Hierarchical Generative Plagiarism Detection Method

Notebook for PAN at CLEF 2025

Zongbao Su, Yong Han*, Yihao Jia and Leilei Kong

Foshan University, Foshan, China

Abstract

Generative Plagiarism Detection is a challenging task that requires systems not only to identify literal reuse but also to detect semantically similar text segments with different surface expressions. In the Generative Plagiarism Detection task at PAN@CLEF 2025, we propose a hierarchical detection approach. Our method integrates multiple embedding models and semantic similarity evaluation mechanisms to effectively identify complex paraphrased content. Specifically, we employ Sentence-BERT, MPNet, and TF-IDF to perform sentence-level vectorization of both suspicious and source documents, independently generating candidate pairs based on similarity scores. These candidate sets are then merged through a multi-strategy fusion mechanism. Furthermore, a fine-tuned BERT model is used to verify semantic similarity, enhancing the system's ability to detect generative paraphrasing. The final system outputs aligned text segments with high confidence. Experimental results demonstrate that our hierarchical matching strategy exhibits robustness and generalization across multiple evaluation metrics.

Keywords

Generative Plagiarism Detection, large language model, Hierarchical

1. Introduction

Plagiarism detection, or more broadly, text reuse detection, has long been a critical research area in natural language processing and information retrieval. With the rapid progress of large language models (LLMs) and the widespread adoption of generative AI, a new and more challenging form of plagiarism—generated plagiarism—has emerged. This type of plagiarism often preserves semantic meaning while rephrasing the original content, making it difficult for traditional surface-level similarity methods to identify such rewritten segments [1, 2]. Moreover, the proliferation of social media, Q&A forums, and academic writing assistance tools has further increased the risk of large-scale text reuse generated by machines [2].

The PAN (Plagiarism, Authorship, and Near-Duplicate Detection) shared task series has played a leading role in advancing research in this domain. In recent years, PAN has significantly increased task difficulty—from machine-translated plagiarism and synonym substitution to now confronting systems with generated plagiarism detection challenges (PAN 2025). The current task requires systems to identify aligned text fragments rewritten and reused from source documents using large language models [3, 4].

To this end, we propose a hierarchical detection strategy that integrates multiple semantic representations and verification mechanisms for the PAN@CLEF 2025 Generated Plagiarism Detection task. The motivation behind this design is to improve candidate fragment coverage by combining various similarity evaluation methods, while using a semantic verification model to filter out false matches, thereby achieving a better balance between precision and recall.

[1]Source code available at https://github.com/CCheZi/Generative-Plagiarism-Detection

CLEF 2025 Working Notes, 9 – 12 September 2025, Madrid, Spain

*Corresponding author.

2199845212@qq.com (Z. Su); hanyong2005@fosu.edu.cn (Y. Han); 2310743981@qq.com (Y. Jia); kongleilei@fosu.edu.com (L. Kong)

0009-0007-8512-6049 (Z. Su); 0000-0001-7832-1662 (Y. Han); 0009-0003-7352-4489 (Y. Jia); 0000-0002-4636-3507 (L. Kong)

2. Related Works

Plagiarism detection research has evolved through multi-stage frameworks and adaptive strategies to address diverse obfuscation techniques. We referred to some papers from the 2014 CLEF competition and compared them, as shown in Table 1.Table 1 summarizes five prominent methods from that competition [5, 6, 7, 8, 9].Finally, a hierarchical generative plagiarism detection method was proposed.

Table 1
Comparison of plagiarism detection methods

Method	Core Technique	Adaptive Strategy
TF-ISF + Recursive Extension (PAN 2014 Winning Approach)	• TF-ISF weighting to retain stopwords while reducing false positives • Recursive algorithm to extend seed matches into maximal passages with gap tolerance • Overlap filtering via quality function and minimum length threshold	Switches to variant B (optimized for summary obfuscation) when suspicious document length is significantly shorter than source document
Multi-Type N-Gram + Elliptical Clustering	• Fusion of regular n-grams, stopword n-grams, named entity n-grams, and context-aware features • Noise-sensitive elliptical clustering for feature aggregation • VSM cosine similarity for result verification	Dynamically selects from 4 preset strategies based on global noise level and cluster characteristics
Hybrid Architecture (Alignment + Clustering)	• Text alignment via Smith-Waterman algorithm for ordered plagiarism • Clustering with Jaccard coefficient for non-ordered cases (e.g., summaries) • Tiered content word thresholds for precision-recall balance	Clustering activated when alignment fails, with adaptive parameters for different obfuscation types
TER-p + Bigram N-Gram Dual Strategy	• TER-p (machine translation metric) for strict sentence-level matching • Bigram n-gram for fragmented plagiarism detection • Result merging with 80-character gap threshold	Balances precision (TER-p ≥ 0.9) and recall (bigram n-gram) for different obfuscation levels
CoReMo 2.3 Self-Tuning Model	• Extended Contextual N-grams (XCTnG) allowing skip words for word order adjustment • Dynamic parameter adjustment based on suspicious/source document length ratio	3-stage rules (e.g., 8% filtering distance for susp/src < 1.6) for cross-corpus adaptability

