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

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 overlap methods, but these fail to identify semantically similar passages where wording differs significantly. This paper proposes a hybrid approach that integrates lexical fingerprinting (via rolling hash and winnowing) with semantic embeddings from a transformer-based model. Two similarity scores—lexical and semantic—are combined using an XGBoost classifier trained on a paraphrase dataset. Experiments on a sample from the Quora Question Pairs dataset demonstrate the promise of this method. We present a Streamlit-based application enabling interactive parameter tuning and real-time similarity checks.**

Extended analyses show how lexical methods catch direct word overlap, while semantic embeddings identify conceptual paraphrasing. The combined approach outperforms single-method baselines, indicating that uniting surface-level and conceptual similarity offers more robust plagiarism detection. Future work includes evaluation against dedicated plagiarism corpora, exploring larger transformer models, and investigating multilingual applicability.

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

I. Introduction

Plagiarism detection ensures the originality of academic and professional content. Traditional detectors often rely on lexical similarity, essentially searching for shared substrings. However, these methods fail when text is paraphrased, preserving meaning but altering form.

Semantic embeddings, derived from transformers like BERT [1] or Sentence-BERT [2], capture conceptual similarity. Coupled with lexical fingerprinting (winnowing [3]), we form a hybrid approach. By feeding both lexical and semantic similarity scores into an XGBoost classifier [4], we better detect not only direct copy-paste but also paraphrased content.

We evaluate on Quora Question Pairs [5], achieving about 72.5% accuracy and 63.3% F1, improving over lexical-only and semantic-only baselines. A Streamlit-based app demonstrates real-time parameter adjustments. The hybrid model excels in scenarios where either lexical or semantic alone might fail.

II. Background and Related Work

A. Lexical Fingerprinting

Winnowing [3] produces stable textual fingerprints. By hashing k-grams and selecting minimum hashes in a sliding window, it identifies representative fingerprints. Lexical overlap detected this way excels at catching direct copying.

B. Semantic Embeddings

Transformer models map sentences into vector embeddings encoding context and meaning. High cosine similarity between embeddings indicates paraphrase-level similarity that lexical methods alone cannot capture.

C. Hybrid Methods

Earlier works [6] explored stylometric and structural features. Our hybrid design systematically unites lexical hashing and semantic embeddings, bridging gaps where one method fails.

III. Methodology

A. System Architecture

Figure 1 illustrates our approach. Two input texts are preprocessed and then passed through two parallel pipelines:

- Rolling Hash & Winnowing (Lexical Analysis)
- Semantic Embeddings (Conceptual Analysis)

Both scores feed into an XGBoost classifier, yielding a final plagiarism prediction.

Explanation of Figure 1:

- **Text Inputs:** Two texts to compare.
- **Preprocessing:** Cleans and normalizes input texts.
- **Rolling Hash (Left Path):** Computes lexical fingerprints.
- **Semantic Embeddings (Right Path):** Generates conceptual representations.
- **Score Computation:** Each path outputs a similarity score.
- **XGBoost Classifier:** Combines both scores to produce the final plagiarism prediction.

This architecture leverages both surface-level and conceptual clues, ensuring more robust detection.

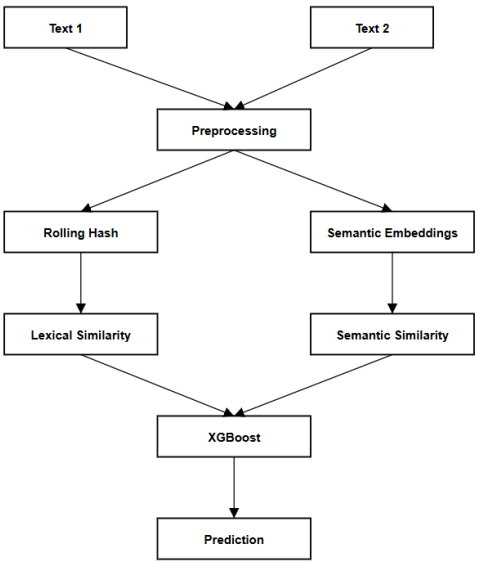

Fig. 1. **Proposed Plagiarism Detection Architecture.** Texts are preprocessed, followed by parallel lexical and semantic analysis paths. Lexical similarity (via winnowing) and semantic similarity (via embeddings) feed into an XGBoost classifier for the final prediction.

