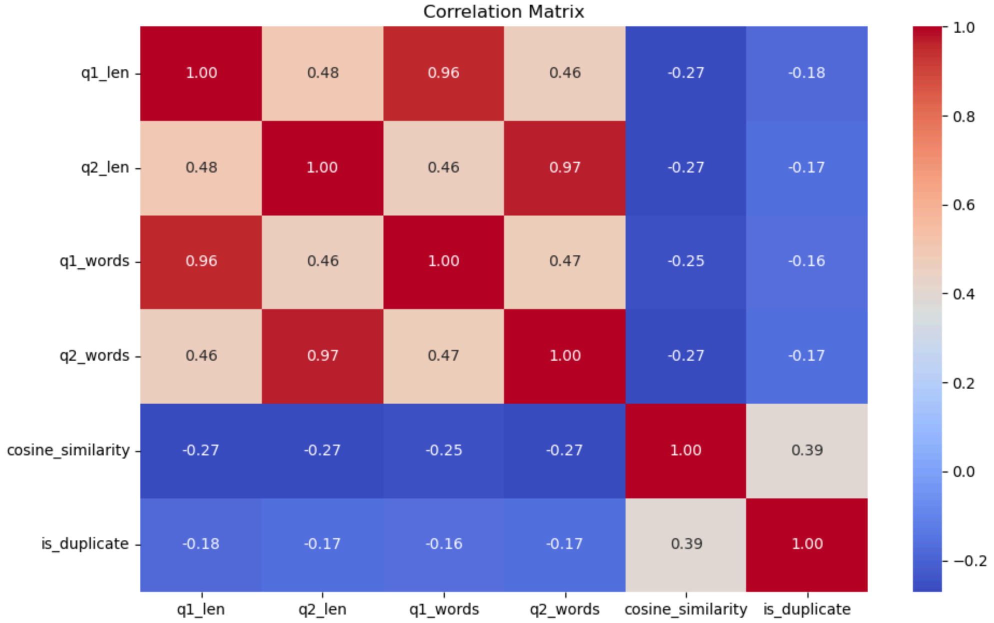
Character-level and word-level frequency analyses highlighted the prevalence of common lexical patterns. In particular, high-frequency tokens such as "what", "how", and "best" were prominent, suggesting the need for normalization and careful treatment of stopwords. The degree of lexical overlap between duplicate and non-duplicate pairs was also assessed, revealing that duplicates tend to share more content words—a cue exploited later in feature engineering.



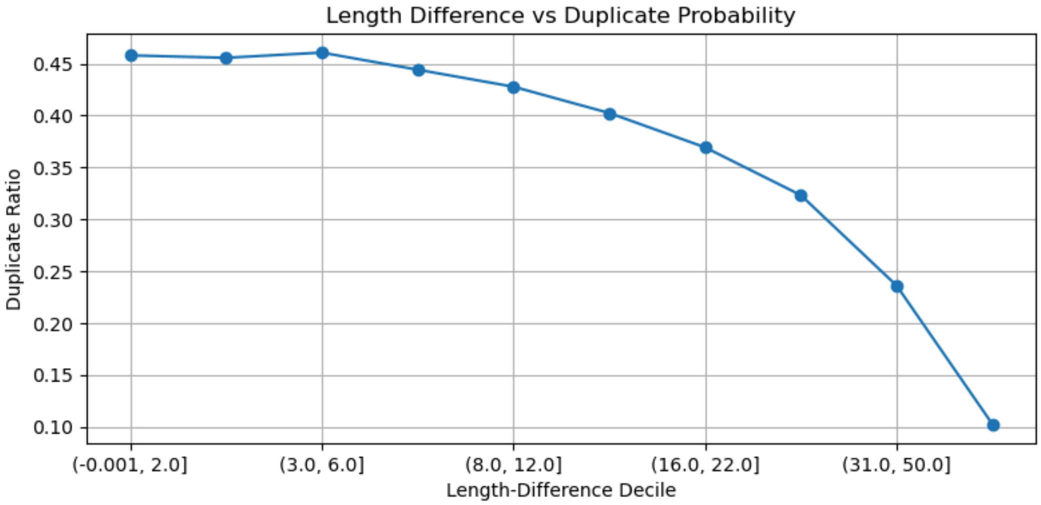
The cosine similarities of the 404 290 Quora question-pairs form a clear trimodal pattern—≈ 5.5 % at zero (unrelated questions), a broad middle range (≈ 94.5 % with partial overlap, non-duplicates mostly < 0.4 and paraphrase duplicates > 0.4), and a tiny spike at 1.0 (< 0.01 % exact repeats). A threshold around 0.7 strikes a practical balance for detecting paraphrase duplicates while excluding unrelated pairs, with higher cutoffs (≥ 0.9) reserved for exact or near-verbatim matches.



