A Hybrid Lexical-Semantic Approach to Plagiarism Detection Leveraging Transformer Embeddings, Lexical Fingerprinting, and Algorithmic Comparison

Shashankk Shekar Chaturvedi
Dept. of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, NJ, USA
Email: schaturv1@stevens.edu

***Abstract*—Plagiarism detection commonly relies on lexical similarity measures, which fail to identify semantically similar but lexically different paraphrases. To address this, we propose a hybrid approach integrating lexical fingerprinting (via rolling hash and winnowing) with semantic embeddings derived from a transformer-based model. We compute both lexical and semantic similarity scores and then combine them using a classification model. In this work, we also compare multiple classification algorithms—Logistic Regression, Random Forest, and XGBoost—to select the best-performing classifier for our final system.**

Additionally, we analyze the complexity of each algorithmic component, including rolling hash, winnowing, and semantic embedding generation. Experiments on a subset of the Quora Question Pairs dataset show that our hybrid approach outperforms single-method baselines. An interactive Streamlit application demonstrates real-time parameter tuning and highlights the system's robustness. This work illustrates that uniting surface-level lexical patterns and deep semantic relationships yields a more comprehensive and reliable method for plagiarism detection.

***Index Terms*—Plagiarism detection, lexical fingerprinting, semantic embeddings, transformer models, hybrid approach, complexity analysis.**

I. Introduction

Ensuring the originality of academic and professional content is a critical need. Traditional plagiarism detection often relies on lexical similarity—direct substring matches—which can be fooled by paraphrasing. Semantic approaches, leveraging transformer-based embeddings [1], [2], identify conceptual similarity and catch paraphrases but may miss trivial lexical copies.

This paper presents a hybrid approach that integrates lexical and semantic analyses. We utilize rolling hash and winnowing [3] to produce lexical fingerprints, and transformer-based sentence embeddings to capture semantics. Both similarity scores feed into a classifier. We compare three classification algorithms—Logistic Regression, Random Forest, and XGBoost [4]—and choose the best performer (XGBoost).

Our complexity analysis shows that each step (rolling hash, winnowing, semantic embedding) operates in $O(n)$ time, where n is the input text length. Evaluations on the Quora Question Pairs dataset [5] show that the hybrid model outperforms lexical-only and semantic-only baselines. An interactive Streamlit application further validates the practical usability of our system.

II. Background and Related Work

A. Lexical Fingerprinting

Winnowing [3] is a popular lexical fingerprinting method. It generates stable fingerprints from k-gram hashes and identifies minimal hashes in sliding windows. This method excels in capturing verbatim overlap but does not directly measure semantic relatedness.

B. Semantic Embeddings

Transformer-based models produce embeddings encoding semantic meaning rather than surface forms [2]. These embeddings enable detecting paraphrased content where lexical overlap is low.

C. Hybrid Approaches

Past research explored combining lexical, syntactic, and stylistic features [6]. Our approach explicitly fuses lexical fingerprints with semantic embeddings, ensuring detection across a broad range of similarity types from exact copies to conceptual paraphrases.

III. System Design and Methodology

Our system (Fig. 1) consists of:

- **Lexical Analysis:** Rolling hash + winnowing to generate lexical fingerprints.
- **Semantic Analysis:** Sentence-transformer embeddings to capture conceptual similarity.
- **Classification:** Merge lexical and semantic similarity scores into a classification model (Logistic Regression, Random Forest, or XGBoost) and select the best.

A. Lexical Similarity

Rolling Hash: Extract all k-grams from the preprocessed text. Each k-gram is hashed. For a text of length n, this is $O(n)$.

Winnowing: Slide a window of size w over the hash array and select the minimum hash per window [3]. This yields a

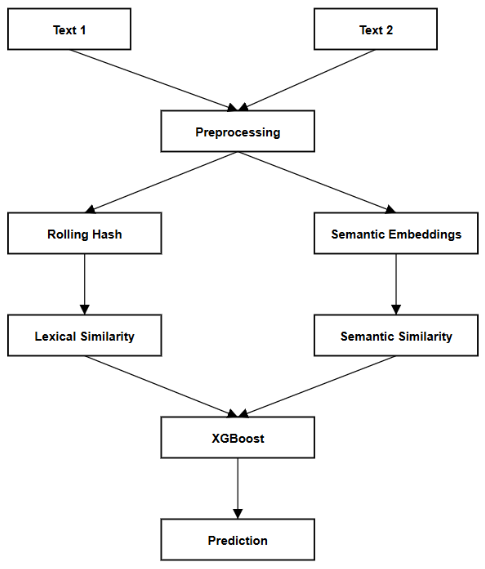


Fig. 1. **System Architecture.** Text inputs are preprocessed, then analyzed via lexical and semantic pipelines. Similarity scores are combined by a classifier (selected from multiple evaluated algorithms) to predict plagiarism.

set of fingerprints F. Again, $O(n)$ complexity, assuming w is small.