The proposed “hierarchical generative plagiarism detection method” sets up triple similarity filtering and BERT verification by referencing and analyzing the above methods, so as to carry out more rigorous and accurate plagiarism detection.

3. Method

Our system adopts a multi-stage processing pipeline that integrates various state-of-the-art natural language processing techniques and similarity computation methods. It consists of four main stages: text preprocessing and sentence segmentation, multi-model vector representation, hierarchical similarity matching with block merging, and result output. The following sections provide a detailed description of the entire processing workflow.

3.1. Text Preprocessing and Sentence Segmentation

Given a pair of input documents (a suspicious document and a source document), we first perform text preprocessing and sentence segmentation:

1. Use the English language model `en_core_web_sm` from the spaCy library (version 3.8.5) for accurate sentence segmentation [10].
2. Record character-level offsets (start position and length) for each sentence.
3. Normalize sentence text (e.g., strip leading/trailing whitespace).

The output of the preprocessing stage includes:

1. A list of sentences: textual content of all sentences in the document.
2. A list of offsets: positional information (start offset, length) of each sentence in the original text.
3. Original raw text: retained for precise offset calculation in later stages.

3.2. Multi-Model Vector Representation

To comprehensively capture the semantic features of text, we employ three different vectorization methods:

1. **Sentence-BERT encoding**
   (a) Encode sentences into semantic vectors using a pre-trained Sentence-BERT model [11] from HuggingFace (`all-MiniLM-L6-v2`).
   (b) Generate 384-dimensional dense vector representations.
   (c) Build efficient vector indices using the FAISS library (version 1.9.0) [12].
2. **MPNet encoding**
   (a) Use a pre-trained MPNet model [13] from HuggingFace (`all-mpnet-base-v2`) to obtain alternative semantic embeddings.
   (b) Serve as a complement to Sentence-BERT, offering a different semantic perspective.
   (c) Also indexed with FAISS for fast similarity search.
3. **TF-IDF representation**
   (a) Generate sparse feature vectors using TF-IDF with 1–2 grams.
   (b) Capture lexical-level frequency-based features.

3.3. Hierarchical Similarity Matching with Block Merging

To achieve efficient and accurate sentence-level plagiarism detection, we propose a hierarchical similarity matching algorithm that integrates multiple similarity metrics in a tiered decision structure.

Given two sentence sets from a suspicious and a source document, we compute similarity scores using three distinct methods:

- **Sentence-BERT similarity ($\text{sim}_{\text{SBERT}}$)**: We encode sentences using a pre-trained Sentence-BERT model and compute cosine similarity between their 384-dimensional embeddings. A sentence pair is considered matched if $\text{sim}_{\text{SBERT}}(x, y) > \alpha$ (where $\alpha = 0.35$).
- **MPNet similarity ($\text{sim}_{\text{MPNet}}$)**: We encode the same sentences using the MPNet model and compute cosine similarity between embeddings. A match is accepted if $\text{sim}_{\text{MPNet}}(x, y) > \beta$ (where $\beta = 0.50$).
- **TF-IDF similarity ($\text{sim}_{\text{TFIDF}}$)**: Using 1–2 gram TF-IDF vectors, we compute cosine similarity between sparse representations. A match is accepted if $\text{sim}_{\text{TFIDF}}(x, y) > \gamma$ (where $\gamma = 0.55$).

These three methods are applied in sequence, and the first method to surpass its threshold results in early acceptance. If none of the methods exceed their thresholds, we invoke a fallback scoring procedure using a fine-tuned BERT model:

- **BERT re-evaluation (sim_{BERT})**: A sentence pair (x, y) is passed to a pairwise classifier based on BERT, and the predicted similarity score is used. If $\text{sim}_{\text{BERT}}(x, y) > \delta$ (with $\delta = 0.45$), the match is accepted.To reduce computational overhead, the BERT re-evaluation is applied only as a fallback mechanism. Specifically, the set of candidate sentences y is constructed by taking the union of the top-k most similar sentences retrieved by Sentence-BERT, MPNet, and TF-IDF. This ensures that BERT operates exclusively on a compact, high-quality candidate pool, balancing precision with efficiency.