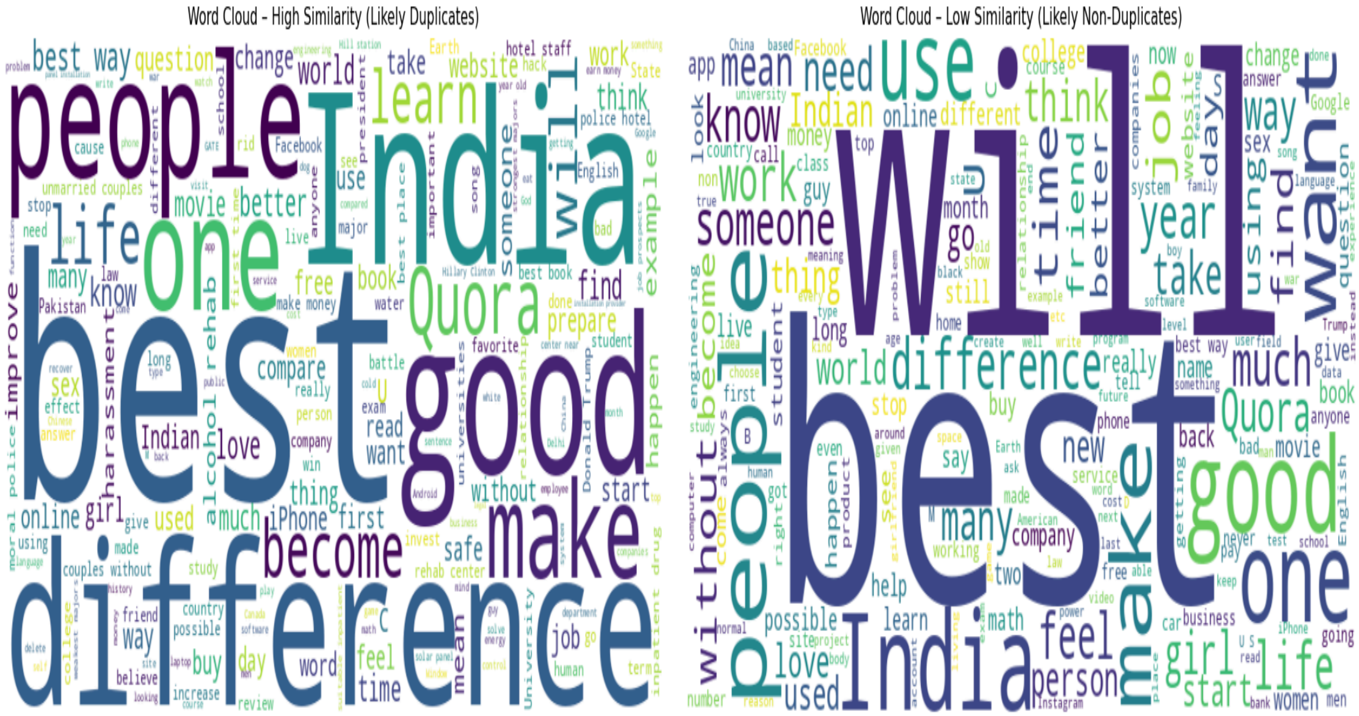
The Jaccard‑overlap distribution is trimodal: a large spike at 0 (≈5 % of pairs share no tokens), a broad, fairly uniform spread of intermediate overlaps (≈94 % of pairs with partial token overlap), and a small spike at 1 (the ~19 verbatim duplicates). This indicates most question‑pairs have some token overlap—non‑duplicates clustering at very low Jaccard scores (<0.2) and paraphrase duplicates occupying mid‑range values (0.2–0.8)—with only a vanishingly small fraction being exact repeats.



The heatmap shows that question lengths and word counts are almost perfectly inter‑correlated (r≈0.96–0.97) and moderately so across Q1/Q2 (r≈0.46–0.48), but both length and word‑count features have only weak negative correlations (≈–0.16 to –0.27) with the duplicate flag. In contrast, cosine similarity is slightly negatively correlated with question length (r≈–0.27) and shows a moderate positive correlation with duplication (r≈0.39), marking it as the strongest single predictor among these features.



Question pairs with very small length differences (≤2 tokens) have the highest duplicate rate—around 45–46%—and this duplicate probability steadily falls as length disparity grows. By the largest length‑difference decile (>31 tokens), only about 10% of pairs are duplicates, indicating that big length gaps strongly predict non‑duplication.



Word Cloud - High Similarity (Likely Duplicates)

Word Cloud - Low Similarity (Likely Non-Duplicates)

The high‑similarity cloud is dominated by generic “how‑to” and comparative terms—words like “best,” “difference,” “one,” “people,” “good,” “make,” and “learn”—reflecting the common framing of duplicate questions. By contrast, the low‑similarity cloud features a wider variety of topic‑specific tokens—e.g. “India,” “will,” “use,” “time,” and “work”—highlighting the greater semantic diversity of non‑duplicate pairs.

The 323 613‑pair training set is 36.9 % duplicates (119 283/204 330 non‑duplicates) with each question averaging ~60 chars (≈11.3 words) and a mean length difference of 20.3 chars. Semantic overlap is moderate—mean Jaccard = 0.387, mean TF–IDF cosine = 0.520—and the minimum shared token frequency has a moderate positive correlation with duplication (r≈0.406), showing that more common shared terms help predict duplicate pairs.