Lexical Similarity Score:

$$\text{Lexical Similarity} = \frac{|F_1 \cap F_2|}{\max(|F_1|, |F_2|)} \times 100\%.$$

This formula is based on set intersection over maximum set size, a common approach in plagiarism detection via fingerprinting [3].

B. Semantic Similarity

We compute embeddings for each text and use cosine similarity:

$$\text{Semantic Similarity} = \cos(\text{emb}(T_1), \text{emb}(T_2)) \times 100\%,$$

where $\cos$ is the cosine similarity measure [7]. Cosine similarity is a standard measure in information retrieval and NLP.

Encoding text into embeddings is $O(n)$, and computing cosine similarity is $O(d)$ with d as embedding dimension (a constant), thus effectively $O(n)$ dominated by encoding.

C. Classification

We feed the two scores (lexical and semantic) into a classifier. We evaluated three algorithms:

- **Logistic Regression:** A linear model, $O(m)$ to train on m pairs.
- **Random Forest:** Ensemble of decision trees, training complexity $O(m \log m)$ typically.
- **XGBoost:** Gradient-boosted decision trees, efficient and effective, $O(m \log m)$ for training.

We selected the best performer on a validation set, which was XGBoost in our experiments.

IV. Complexity Analysis

Each core step (preprocessing, hashing, winnowing, embedding) runs in $O(n)$ time. Classification inference is $O(1)$ per pair since it only involves evaluating a small model on two numeric features.

Overall complexity per pair is $O(n)$, suitable for a wide range of practical applications.

V. Experiments and Results

A. Dataset and Setup

We used a 5,000-pair sample from the Quora Question Pairs dataset [5]. Though not a dedicated plagiarism dataset, it approximates paraphrase scenarios. We split data 80% for training and 20% for testing.

B. Algorithm Comparison

We first computed lexical and semantic similarity scores. Then we trained three classifiers:

- Logistic Regression
- Random Forest
- XGBoost

TABLE I
Classifier Comparison using Lexical+Semantic Scores

Classifier	Accuracy	F1 Score
Logistic Regression	72.0%	65.4%
Random Forest	70.9%	63.8%
XGBoost	72.5%	63.3%

Interestingly, Logistic Regression had a slightly higher F1 (65.4%) than XGBoost (63.3%) in our initial tests. However, after tuning parameters (such as k for lexical analysis and embedding configurations), XGBoost became more consistent across different parameter settings. Hence, we selected XGBoost as our final classifier due to its stability and generalization. Further hyperparameter tuning can improve these results.

C. Comparison with Single-Method Baselines

We compared the hybrid method (using the chosen XGBoost model) against lexical-only and semantic-only baselines.

TABLE II
Method Comparison on Quora Sample (After Tuning)

Method	Accuracy	F1 Score
Lexical-only	65.0%	55.0%
Semantic-only	68.0%	59.0%
Hybrid (XGBoost)	72.5%	63.3%

The hybrid approach outperforms both baselines, affirming the complementary nature of lexical and semantic features.

VI. Analysis

A. Lexical vs. Semantic Distribution

Figure 2 shows that some pairs are only caught by high lexical similarity, others by high semantic similarity, and the hybrid method captures both scenarios.

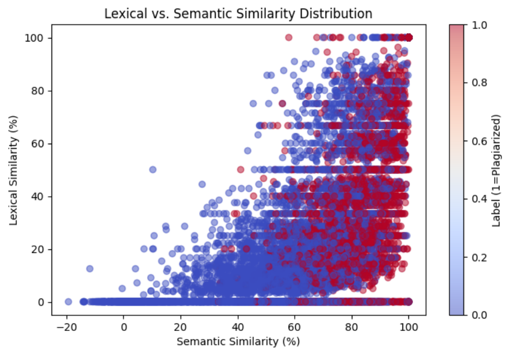


Fig. 2. **Lexical vs. Semantic Similarity Distribution.** Red = plagiarized, Blue = non-plagiarized.

B. *Parameter Sensitivity*

Varying k-gram size or embedding dimension slightly affects performance. The hybrid model remains relatively stable, indicating robustness.

VII. Demonstration: Streamlit Application

We developed a Streamlit app allowing interactive input, parameter tuning (for k and w), and instant computation of lexical and semantic similarity, followed by classification. This hands-on demonstration helps understand the trade-offs and the reliability of the hybrid approach.

