Paraphrase Identification: A Multi-modal Evaluation on Classifying Duplicate Questions on Quora

1. INTRODUCTION (Predictive Task)

A question and answer platform, or Q&A platform, is an online platform that allows users to ask questions, share their knowledge, and brainstorm ideas with others. The past ten years have witnessed the rapid blossoming of forums like Quora and Stack Overflow, which have a quick growth of active users, user daily time spent, and market values. So far, Quora has 400 million monthly active users, 27 million daily active visitors on average, and 3000 to 5000 questions posted by users each day[5]. To keep Quora providing efficient user-to-user service, an essential maintenance job for each Q&A platform pops out – reducing duplicate questions. A large amount of duplicate questions dilutes the quality of information and hampers user experience by making it harder to locate unique and relevant answers. As Kornél Csernai, a Machine Learning Platform Engineer at Quora, pointed out, one of the product principles for Quora is to keep logically identical questions on the same page[4]. This principle helps the users search for their answers more efficiently, which means they do not have to go through different pages answering the same questions.

Considering the exponential growth of new questions and the large scale of existing questions, manually deleting and merging these duplicate questions is nearly impossible. Paraphrase identification, an application of Natural Language Processing (NPL), provides the platform with efficient strategies to identify these questions with appropriate models. However, this process is not easy. Zhou et al. proposed 24 different types of paraphrasing, e.g., Same-Polarity Substitutions, Opposite-Polarity Substitutions, Converse Substitutions, etc[9]. These types of paraphrasing ask the model to precisely identify the duplicate keywords among the questions and avoid overkilling since words may have different meanings under different contexts[6].

Despite all these difficulties, this paper presents our comprehensive study of applying state-of-the-art machine learning techniques to the Quora Question Pairs dataset. **We aim to explore various models,**

feature engineering strategies, and evaluation metrics to effectively classify question pairs as duplicates or non-duplicates. Our work contributes to the ongoing efforts in duplicate detection and offers insights into optimizing ML approaches for large-scale, real-world datasets.

2. DATASET STATISTICS (Dataset)

2.1. Basic Statistics

In 2017, Quora published its first dataset, *First Quora Dataset Release: Question Pairs*[4]. This dataset consists of 404351 question pairs labeled by 1 or 0, indicating whether the question pairs are duplicated. Because Quora's original sampling method returned a very imbalanced dataset with more positive samples, the engineers supplemented the dataset with negative samples. One source of negative samples is from "related questions" to make negative samples not too easy to detect. In detail, this dataset contains 255045 negative (non-duplicate) and 149306 positive (duplicate) instances(Fig. 1.). Each instance consists of the following features:

1. `id`: the id for each question pair
2. `qid1`: the id for question 1 in the pair
3. `qid2`: the id for question 2 in the pair
4. `question1`: the full text for question1
5. `question2`: the full text for question2
6. `is_deplicate`: 1(yes); 0(no)

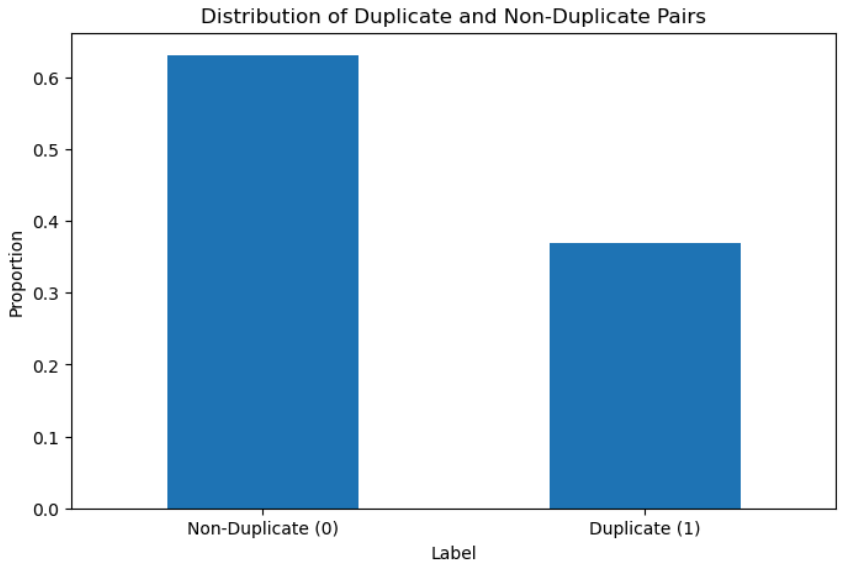

Fig. 1. Distribution of Duplicate and Non-Duplicate Pairs

Then, we started to look deeper into the statistical details of the dataset. The most basic but essential features to extract in PI are question pairs' length difference distribution and word overlap ratio. We first removed meaningless words and characters like punctuation and stop words to visualize the question pairs' length difference distribution. Then, we cast all words in lowercase to avoid different representations of proper nouns. Finally, we calculated the absolute difference between word counts of two questions to draw a histogram.