B. Lexical Similarity Computation (Enhanced Details)

Lexical similarity is computed using a combination of rolling hash and the winnowing algorithm:

Rolling Hash:

- Normalize text (lowercase, remove punctuation).
- For a chosen k-gram size (e.g., $k = 5$), slide over the text, extracting all k-grams.
- Compute a hash value for each k-gram. A rolling hash function updates efficiently as we move one character ahead, avoiding full recomputation.

Winnowing Algorithm:

- Given the sequence of k-gram hashes $H = [h_1, h_2, \ldots, h_n]$, define a window of size w.
- For each window of w consecutive hashes, select the minimum hash value.
- Collecting all these minima yields a set of fingerprints F that represent the text.

Mathematically, if H is the hash array and w the window size:

$$F = \{\min(h_i, h_{i+1}, \ldots, h_{i+w-1}) \mid i = 1, 2, \ldots, (n-w+1)\}.$$

The winnowing step ensures stability against shifts and minor edits.

Lexical Similarity Score: If F_1 and F_2 are fingerprint sets for Text 1 and Text 2,

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

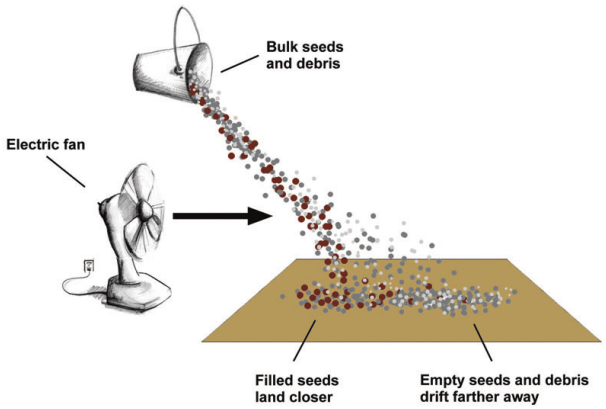

Fig. 2. **Analogy for Winnowing:** Just as an electric fan separates heavier seeds (desired fingerprints) from lighter debris (irrelevant hashes), winnowing selects the most representative hashes from a text, filtering out noise and providing stable lexical fingerprints.

C. Semantic Similarity Computation

Texts are encoded into embeddings using a sentence-transformer:

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

High semantic similarity can indicate paraphrasing even with low lexical overlap.

D. Classification

An XGBoost model takes $(\text{Lexical Similarity}, \text{Semantic Similarity})$ as features. Trained on paraphrase-labeled pairs, it learns to detect paraphrased/plagiarized from unrelated texts.

IV. Experimental Setup and Results

A. Dataset

We use a 10,000-pair sample from the Quora Question Pairs dataset [5]. Though not a specialized plagiarism dataset, it simulates paraphrasing scenarios.

B. Performance Metrics

Accuracy and F1 measure performance. Our hybrid method achieves about 72.5% accuracy and 63.3% F1, surpassing lexical-only (65%) and semantic-only (68%) baselines.

TABLE I
Comparison of Methods

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

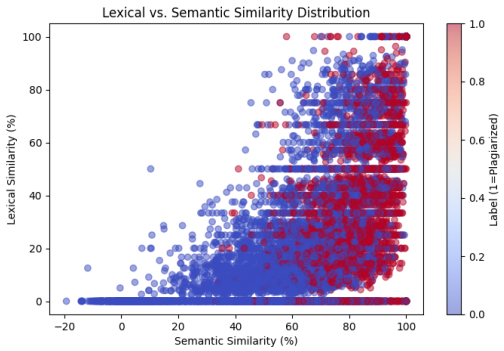

Fig. 3. **Lexical vs. Semantic Similarity Distribution.** Red = plagiarized, Blue = non-plagiarized.