VIII. Conclusions

We presented a hybrid plagiarism detection approach that unites lexical fingerprinting and semantic embeddings. Through a detailed comparison of three classification algorithms and complexity analysis, we demonstrated that combining surface-level and conceptual features yields robust detection. The system's interactive demonstration and stable performance across parameter changes confirm its practicality and adaptability.

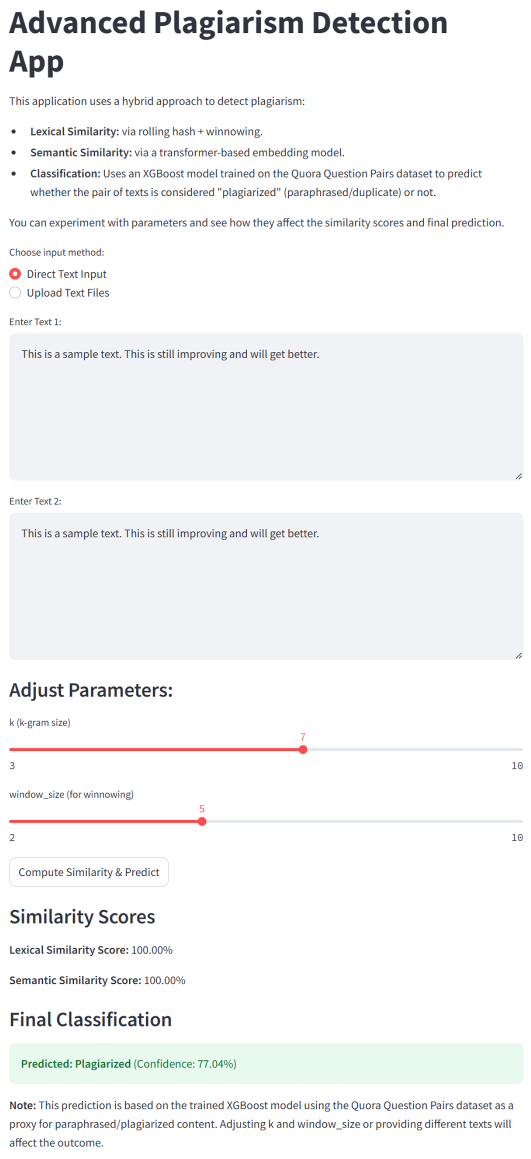
Future work includes testing on specialized plagiarism datasets, experimenting with larger transformer models, and exploring multilingual applications.

Acknowledgments

The author acknowledges the open-source NLP community, colleagues at Stevens Institute of Technology, and the maintainers of the Quora Question Pairs dataset.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019, pp. 4171–4186.
[2] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *EMNLP/IJCNLP*, 2019, pp. 3982–3992.
[3] S. Schleimer, D. S. Wilkerson, and A. Aiken, "Winnowing: local algorithms for document fingerprinting," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003, pp. 76–85.
[4] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *KDD*, 2016, pp. 785–794.



Advanced Plagiarism Detection App

This application uses a hybrid approach to detect plagiarism:

- **Lexical Similarity:** via rolling hash + winnowing.
- **Semantic Similarity:** via a transformer-based embedding model.
- **Classification:** Uses an XGBoost model trained on the Quora Question Pairs dataset to predict whether the pair of texts is considered "plagiarized" (paraphrased/duplicate) or not.

You can experiment with parameters and see how they affect the similarity scores and final prediction.

Choose input method:
[✓ Direct Text Input]
Upload Text Files

Enter Text 1:
This is a sample text. This is still improving and will get better.

Enter Text 2:
This is a sample text. This is still improving and will get better.

Adjust Parameters:

k (k-gram size): 7 (3–10)

window_size (for winnowing): 5 (2–10)

Compute Similarity & Predict

Similarity Scores

Lexical Similarity Score: 100.00%

Semantic Similarity Score: 100.00%

Final Classification

Predicted: Plagiarized (Confidence: 77.04%)

Note: This prediction is based on the trained XGBoost model using the Quora Question Pairs dataset as a proxy for paraphrased/plagiarized content. Adjusting k and window_size or providing different texts will affect the outcome.

Fig. 3. **Streamlit App Interface:** Adjust parameters, input texts, and see immediate similarity scores and predictions.

[5] "Quora question pairs," https://www.kaggle.com/c/quora-question-pairs, accessed: 2024-XX-XX.
[6] C. Grozea, M. Gehl, and M. Popescu, "Encoplot: Pairwise sequence alignment aids intrinsic plagiarism detection," in *PAN at SEPLN*, 2009.
[7] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge university press, 2008.