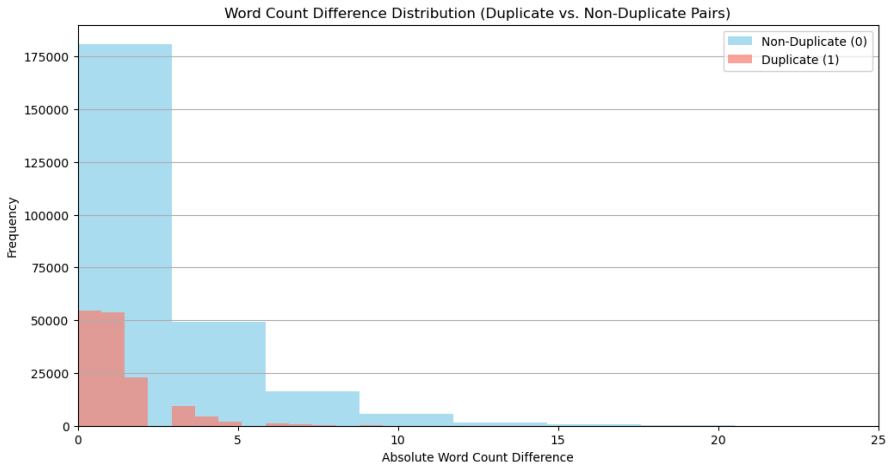

Fig. 2. Absolute Word Count Difference Distribution

It is evident in the figure (Fig. 2.) that **when two questions have an absolute word count over 5, it's very likely that the questions are non-duplicate**. It fits our intuition that when we see two questions have a significant difference in their length, we tend to think these two questions may not ask the same thing. However, based on the ratio of non-duplicate and duplicate question pairs, which is about 65%/35%, the frequency of non-duplicate is much inflated, and the figure can't accurately represent the actual word count different when we randomly sample the same amount of duplicate and non-duplicate samples in the test dataset.

We then shift to check if the word overlap ratio can demonstrate a more significant feature of the difference between non-duplicate and duplicate questions.

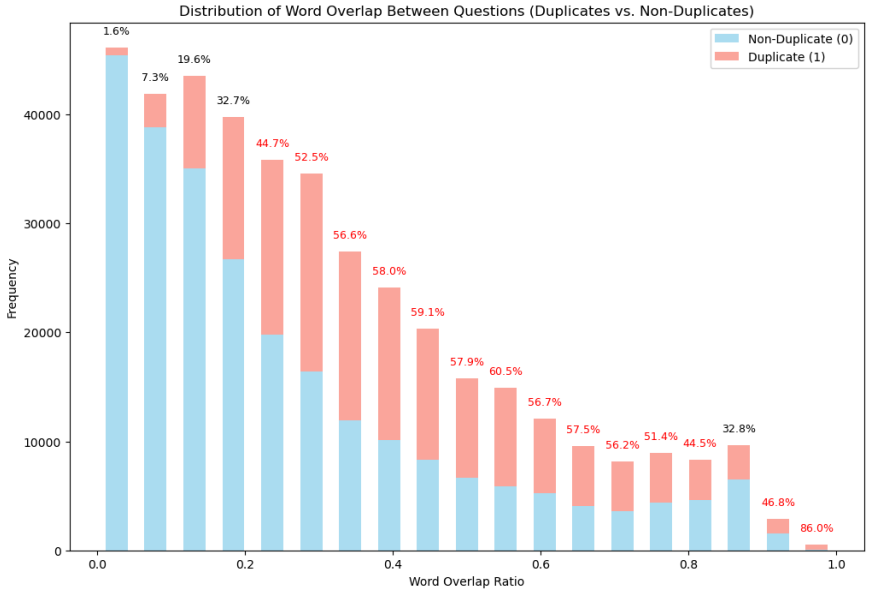

Fig. 3. Distribution of Word Overlap Between Questions

Because duplicate question pairs make up around 35% of the whole dataset, we label the bins red if the percentage of duplicate question pairs is higher than 35%. From the figure (Fig. 3.), we found that if the word overlap ratio is over 20%, the question pairs have a higher duplicate question expectation than the average expectation. However, not all words have the same weight when determining whether the

question pairs are duplicates. We find the top 10 most frequent words in duplicate and non-duplicate question pairs:

Duplicate (word, freq):

('best', 34465), ('get', 14954), ('india', 12735), ('quora', 12224), ('people', 10624), ('money', 8480), ('way', 8466), ('would', 8350), ('life', 8006), ('good', 7596)

Non-Duplicate (word, freq):

('best', 36038), ('get', 24560), ('like', 18736), ('good', 17190), ('india', 15976), ('people', 15441), ('would', 15242), ('one', 12766), ('make', 10917), ('much', 9656)

Here, we found a considerable number of words shared by both categories, indicating they have less weight in determining whether the question pairs are duplicates. Thus, we remove the top 20 most frequent words shared by both categories and bootstrap 100 samples to make a smooth KDE plot to show an accurate word-overlap relationship between duplicate and non-duplicate question pairs.