Metric	Value
Train Rows	323613
Duplicate Count	119283
Non-Duplicate Count	204330
Duplicate Ratio	36.860%
Mean of q1 chars	59.7
Mean of q2 chars	60.5
Mean of q1 words	11.3
Mean of q2 words	11.3
Mean len_diff	20.3
Mean Jaccard	0.387
Mean Cosine TF-IDF	0.520
Corr(min_freq, is_dup)	0.406

4. Text Preprocessing

For textual similarity tasks, where subtle surface-level differences could mask deeper semantic equivalency, effective preprocessing is crucial. In order to prepare the Quora questions for both conventional feature extraction and deep representation learning, this study used a rigorous, multi-stage pipeline. The entire procedure was created with efficiency, reproducibility, and compatibility with various modeling architectures in mind.

4.1 Text Cleaning and Normalization

The raw question strings exhibited heterogeneous formatting, including HTML artifacts, mathematical tags, escape sequences, and colloquial contractions. To standardize and clean the text, the following transformations were sequentially applied:

- **Unicode normalization**: All characters were normalized to NFKD form and encoded to ASCII. This addressed inconsistencies due to diacritics, ligatures, and symbol variants (e.g., “ﬁ” → “fi”).

- **HTML decoding**: Any residual HTML tags (e.g., `<br>`, `<i>`) and embedded MathJax syntax (`[math]...[/math]`) were stripped using regular expressions.

- **Escape sequence removal**: Common sequences such as `\n`, `\t`, and non-visible control characters were eliminated to produce uniform sentence boundaries.

- **Lowercasing and punctuation simplification**: After initial cleaning, all text was lowercase. Non-alphanumeric characters, except apostrophes and question marks (important in interrogative forms), were replaced with whitespace.

- **Tokenization and contraction handling**: Tokens were extracted using a rule-based regular expression that preserved apostrophes in words (e.g., "didn't" was preserved prior to stemming). Common English contractions were normalized using curated dictionaries (e.g., "can't" → "cannot").

4.2 Stopword Removal, Stemming, and Token Filtering

Each question was tokenized into words and further filtered using the following criteria:

- **Stopword removal**: Tokens corresponding to common English stop words were discarded using the NLTK corpus. This step removed high-frequency but low-discriminative terms such as "the", "is", "this".

- **Porter stemming**: Remaining tokens were passed through the Porter Stemmer to reduce inflectional variations (e.g., "running" → "run", "cars" → "car"). Stemming was memoized using an LRU cache to optimize batch performance.

- **Low-information token filtering**: Tokens consisting of digits, single characters, and other non-alphabetic units were excluded, along with strings deemed uninformative (e.g., isolated punctuation after filtering).

The resulting cleaned string represented a concise and content-rich form of the original input. In parallel, for each question, the system recorded the **total number of characters** and the **total number of words after cleaning.** These statistics were later used as structural features correlated with duplicate class tendencies.

4.3 Multi-Level Text Representations

To support diverse downstream tasks—ranging from lexical overlap features to sentence-level embeddings—the preprocessing pipeline produced and retained multiple synchronized versions of each question:

- **Raw**: The original input string was preserved for reference and traceability.
- **Cleaned lowercase**: The normalized and token-filtered version was used for embedding models (e.g., SBERT).
- **Lemmatized**: Although stemming was prioritized in the core cleaning function, lemmatized forms were retained in certain structural features (e.g., part-of-speech ratios, root forms).

This multi-granularity design ensured flexibility: syntactic features could leverage filtered tokens while semantic models could operate on context-preserving representations.

4.4 Embedding Preparation and Caching

Sentence embeddings were generated using two pretrained SBERT models **[1]**:

- `all-MiniLM-L6-v2` for **384-dimensional** embeddings **[1]**
- `all-mpnet-base-v2` for **768-dimensional** embeddings **[2]**

Aspect	**MiniLM-L6-H384-uncased**	**Microsoft/mpnet-base**
Model architecture	6 Transformer layers; hidden size = 384; 12 attention heads	12 Transformer layers; Hidden size = 768;12 attention heads
Fine-tuning objective	In-batch contrastive learning: cross-entropy over cosine similarities to separate true pairs	
Training data	> 1 billion sentence pairs from weighted mix of public corpora	
Compute environment	7 × TPU v3-8 under JAX/Flax with Google-advised optimizations	
Training schedule	100 000 steps; global batch size = 1 024 (128 per core)	
Sequence length	≤ 128 tokens (truncate beyond 256 word-pieces)	≤ 128 tokens (truncate beyond 384 word-pieces)
Optimizer & schedule	AdamW, lr = 2×10^{-5} with 500-step linear warmup	
Regularization	Dropout = 0.1 on attention and feed-forward layers	
Loss function	Cross-entropy over in-batch cosine similarities	
Intended application	Sentence/short-paragraph encoding for retrieval, clustering, and semantic similarity	

These were selected for their proven performance on sentence similarity benchmarks. All embeddings were cached on disk using a SHA-1 hash of the model name to avoid redundant computation across runs. Cleaned texts were processed in batches of 512 using GPU acceleration when available, and the resulting embedding matrices were saved to disk.