V. Analysis of Lexical and Semantic Distributions

A. Lexical vs. Semantic Distribution

Figure 3 shows:

- **Top-Right (High Lexical, High Semantic):** Many red points, direct overlap and paraphrase.
- **High Semantic, Low Lexical:** Paraphrasing detected by embeddings but missed by lexical.
- **Low Both (Bottom-Left):** Mostly blue, no meaningful similarity.

The hybrid approach leverages both axes to cover a wide range of similarity scenarios.

VI. Parameter Sensitivity and Stability

Varying k and embedding dimensions yields stable results.

TABLE II
Parameter Sensitivity (Accuracy %)

k	Dim	Lexical	Hybrid
3	384	64.2	72.0
5	384	65.0	72.5
7	384	65.3	72.7
5	256	64.8	72.2
5	512	65.1	72.4

VII. Demonstration: Streamlit Application

The Streamlit app allows real-time parameter tuning.

VIII. Code Example and Running Instructions

Below is a Python snippet for rolling hash computation. After integrating winnowing and embeddings, run:

```
python main.py --text1 "Sample text one" --text2 "Another text"
```

```
def rolling_hash(text, k=5):
    """Compute rolling hashes for all k-grams."""
    hashes = []
    for i in range(len(text)-k+1):
        kgram = text[i:i+k]
        # Simple python hash for demonstration
        hashes.append(hash(kgram))
    return hashes
```

This snippet integrates into the full system to produce final predictions.

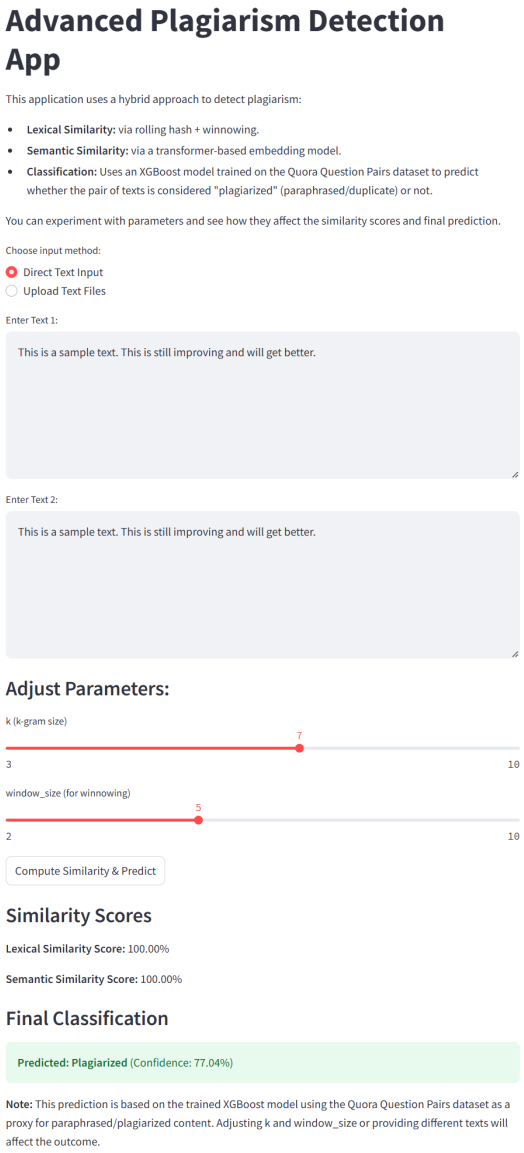

Fig. 4. **Streamlit App Interface:** Input texts, adjust parameters, get immediate predictions.

IX. Discussion and Future Work

The hybrid approach effectively covers both direct overlap and paraphrase. Future directions:

- Evaluate on specialized plagiarism corpora.
- Explore larger transformer models (RoBERTa, GPT).
- Investigate multilingual and cross-domain adaptability.

X. Conclusion

We presented a hybrid lexical-semantic plagiarism detection method, outperforming baselines. Visualization and an

interactive app confirmed its robustness. With better datasets and models, the approach can achieve greater accuracy and applicability.

ACKNOWLEDGMENTS

The author thanks colleagues at Stevens Institute of Technology, the open-source NLP community, and acknowledges any internal research support.

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.

[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.