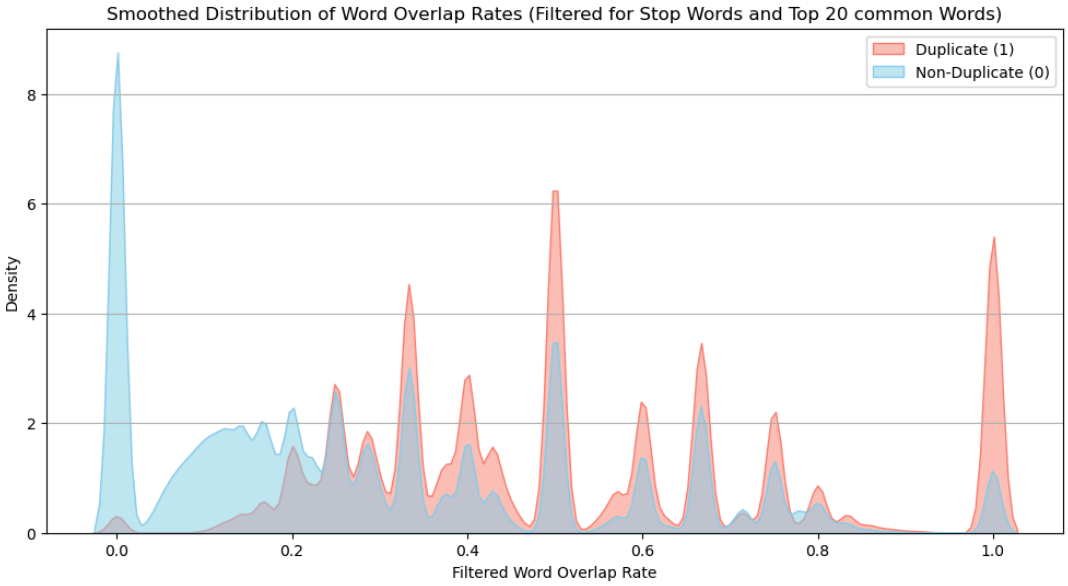

Fig. 4. KED smoothed Distribution of Word Overlap Rate

Compared to the previous figure, we see many peaks are much lower and normalized (Fig. 4.). It's evident that when the overlap ratio is smaller or equal to 0.2 percent, the question will likely be non-duplicate. In

contrast, when the overlap ratio is more significant than ~0.35%, the question pairs have a higher possibility of being duplicated.

2.2. Motivation

Even though both question pairs' length difference distribution and word overlap ratio seem to give a strong prediction baseline on this dataset, the prediction may be heavily biased when we test on Quora. This is because we assume our dataset distribution is similar to the actual distribution of questions asked on Quora. Based on the author's description [4], the sampling procedure involves some sanitization measures, such as removing the questions with extremely long question details because these questions have more hidden meaning in the detail than the question itself. Therefore, we need to utilize more advanced machine-learning models to eliminate the bias in this dataset and prevent overfitting[6].

3. PREVIOUS WORK (Literature)

3.1. The Development of Paraphrase Identification

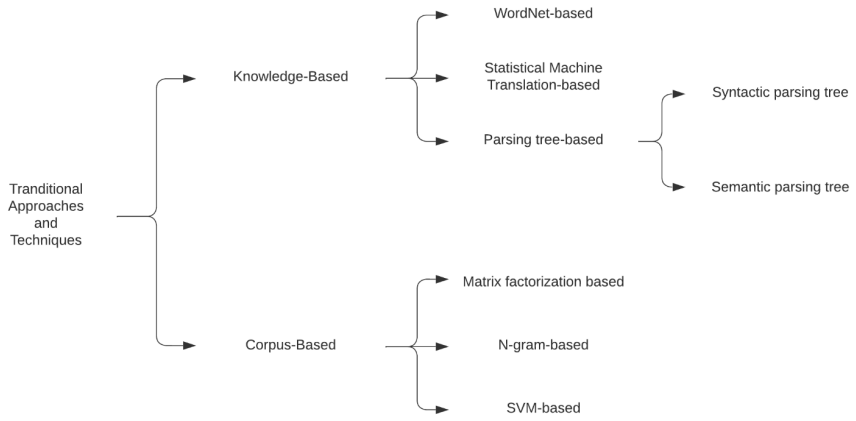

Fig. 5. Traditional Approaches and Techniques on Paraphrase Identification [9]

Through decades of engineers' hard work, Paraphrase Identification (PI) has experienced significant advancements, evolving from traditional, rule-based methodologies to more sophisticated, data-driven approaches that leverage large-scale textual data and advanced computational techniques (Fig. 5.). At its

early stage, PI mainly focused on the knowledge-based approach. This approach utilizes lexical databases such as WordNet to identify synonyms and semantic relationships to filter potential paraphrases by exploiting predefined semantic connections. However, considering the rapid emergence of new vocabulary, these resources are constrained by narrow lexical coverage and insufficient vocabulary diversity[9]. To overcome these constraints, the field progressively shifted towards corpus-based approaches.

TF-IDF (Term Frequency-Inverse Document Frequency) and Word2Vec are central to these corpus-based methods, both of which have become foundational tools in modern PI. TF-IDF transforms textual data into numerical feature vectors by quantifying the significance of terms. This approach is based on terms' frequency within a document relative to their frequency across the entire corpus[1]. This approach effectively highlights informative words and dampens the impact of common terms to enable proper similarity assessments using measures such as Cosine Similarity.

Meanwhile, Word2Vec changed the paradigm in PI with its dense, continuous vector representations of words, which capture contextual and semantic relationships through their distributional patterns over a large corpus. These types of word embeddings enable the aggregation of semantic knowledge both at the sentence and question levels, hence permitting models to detect more profound kinds of paraphrastic relations well beyond superficial lexical similarity. By encoding words into a semantic vector space, Word2Vec enhances the capability of PI models in detecting subtle semantic equivalences, increasing the accuracy and robustness of paraphrase detection.

The combination of TF-IDF and Word2Vec in corpus-centric frameworks has enhanced the effectiveness of PI models as well as their scalability and flexibility in different linguistic environments. These methods have further been combined with machine learning classifiers such as Support Vector Machines and Gradient Boosted Decision Trees (GBDT-LR, LGBM), which capitalize on their large-scale feature representations, enhancing efficacy in classification. In addition to TF-KLD-KNN and its variants, which

were more sophisticated metrics, other developments in matrix factorization methods, such as LSA, have also consolidated the gains of TF-IDF and Word2Vec in addressing the challenges of high dimensionality and data sparsity.