The final dataset consisted of over 537,000 unique questions across all splits, with each representation available in both 768-D and 384-D formats for model compatibility. All derived artifacts—cleaned questions, embeddings, and per-question metadata—were stored in a structured and versioned manner under the `data/processed/` directory.

5. Feature Engineering

A thorough set of features was built in order to reliably capture the lexical, semantic, and structural relationships between question pairs. In addition to surface-level similarities, these features were intended to capture deep latent patterns that differentiate semantically equivalent questions from non-duplicates. Several heterogeneous blocks made up the final feature matrix, which was subsequently dimensionality reduced to allow for tractable and effective modeling.

5.1 Lexical Features

Lexical comparisons were computed on normalized text forms to capture string-based similarities between question pairs. Three principal groups were constructed:

- **Length-based metrics**: Included absolute character counts for each question, as well as their absolute difference. To enhance granularity, the length difference was also bucketed into decile-based one-hot vectors.

- **Overlap-based similarities**: Employed Jaccard index, Sørensen–Dice coefficient, and token-level cosine similarity. These were calculated on filtered token sets, with stopwords removed to reduce noise **[3] [4]**.

- **TF-IDF projections and similarities**: Word-level and character-level TF-IDF vectorizers were trained with aggressive pruning (`min_df` thresholds and n-gram windows). Cosine similarities between vector representations were computed, and additional projections using Truncated SVD (150 components for word n-grams, 100 for character n-grams) were concatenated to form a compressed, informative lexical signature **[5]**.

5.2 Semantic Features

Semantic similarity was modeled using dense sentence embeddings and cross-encoder architectures:

- **Sentence-BERT embeddings**: Two pretrained models were used:
  - `all-mpnet-base-v2` for **768-dimensional** representations
  - `all-MiniLM-L6-v2` for **384-dimensional** representations

- For each question pair, a four-part vector was computed: the two embeddings `u` and `v`, their element-wise absolute difference `|u-v|`, and their element-wise product `u·v`. This yielded a 4×D-dimensional feature block **[6]**.

- **Cross-Encoder duplicate probabilities**: Probabilistic scores from a fine-tuned RoBERTa-based cross-encoder (`cross-encoder/quora-roberta-large`) were computed to capture interaction-level semantics. These served as an independent predictive signal, especially useful for difficult borderline cases **[7]**.

- **Dimensionality reduction of dense features**: To manage the curse of dimensionality **[8]** and improve generalization, two techniques were employed:
  - **Incremental PCA (IPCA)** was used in a two-pass chunked fashion to reduce the full feature set while retaining 95% of variance (auto-detected `k95`).
  - **UMAP [9]** was optionally applied after IPCA to project the data into lower-dimensional manifolds while preserving local geometric relationships. The final number of UMAP components was fixed (e.g., 10) to standardize downstream modeling inputs.

5.3 Structural and Statistical Features

Additional features were designed to encode structural patterns and statistical regularities within and between questions:

- **Fuzzy matching**: String similarity scores were computed using `RapidFuzz`, including both raw ratio and token-set-based ratio **[10]**.
- **Token statistics**: Metrics included word and character count differences, token alignment patterns, and presence of identical tokens in identical positions.
- **POS and syntactic resemblance**: Although not explicitly lemmatized in the feature matrix, the preprocessing stage enabled downstream part-of-speech (POS) ratio features and matched token categories.
- **Frequency-aware "magic" features**: The minimum global frequency of each question (i.e., how many times it appears in the corpus) was recorded and transformed using `log1p`, followed by quartile-based one-hot encoding. These features proved particularly useful in disambiguating question pairs involving highly generic versus niche queries.

The final feature vector for each question pair thus combined shallow string features, deep semantic embeddings, distributional projections, and handcrafted structural indicators. Depending on the SBERT dimensionality (`D` ∈ {384, 768}), the raw feature matrix had shape `(n_pairs, 526 + 4×D)`, which was then reduced via IPCA or UMAP as described.

All transformation artifacts, including trained vectorizers, dimensionality reducers, and raw feature arrays, were persisted to disk for reproducibility. This modular architecture supported multi-resolution experimentation across embedding granularities and dimensionality reduction strategies.

6. Expert Modeling and Ensemble Framework

A dual-path modeling approach that combined deep transformer-based architectures and traditional machine learning models was used to effectively tackle the Quora duplicate question detection task. These specialists offered complementary viewpoints across the question pair similarity space, despite their differences in structure and inductive bias. Eventually, a Mixture-of-Experts (MoE) architecture controlled by a learned gating mechanism was used to integrate their outputs.

6.1 Classical ML Experts on Reduced Feature Spaces

A suite of classical classifiers was trained on high-quality, low-dimensional representations obtained through feature engineering followed by dimensionality reduction. The process involved:

- **Input Features**: 3,600+ lexical, structural, statistical, and embedding-based features.
- **Dimensionality Reduction [11]**:
  - *Incremental PCA (IPCA)* on both 384D and 768D SBERT feature tracks.
  - *UMAP* with various target dimensions (e.g., 10, 20) for manifold-aware reduction.