3.2. Previous Work on Current Dataset

The dataset (Quora Paired Question Dataset) has been used in many papers to validate their model. Two of the most frequently used models are BERT (Bidirectional Encoder Representations from Transformers) and MASS (Masked Sequence to Sequence).

Raffel et al. [7] reported their score on this dataset by using these two models in their paper *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.* First, they use a BERT-style[2] objective, using a masked language modeling (MLM) approach in which 15% of the tokens in a text block are modified, a mask token replaces 90%, and 10% by random tokens. This is an encoder-only approach, aiming at reconstructing the full, uncorrupted sequence, which benefits from deep bidirectional contextual understanding to excel in performance.

To this end, the authors introduce a MASS-style[8] objective that borrows from the work of Song et al. by extending the BERT-style approach by eliminating the element of random token swapping. In place of the corrupted spans, there are unique mask tokens, and only the corrupted tokens—not the whole sequence—are predicted by the trained encoder-decoder model. This modification not only eases the reconstruction process but also improves training efficiency by making the target sequences shorter. Their experimental results show that the MASS-style objective performs very similarly to the BERT-style approach. The detailed results are shown in the table:

	F1	Acc
BERT	88.47	91.44
MASS	88.58	91.44

Table 1. F1 and Acc Score of BERT and MASS Model on Current Dataset

4. METHODOLOGY (Predictive Task/Model)

4.1. Introduction to Cosine Similarity, LR, GBDT-LR, LGBM

4.1.1. Feature Extraction (TF-IDF, Word2Vec)

To numerically represent the textual data, we used two standard feature extraction techniques: Term Frequency-Inverse Document Frequency (TF-IDF) and Word2Vec. Each method captures different aspects of the text, enriching the feature space and resulting in better models' abilities to recognize semantic similarities between question pairs.

The baseline models provide a base reference point using more straightforward techniques to set preliminary performance baselines. These are Cosine Similarity and Logistic Regression, both trained on TF-IDF and Word2Vec feature vectors, giving four baseline configurations.

4.1.2. Introduction to Baseline Model – Cosine Similarity, LR

Cosine Similarity is defined as the cosine of the angle between two multi-dimensional vectors, indicating how well their directions are correlated. For each pair of questions, we calculated the cosine similarity over their TF-IDF or Word2Vec vectors. Based on validation results, we then used an empirically determined threshold to classify pairs as duplicates or non-duplicates. The cosine similarity is an intuitive and interpretable measure of similarity, making it a good baseline for comparison against more complex models.

Logistic Regression is a linear classification algorithm that estimates the probability of a binary response based on given input features. In our study, combined TF-IDF or Word2Vec vectors from pairs of

questions were used as input features. The model was trained on the training dataset, and the regularization parameters were tuned with cross-validation procedures to reduce overfitting.

4.1.3. Introduction to GBDT-LR, LGBM

Baseline models, such as Cosine Similarity and Logistic Regression, provide essential reference points and prove to be good at capturing linear associations and simple measures of similarity; however, they have limitations in handling more complex, non-linear interactions typical in natural language data. More specifically, these models may find it challenging to identify paraphrases accurately when the semantic relationships are complicated or when the interactions between features are non-linear. To address these challenges, we shifted our focus onto more advanced models using ensemble learning methodologies: Gradient Boosted Decision Trees with Logistic Regression (GBDT-LR) and Light Gradient Boosting Machine (LGBM).

Gradient Boosted Decision Trees—Logistic Regression (GBDT-LR) combines the strengths of Gradient-Boosted Decision Trees with Logistic Regression. GBDT excels at capturing non-linear patterns and interactions among features by building a sequence of decision trees, each focusing on the residuals of the previous trees. Output from GBDT is fed into a Logistic Regression classifier to refine probability estimates for classification. This hybrid approach leverages the hierarchical decision-making process of GBDT and the probabilistic output of Logistic Regression to improve overall classification accuracy.

Light Gradient Boosting Machine (LGBM) is an efficient gradient-boosting implementation optimized for speed and memory usage; hence, it is very suitable for large-scale datasets. LGBM uses a new tree-building algorithm, growing trees leaf-wise rather than level-wise, which can result in better accuracy with fewer iterations. Moreover, LGBM adopts several techniques, such as Gradient-based one-sided sampling (GOSS) and Exclusive Feature Bundling (EFB), to efficiently scale up to large feature spaces and reduce computational requirements. All these features make LGBM a potent tool for PI tasks and often outperform traditional gradient-boosting frameworks in both efficiency and accuracy. By integrating

these advanced tree-based models, we intended to harness the capability of modeling complex and non-linear relationships within a feature space, which overcomes the baseline models' limitations and significantly increases the classification performance.

4.2. Introduction to Deep Structured Semantic Model (DSSM)

4.2.1. Feature Extraction (Word Hasing)

In addition to the baseline and advanced tree-based models, we integrated the Deep Structured Semantic Model (DSSM), which uses deep neural network architectures for Paraphrase Identification. The DSSM architecture has two major components: a shared network and a classifier. Together, the two components form a double-tower structure that transforms every pair of input questions independently but symmetrically (Fig. 13.).

We began the development of DSSM by building a bag-of-words (BOW) representation for each input query, which we denote as the Term Vector. However, typical BOW representations have serious shortcomings: they cannot encode contextual and semantic subtleties of sentences and are poorly scalable with growing vocabulary. To mitigate these inefficiencies, we followed a word-hashing strategy with bag-of-CharTriGrams, where each word is decomposed into fixed-length character trigrams. For instance, the word "fruit" is transformed into the trigrams `[#fr, fru, rui, uit, it#]`. This way, the dimensionality of the feature space is reduced by maintaining a fixed size independent of vocabulary growth; hence, the scalability issues intrinsic to the BOW representations are somehow mitigated.