The classifiers trained on these reduced vectors included:

- **Logistic Regression** with L2 regularization, optimized for probabilistic output.
- **Support Vector Machines** with linear kernels and probability calibration.
- **Random Forests**, configured with hundreds of estimators and tuned depth.

- **XGBoost [12]** and **LightGBM [13]**, leveraging gradient boosting with learning rate tuning.
- **k-Nearest Neighbors**, using Euclidean distance with dynamic neighborhood sizes.

All models were cross-validated using stratified folds, with F1-score as the primary selection metric.

6.2 Transformer-Based Experts

Pretrained transformer models fine-tuned on the Quora dataset were incorporated to capture contextual semantics:

- **Sequence Classification Experts**:
    - BERT-base (TextAttack QQP) **[14]**
    - RoBERTa-large (Howey QQP) **[15]**
    - XLNet-base (TextAttack QQP, optional based on environment)
- **Embedding-Based Expert**:
    - DistilBERT (ST: distilbert-base-nli-stsb-quora-ranking), whose 768D embeddings were projected to a 1,536D interaction space via concatenation of absolute differences and dot-products, then modeled via a logistic regression head **[16]**.
- **Cross-Encoder Expert**:
    - A RoBERTa-based cross-attention model that jointly encodes both questions, achieving higher fidelity at the cost of latency **[17]**.

All transformer models operated in frozen inference mode during gate training, ensuring the gating layer was solely responsible for adaptation.

7. Mixture-of-Experts (MoE) Gating Framework

To unify the outputs of heterogeneous experts, a lightweight trainable gating network was employed, assigning instance-specific weights to each expert's prediction **[18]**.

7.1 Gate Architecture

The gate is a feed-forward softmax-based layer that consumes a vector of expert probabilities for each question pair and produces a normalized weight distribution over them **[19]**. The final prediction is computed as a convex combination of expert outputs, ensuring both interpretability and differentiability.

Mathematically, for each instance:

$$\mathbf{p\_final = \Sigma \text{ (from } i=1 \text{ to } K) \; w\_i * p\_i}$$

where $\mathbf{p\_i}$ is the prediction from expert *i*, and $\mathbf{w\_i}$ is the corresponding learned weight.

7.2 Gate Training Protocol

To ensure fair comparison across expert combinations, all experts were frozen during gate training. The gate, implemented as a softmax-weighted layer over expert logits or probabilities, was trained using binary cross-entropy on validation predictions only. Expert outputs were precomputed on both training and validation sets, with the gate trained solely on the validation split to avoid label leakage.

An exhaustive search was performed over all non-empty subsets of the 10 available experts, resulting in **$2^{10} - 1$ = 1,023** unique expert combinations per hyperparameter setting.

7.3 Gate Variants and Hyperparameter Sweep

Each of the 1,023 expert subsets was evaluated under **three fixed training configurations**:

- **(lr = 0.001, epochs = 1)**
- **(lr = 0.01, epochs = 2)**
- **(lr = 0.05, epochs = 10)**

This resulted in a total of **3,069 gate models** trained and evaluated. For each configuration, gates were optimized using Adam with early stopping (patience = 5) and evaluated based on validation log-loss.

After this sweep, the **top 10 performing gates per training configuration** were selected, **retrained on the combined training and validation sets**, and **benchmarked on the held-out test set** using the full suite of metrics. These final results are presented in **Section 8: Results and Benchmarking**.

8. Results and Benchmarking

To assess the effectiveness of the Mixture-of-Experts (MoE) architecture and its constituent experts, an extensive benchmarking procedure was conducted using a held-out test set. Each evaluation captured a comprehensive suite of performance metrics, ensuring both predictive fidelity and practical viability were measured.

8.1 Evaluation Protocol

The evaluation was performed on final MoE models selected through a validation-based search across various expert subsets. For each configuration, a gate was trained with a specified learning rate and number of epochs while keeping all experts frozen. The following performance metrics were recorded:

- **Log-Loss**: Measures the calibration of predicted probabilities.
- **Accuracy**: Fraction of correctly classified duplicate/non-duplicate pairs.
- **F1 Score**: Harmonic mean of precision and recall, emphasizing class balance.
- **Precision**: True positives divided by total predicted positives.
- **Recall**: True positives divided by total actual positives.
- **AUC (ROC-AUC)**: Area under the receiver operating characteristic curve.
- **Inference Time**: Time in seconds required for forward-pass inference over the test set.

All evaluation runs were automatically logged using a lightweight CSV-based tracking system. Logs were versioned and stored under structured filenames, and all pipeline components were designed to be executable sequentially using a modular `main.py` entry point. This ensured full reproducibility and portability across environments.

8.2 Gate Benchmark Results

The results below summarize the performance of top 10 gate configurations under three training regimes:

Learning Rate = 0.001 and Epochs = 1

Model	Log-Loss	Accuracy	F1	Precision	Recall	ROC-AUC	Inference Time (s)
BERT+RoBERTa+CrossEnc	**0.047356**	0.9903	0.987	0.9834	0.9906	**0.9981**	203.74
RoBERTa+CrossEnc	0.050735	0.9903	0.987	0.9842	0.9898	0.9977	170.64
BERT+RoBERTa	0.052055	0.988	0.9838	**0.9853**	0.9824	0.9978	119.7
RoBERTa	0.057652	0.9901	0.9867	0.9838	0.9897	0.998	**86.8**
BERT+RoBERTa+LGBM	0.076406	0.9903	0.987	0.984	0.99	0.9973	204.44
RoBERTa+QD+CrossEnc	0.082726	0.9903	0.987	0.9839	0.99	0.9978	179.9
BERT+RoBERTa+CrossEnc+XGB	0.094231	0.9827	0.9771	0.9623	**0.9923**	0.9975	288.22
BERT+RoBERTa+CrossEnc+XGB+RF	0.101186	**0.9904**	**0.9872**	0.9844	0.99	0.9974	373.27
BERT+RoBERTa+CrossEnc+LGBM	0.110104	0.978	0.9708	0.9614	0.9804	0.997	302.37
RoBERTa+LGBM	0.220284	0.9895	0.9859	0.9818	0.99	0.994	171.3

Learning Rate = 0.01 and Epochs = 2

Model	Log-Loss	Accuracy	F1	Precision	Recall	ROC-AUC	Inference Time (s)
BERT+RoBERTa+CrossEnc	**0.038311**	0.9902	0.9868	0.9839	0.9898	**0.9982**	**215.27**
BERT+RoBERTa+CrossEnc+LR	0.039321	0.9902	0.9868	0.9839	0.9898	**0.9982**	305.51
BERT+RoBERTa	0.039342	0.9902	0.9869	0.9839	0.9898	0.9981	125.59
BERT+RoBERTa+QD+CrossEnc+LR+XGB+LGBM+SVM	0.039372	0.9902	0.9868	0.9839	0.9898	0.9980	550.71
BERT+RoBERTa+QD+CrossEnc+LR+LGBM	0.039423	0.9902	0.9868	0.9839	0.9898	0.9980	390.16
BERT+RoBERTa+QD+CrossEnc+LR+LGBM+SVM	0.039507	0.9902	0.9868	0.9839	0.9898	0.9980	485.51
BERT+RoBERTa+CrossEnc+SVM	0.039548	0.9902	0.9868	0.9839	0.9898	**0.9982**	303.16
BERT+RoBERTa+QD+CrossEnc+LR	0.039559	0.9902	0.9868	0.9839	0.9898	0.9980	303.98
BERT+RoBERTa+CrossEnc+XGB	0.040231	0.9902	0.9868	0.9839	0.9898	**0.9982**	304.48
BERT+RoBERTa+LGBM	0.040712	0.9902	0.9868	0.9839	0.9898	0.9980	215.66

Learning Rate = 0.05 and Epochs = 10

Model	Log-Loss	Accuracy	F1	Precision	Recall	ROC-AUC	Inference Time (s)
BERT+RoBERTa+QD+CrossEnc	**0.038053**	0.9901	0.9868	0.9838	0.9898	**0.9983**	**222.9**
BERT+RoBERTa+QD+CrossEnc+ XGB	0.038053	0.9901	0.9868	0.9838	0.9898	0.9982	311.8
BERT+RoBERTa+QD+CrossEnc+ SVM	0.038076	0.9901	0.9868	0.9838	0.9898	0.9982	313.18
BERT+RoBERTa+CrossEnc+LR+ XGB	0.038096	0.9901	0.9868	0.9838	0.9898	**0.9983**	391.47
BERT+RoBERTa+QD+CrossEnc+ XGB+LGBM	0.038168	**0.9902**	0.9868	**0.9839**	0.9898	**0.9983**	393.88
BERT+RoBERTa+CrossEnc+LR	0.038172	0.9901	0.9868	0.9838	0.9898	**0.9983**	303
BERT+RoBERTa+CrossEnc+LR+ SVM	0.038348	0.9901	0.9868	0.9838	0.9898	**0.9983**	393.45
BERT+RoBERTa+QD+CrossEnc+ XGB+SVM	0.038373	0.9901	0.9868	0.9838	0.9898	**0.9983**	394.94
BERT+RoBERTa+CrossEnc +SVM	0.038425	0.9901	0.9868	0.9838	0.9898	**0.9983**	301.79
BERT+RoBERTa+QD+CrossEnc+ LGBM+SVM	0.039055	**0.9902**	0.9869	0.9838	**0.9899**	0.9982	400.65

8.3 Observations

- **Consistency across metrics**: High-performing gates (especially those involving BERT, RoBERTa, and CrossEnc experts) consistently achieved log-loss values below 0.05 and AUCs above 0.998, indicating excellent probability calibration and discrimination.

- **Trade-offs**: While larger expert ensembles generally improved log-loss and AUC, they also incurred higher inference times, with some configurations needed around 500 seconds.