4.2.2. Model Implementation

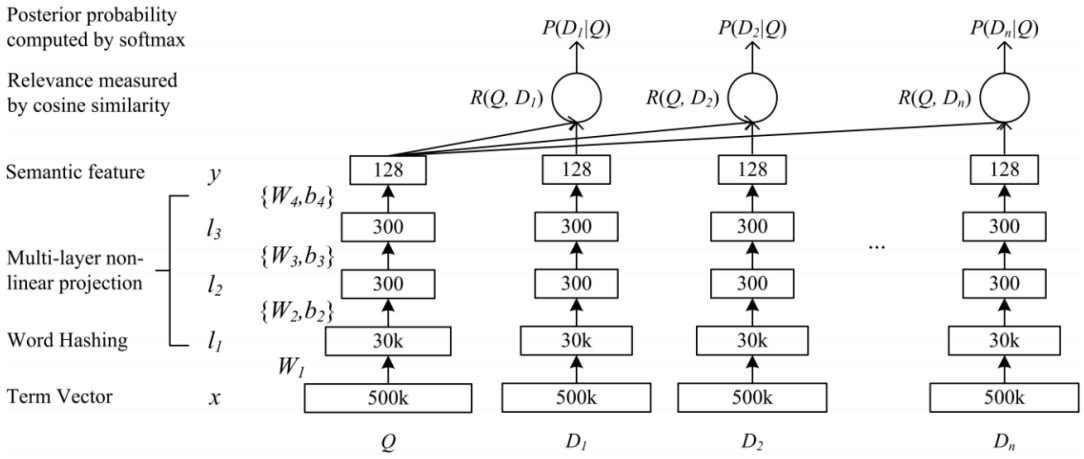

Fig. 6. Illustration of the DSSM[3]

The shared network should map each input vector into a meaningful latent representation. It consists of three consecutive linear (fully connected) layers, each followed by the Leaky ReLU activation function with a negative slope equal to 0.01. More precisely, the network architectural design is as follows:

1. First Linear Layer: Transforms the input vector of dimensionality `input_dim` to a hidden dimension `hidden_dim`, followed by Leaky ReLU activation.
2. Second Linear Layer: Maps the `hidden_dim` to another `hidden_dim`, maintaining the same activation function.
3. Third Linear Layer: Further processes the `hidden_dim` to `hidden_dim`, concluding with Leaky ReLU activation.

This configuration ensures that both input questions are processed identically, allowing for effective comparison by embedding them into the same semantic space where similar inputs are mapped to points that are close to each other. Proper initialization of weights is important for the DSSM's performance; hence, all linear layers use Xavier Uniform Initialization (`init.xavier_uniform_`) to preserve the variance of activations throughout the network, which promotes stable and efficient training. In addition,

the biases of linear layers are initialized to zero (`init.constant_(m.bias, 0)`) so that the initial activations depend solely on the input data and weight parameters.

The classifier combines the latent representations obtained from the two input questions to produce the final prediction. After obtaining latent vectors out1 and out2 from the shared network by using the forward_once method, the vectors are concatenated together along the feature dimension. The resulting vector has the shape of `hidden_dim * 2`. The concatenated vector goes through the classifier, which comprises two linear layers in the form:

1. First Linear Layer: Reduces the combined vector from `hidden_dim * 2` to 32 dimensions, followed by a ReLU activation function to introduce non-linearity.
2. Second Linear Layer: Maps the 32-dimensional vector to a single output neuron, passing through a Sigmoid activation function to produce a probability score between 0 and 1, indicating the likelihood of paraphrasing the two input questions.

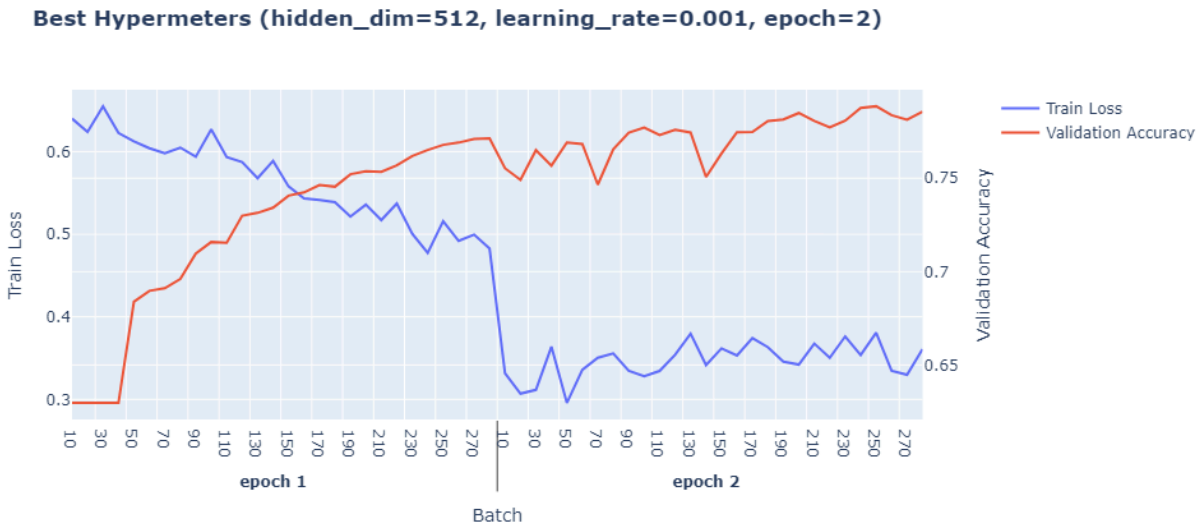

Fig. 7. DSSM Train Loss and Validation Accuracy Under Best Hyperparameter

The DSSM was trained using a judicious set of hyperparameters to perform well without overfitting. We did experiments for hidden_dim size in [256, 400, 512], learning rate in [0.01, 0.001], and training epoch

in [2, 5]. Empirical results show that the best-performing model uses a hidden dimension of 512, a learning rate of 0.001, and is trained for only 2 epochs.