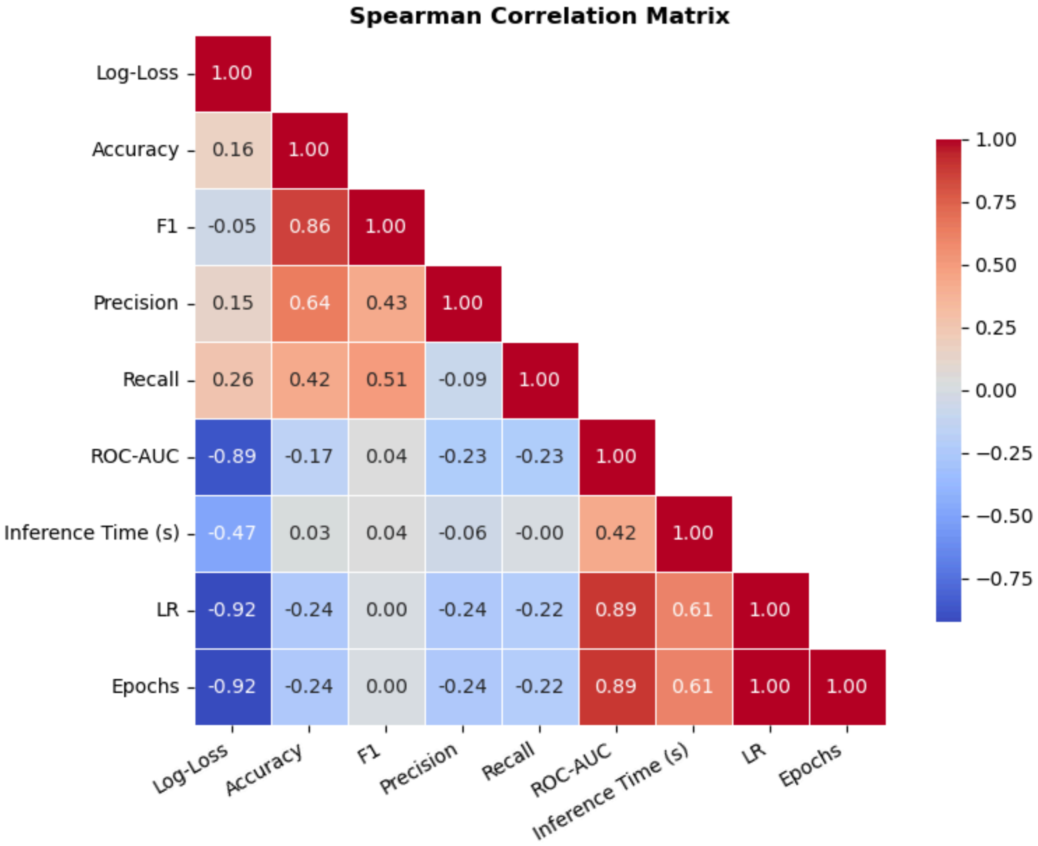
- **Minimal overfitting**: The close alignment between validation and test metrics suggests that the gate training procedure, including early stopping and validation-based selection, provided effective regularization.

- **Best tradeoff**: Configurations using **BERT+RoBERTa+CrossEnc** offered the best balance of performance and inference efficiency across all three training regimes.

These results confirm the hypothesis that combining diverse modeling paradigms through a learnable gating layer yields improved generalization and robust predictive performance on complex similarity tasks such as duplicate question detection.

8.4 Metric Correlation Analysis

The figure below shows the Spearman rank–order correlations between all numeric evaluation metrics (including learning rate and epoch count). We have masked the upper triangle (since it simply mirrors the lower), but retained the main diagonal (where each metric trivially correlates perfectly with itself).



Several key patterns emerge:

- **Log-Loss vs. ROC-AUC**: A strong negative correlation (≈ −0.89) indicates that configurations with lower log-loss also tend to achieve higher discriminative power (ROC-AUC).

- **F1, Precision & Recall**: F1 is highly positively correlated with both precision (≈ 0.43) and recall (≈ 0.51), reflecting its role as their harmonic mean.

- **Accuracy** correlates moderately with both F1 (≈ 0.86) and precision (≈ 0.64), suggesting that small gains in accuracy often coincide with improvements in class‑balanced performance.

- **Inference Time vs. Model Complexity**: Inference time shows a moderate positive correlation with both learning rate and number of epochs (≈ 0.61), implying that more extensive gate training (higher epochs) and larger ensembles (implicitly tied to lower LR) can increase evaluation latency.

- **Hyperparameter Effects**: Learning rate and epochs themselves correlate perfectly (1.00), since we only evaluated three paired settings, and both move together by design.

This analysis confirms that our choice of metrics captures complementary aspects of model behavior: improvements in probability calibration (log-loss) go hand‑in‑hand with ranking performance (ROC-AUC), while trade-offs between predictive quality and inference cost are clearly visible.

9. Conclusion

This study demonstrates that a modular Mixture-of-Experts (MoE) framework—comprising classical machine learning models and transformer-based experts integrated via a trainable gating mechanism—can achieve state-of-the-art performance on the Quora Question Pairs benchmark while preserving efficiency and interpretability.

Through an exhaustive evaluation of all 1,023 non-empty expert combinations across three fixed training configurations (learning rate = 0.001, 0.01, 0.05; epochs = 1, 2, 10 respectively), we trained a total of 3,069 gate variants. From each regime, the top 10 gates—selected based on validation log-loss—were retrained on the combined training and validation sets and benchmarked on a held-out test set.