4.2.3. Challenges and Model Tuning (Model)

During the development and tuning of the DSSM, several challenges arose that required careful adjustment of the model architecture and the training regimen (Fig. 9.). The main challenge was to avoid overfitting, which would happen when the model was trained with too many layers or for too many epochs. In particular, while adding depth to the shared network by adding more layers helped the model learn more complex representations initially, it ultimately led to overfitting, as shown by a sharp increase in validation loss even as the training loss continued to converge. Similarly, training the model for too many epochs, for example 5, caused the training loss to approach very small values, but the validation performance dropped severely, meaning the model had started memorizing the data rather than generalizing for unseen pairs.

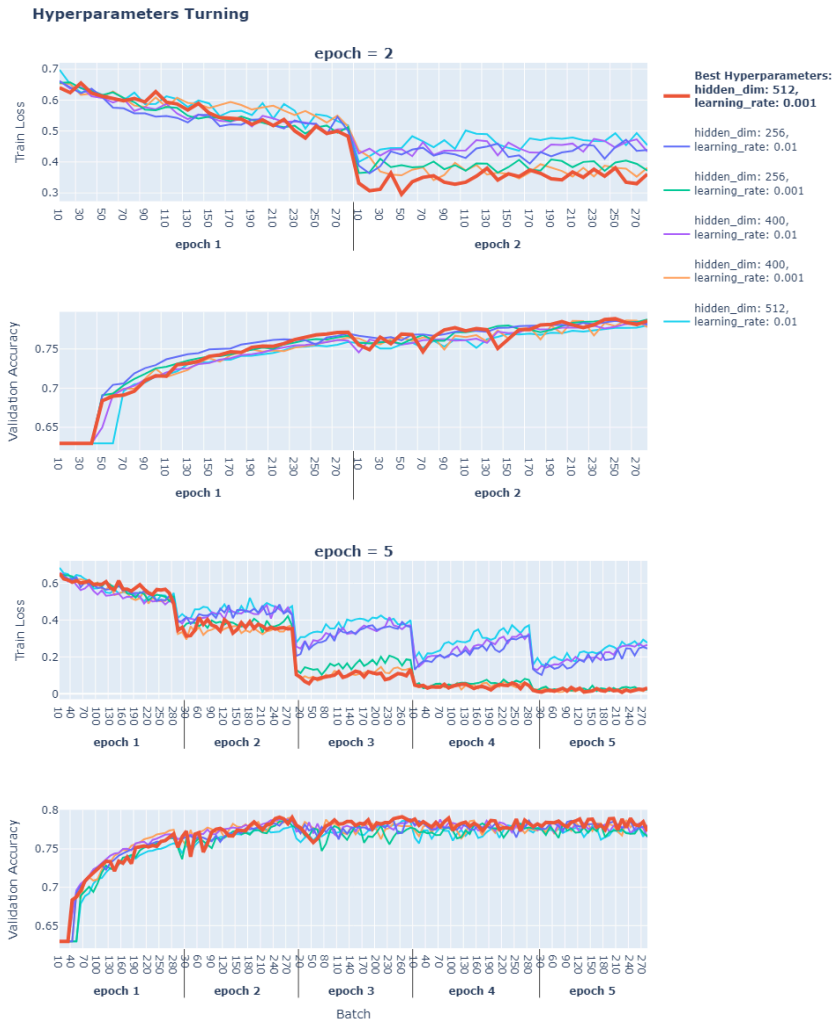

Fig. 8. Training Loss and Validation Accuracy of DSSM of all Hyperparameters with Best Highlighted

To handle these issues, we implemented the following strategies:

- Model Depth Reduction: We restricted the shared network to 3 layers to balance model's performance and generalization. The accuracy score of our validation dataset showed that deeper networks did not give proportional performance gains, making overfitting worse.
- Early Stopping and Epoch Limitation: This risk was quite well tackled by constraining the training to at most 2 epochs to allow good learning from the dataset without overfitting. Based on

this observation, it was established that increasing the number of epochs beyond two considerably increased validation loss.

- Word Hashing Optimization: Each question was initially represented by a word hashing layer with 30,000 trigrams. However, this resulted in high dimensionality, which caused overfitting and increased computational needs. To have a manageable feature space, the number of trigrams was reduced to 10,000, which retained the essential semantic information and made training time more efficient.
- Learning Rate Tuning: An appropriate learning rate was critical for stable and efficient training. Through experiment, we found that a learning rate of 0.001 yielded the best performance at a balance between convergence speed and model stability. Higher learning rates resulted in volatile training dynamics, whereas lower rates slowed down the convergence without much benefit.
- Weight Initialization: Using Xavier Uniform Initialization (init.xavier_uniform_) for all linear layers largely improved the model's performance. This type of initialization preserved the variance of activations across the layers, which in turn supported stable gradients and enabled effective training. Also, initializing biases to zero (init.constant_(m.bias, 0)) ensured initial activations depended only on input data and weights.

5. EVALUATION, RESULT, AND DISCUSSION (Result/Predictive Task/Literature)

5.1. Result

In this section, we evaluate the performance of our models by first comparing the accuracy scores between all models to give an intuitive idea. Then, we choose to use AUC to evaluate the models deeply under different thresholds. AUC evaluates the model’s performance across all possible classification thresholds, unlike accuracy, which depends on a fixed threshold. Also, AUC is less sensitive to imbalanced datasets, as it focuses on the ranking ability rather than exact predictions.

	TF-IDF	word2vec
Cosine similarity	**0.6541**	**0.6519**
Logistic regression	**0.7595**	**0.7211**
GBDT-LR	**0.7108**	**0.6955**
LGBM	**0.7504**	**0.7549**
DSSM	**0.7918**	

Table 2. Acc Table of All Models Implemented

The evaluation of our Paraphrase Identification models using the Quora Question Pairs dataset revealed significant differences in the performances of different feature representations and modeling methods, as shown in Table 2. As a baseline model, Cosine Similarity achieved accuracy scores of 0.6541 with TF-IDF and 0.6519 with Word2Vec. These results highlight the inability of cosine similarity to capture complex semantic associations accurately because it majorly computes geometric alignment without having a holistic, contextual understanding.

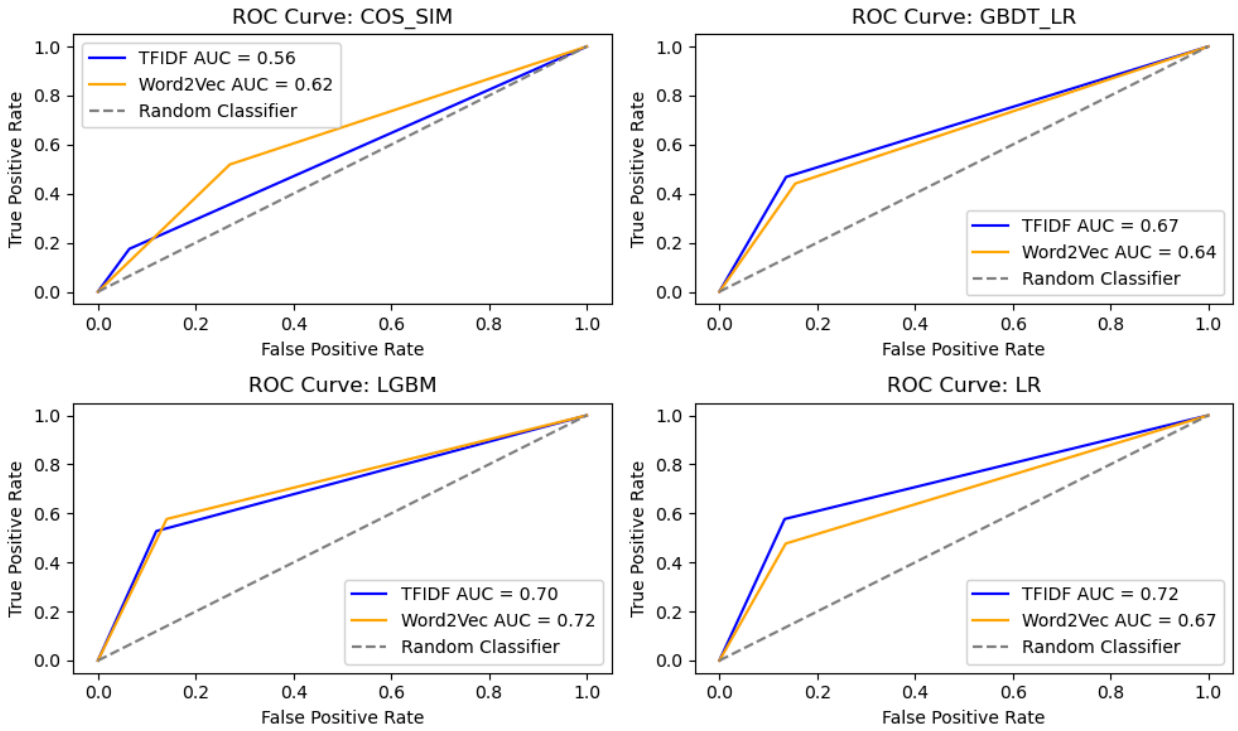

Fig. 9 ROC Curve of Baseline Models, GBDT_LR and LGBM

By contrast, Logistic Regression showed significant improvement with an accuracy of 0.7595 using TF-IDF and 0.7211 using Word2Vec. The better performance for TF-IDF indicates its effectiveness in providing discriminating features for linear classifiers. In contrast, the lower accuracy for Word2Vec shows that even though dense embeddings capture semantic nuances well, they do not align perfectly with the linear decision boundaries used by Logistic Regression.

Moving on to the more complex models, Gradient Boosted Decision Trees-Logistic Regression (GBDT-LR) achieved accuracies of 0.7108 using TF-IDF and 0.6955 with Word2Vec. Although it was expected that GBDT-LR would outperform Logistic Regression by modeling non-linear interactions in an effective way, the results even show a significant regression, which could be explained by the additional complexity introduced by Word2Vec embeddings.

Light Gradient Boosting Machine (LGBM) turned out to be a strong performer with accuracies of 0.7504 using TF-IDF and a striking 0.7549 using Word2Vec. In this respect, LGBM is flexible and strong in dealing with both sparse and dense feature representations in modeling complex semantic patterns, especially by profiting from rich information encoded in Word2Vec embeddings.