The best-performing configurations achieved a **minimum test log-loss of 0.038** and **ROC-AUC scores exceeding 0.998**, evidencing both excellent probability calibration and strong discriminative capability. Notably, ensembles incorporating **BERT, RoBERTa, and CrossEncoder** consistently delivered the most favorable trade-offs between predictive performance and inference latency. Metric correlation analysis further validated that improvements in log-loss aligned with gains in ranking performance, while model complexity and training depth moderately influenced inference cost.

Overall, the proposed architecture confirms the practical viability of heterogeneous expert ensembling with soft, trainable gating. The pipeline's reproducibility, modularity, and performance scalability make it a compelling foundation for broader applications in semantic similarity, information retrieval, and question answering. Future directions may include adaptive expert selection, dynamic gating policies, and cross-domain transfer for enhanced generalization and efficiency.

References

[1] N. Reimers and I. Gurevych, "Sentence‑BERT: Sentence Embeddings using Siamese BERT‑Networks," in Proc. Conf. Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. Natural Language Processing (EMNLP‑IJCNLP), Hong Kong, Nov. 2019, pp. 3973–3983.

[2] Hugging Face, "sentence-transformers/all‑MiniLM‑L6‑v2," Huggingface.co. [Online]. Available: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2. Accessed: Jun. 29, 2025.

[3] Hugging Face, "sentence-transformers/all‑mpnet‑base‑v2," Huggingface.co. [Online]. Available: https://huggingface.co/sentence-transformers/all-mpnet-base-v2. Accessed: Jun. 29, 2025.

[4] S. Chawla, P. Aggarwal, and R. Kaur, "Comparative analysis of semantic similarity word embedding techniques for paraphrase detection," in _Emerging Technologies for Computing, Communication and Smart Cities: Proceedings of ETCCS 2021_, Singapore: Springer Nature Singapore, 2022, pp. 15–29.

[5] A. W. Qurashi, V. Holmes, and A. P. Johnson, "Document processing: Methods for semantic text similarity analysis," in _Proc. 2020 Int. Conf. Innovations in Intelligent Systems and Applications (INISTA)_, Aug. 2020, pp. 1–6.

[6] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "MPNet: Masked and Permuted Pre‑Training for Language Understanding," in Adv. Neural Inf. Process. Syst., vol. 33, 2020, pp. 16857–16867.

[7] M. Opacic and J. Salvatierra, “Seeing double: Detecting duplicate questions using Sentence‑BERT,” _Text Analysis and Retrieval 2024 Course Project Reports_, no. 41, 2024.

[8] J. G. Ortiz‑Barajas, G. Bel‑Enguix, and H. Gómez‑Adorno, “Sentence‑CROBI: A simple cross‑bi‑encoder‑based neural network architecture for paraphrase identification,” _Mathematics_, vol. 10, no. 19, p. 3578, 2022.

[9] L. McInnes, J. Healy, and J. Melville, “UMAP: Uniform manifold approximation and projection for dimension reduction,” arXiv:1802.03426, Feb. 2018.

[10] M. Liu, H. Zhang, Z. Xu, and K. Ding, “The fusion of fuzzy theories and natural language processing: A state‑of‑the‑art survey,” _Applied Soft Computing_, Art. no. 111818, 2024.

[11] V. Raunak, V. Gupta, and F. Metze, “Effective dimensionality reduction for word embeddings,” in _Proc. 4th Workshop on Representation Learning for NLP (RepL4NLP‑2019)_, Aug. 2019, pp. 235–243.

[12] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD), San Francisco, CA, Jul. 2016, pp. 785–794.

[13] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, “LightGBM: A highly efficient gradient boosting decision tree,” in Adv. Neural Inf. Process. Syst., vol. 30, 2017, pp. 3149–3157.

[14] TextAttack, “textattack/bert‑base‑uncased‑QQP,” Huggingface.co, May 20, 2021. [Online]. Available: https://huggingface.co/textattack/bert-base-uncased-QQP/tree/main. Accessed: Jun. 29, 2025.

[15] Howey, “roberta‑large‑qqp,” Huggingface.co, Jun. 4, 2021. [Online]. Available: https://huggingface.co/howey/roberta-large-qqp/tree/main. Accessed: Jun. 29, 2025.

[16] Sentence‑Transformers, “distilbert-base-nli-stsb-quora-ranking: config.json,” Huggingface.co, Jun. 22, 2021. [Online]. Available: https://huggingface.co/sentence-transformers/distilbert-base-nli-stsb-quora-ranking/blob/main/config.json. Accessed: Jun. 29, 2025.

[17] Cross‑Encoder, “quora-roberta-large,” Huggingface.co, Apr. 11, 2025. [Online]. Available: https://huggingface.co/cross-encoder/quora-roberta-large/tree/main. Accessed: Jun. 29, 2025.

[18] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, “Adaptive mixtures of local experts,” _Neural Computation_, vol. 3, no. 1, pp. 79–87, 1991.

[19] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, “Outrageously large neural networks: The sparsely-gated mixture-of-experts layer,” _arXiv preprint arXiv:1701.06538_, 2017.