Comparing the AUC of all 4 models(Fig. 9.), we can see that Word2Vec performs better in Cosine Similarity and TF-IDF performs better in Logistic Regression. In contrast, there is no significant difference in the other two models.

The Deep Structured Semantic Model (DSSM) outperformed all baseline models with a high accuracy of 0.7918. This significantly improved performance can be attributed to the deep neural architecture of DSSM, which learns how to wrap and contrast the semantic complexity of question pairs through its double-tower structure and shared network. Hyperparameter tuning—careful constraining of the number of layers and epochs, in addition to adapting learning rates—greatly helped in enhancing the generalizability of DSSM to unseen test data and largely reduced overfitting for the model.

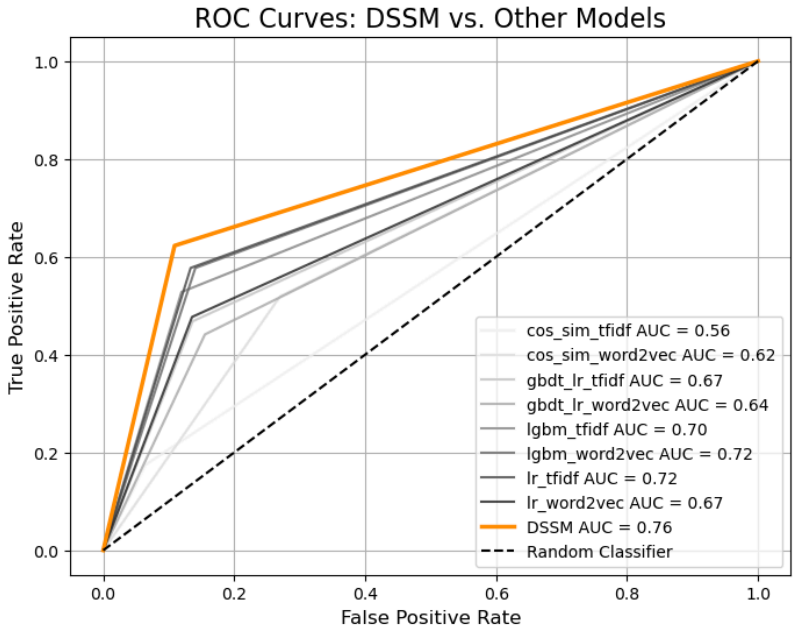

Fig. 10. Roc Curves: DSSM vs. Other Models

DSSM achieves an AUC (Area Under the ROC Curve) value of 0.76, indicating a 76% probability that the model will rank a randomly selected positive instance higher than a randomly selected negative instance (Fig. 10.). This suggests that the model performs better than random guessing (AUC = 0.5).

The True Positive Rate (TPR) at higher thresholds reflects the model's ability to capture 62% of actual positives, which is reasonable but leaves room for improvement. Meanwhile, the False Positive Rate (FPR) is 0.10, meaning that 10% of negative instances are incorrectly classified as positives. Although this FPR is relatively low, it could still pose challenges in scenarios where false positives are costly, such as fraud detection or spam filtering.

5.2. Comparison Between DSSM and Current State-of-the-Art Methods

Although our DSSM model achieves a good accuracy of 0.7918, it still lags behind the more advanced models like BART and MASS. This relative underachievement can be attributed to these models' differences in architecture and training methodologies. BART and MASS are strong transformer-based architectures which profit significantly from complete pre-training on large datasets to capture complex contextual and semantic nuances of the data. Their complex attention mechanisms and ability to represent sophisticated linguistic structures provide a notable advantage in understanding and generating subtle paraphrases. In contrast, the DSSM uses a shallower double-tower neural network architecture with less depth and much less pre-training than transformer models. As a result, DSSM is less able to capture the complex semantic relations and contextual dependencies required for high-precision Paraphrase Identification. Moreover, the lack of transfer learning in DSSM restricts its ability to generalize as strongly as BART and MASS, which have already learned general representations from large amounts of textual data.

Reference

[1] Aizawa, A. (2003). An information-theoretic perspective of tf–idf measures. Information Processing & Management, 39(1), 45-65.

[2] Devlin, J. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[3] Huang, P. S., He, X., Gao, J., Deng, L., Acero, A., & Heck, L. (2013, October). Learning deep structured semantic models for web search using clickthrough data. In Proceedings of the 22nd ACM international conference on Information & Knowledge Management (pp. 2333-2338).

[4] Iyer, S., Dandekar, N., & Csernai, K. (n.d.). *First Quora dataset release: Question pairs*. Quora. https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

[5] Kumar, N. (2024, October 9). *30 Quora Statistics 2024 (Users & revenue)*. DemandSage. https://www.demandsage.com/quora-statistics/

[6] Le, H. T., Cao, D. T., Bui, T. H., Luong, L. T., & Nguyen, H. Q. (2021). Improve quora question pair dataset for question similarity task. *2021 RIVF International Conference on Computing and Communication Technologies (RIVF)*, 1–5. https://doi.org/10.1109/rivf51545.2021.9642071 Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research, 21(140), 1-67.

[7] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research, 21(140), 1-67.

[8] Song, K. (2019). Mass: Masked sequence to sequence pre-training for language generation. arXiv preprint arXiv:1905.02450.

[9] Zhou, C., Qiu, C., Liang, L., & Acuna, D. E. (2022). Paraphrase identification with deep learning: A review of datasets and methods. arXiv preprint arXiv:2212.06933.