Once sentence-level matches are identified, we apply a block merging algorithm to group adjacent matches into longer plagiarized spans. Two sentence pairs (s_i, t_i) and (s_{i+1}, t_{i+1}) are considered part of the same block if:

1. $s_{i+1} > s_i$ and $t_{i+1} > t_i$ (strictly increasing order),
2. $s_{i+1} - s_i \leq \theta$ and $t_{i+1} - t_i \leq \eta$ (gap constraints),
3. The resulting block contains at least δ matched pairs (length constraint).

This merging logic is encapsulated in a function `isABlock()`, which determines whether two pairs can be joined based on the above criteria. The final result is a set of merged blocks indicating contiguous regions of potential plagiarism.

Algorithm 1 Hierarchical Similarity Matching with Block Merging

Input: Sentence sets from suspicious and source documents
Output: Merged matching blocks

Compute sentence embeddings using Sentence-BERT, MPNet, and TF-IDF
Initialize match list as empty
for each sentence x in suspicious document **do**
    Retrieve top-k similar sentences y from source document using each method
    if $\text{sim}_{\text{SBERT}}(x, y) > \alpha$ **then**
        Accept match
        continue
    end if
    if $\text{sim}_{\text{TFIDF}}(x, y) > \beta$ **then**
        Accept match
        continue
    end if
    if $\text{sim}_{\text{MPNet}}(x, y) > \gamma$ **then**
        Accept match
        continue
    end if
    for each candidate y in the union of top-k results **do**
        if $\text{sim}_{\text{BERT}}(x, y) > \delta$ **then**
            Accept match
            break
        end if
    end for
end for
Sort all accepted matches by sentence position
Initialize empty block list
for each pair of adjacent matches b_i, b_{i+1} **do**
    if `isABlock`(b_i, b_{i+1}, θ, η, δ) **then**
        Merge b_i and b_{i+1} into one block
    end if
end for
return all merged blocks

Our approach combines deep learning-based models (Sentence-BERT, MPNet) with traditional TF-IDF features to perform complementary semantic and surface-level analysis. The fine-tuned BERT classifier is used to resolve borderline cases, while the multi-stage merging strategy significantly improves the detection of continuous plagiarized segments.

4. Experiments

4.1. Experimental Settings

To assess the performance of our hybrid similarity computation and block merging algorithm, we used the following hyperparameters:

- Threshold α for Sentence-BERT similarity: 0.35
- Threshold β for MPNet similarity: 0.50
- Threshold γ for TF-IDF similarity: 0.55
- Fallback threshold δ for BERT similarity: 0.45

- Maximum allowed position gap θ for block merging: 5
- Minimum block length η: 2
- Top-k retrieved candidates per method: 5

These parameters were selected empirically based on performance on the training set. Sentence pairs passing any of the first three thresholds are accepted. If none qualify, BERT re-evaluation with threshold δ is applied. Finally, adjacent sentence matches are merged into blocks using a context-aware policy governed by θ and η

All experimental code and configurations used in this study have been released as open-source and are available at: https://github.com/CCheZi/Generative-Plagiarism-Detection

4.2. Results

Our system achieved a Plagdet score of 0.496 on the llm-plagiarism-detection-spot-check-20250521-training dataset. Table 2 shows the detailed evaluation performance, including precision, recall, and granularity. These results demonstrate the effectiveness of the hybrid similarity scoring and block merging strategy.

Table 2
Evaluation results on the dataset

Corpus	Plagdet	Recall	Precision	Granularity
llm-plagiarism-detection-spot-check-20250521-training	0.496	0.578	0.600	1.275

5. Conclusion and Future Work

Our plagiarism detection system achieves initial detection capabilities by integrating multi-model semantic representations (Sentence-BERT, MPNet) with traditional TF-IDF features, combined with a hybrid similarity computation strategy. While the current method demonstrates basic effectiveness, there is still considerable room for improvement. Experimental results show that the system performs well in detecting overt plagiarism (e.g., verbatim copying), but remains limited in handling texts that have undergone complex rewrites.

The main contributions of this work include: (1) Multi-model fusion architecture: This is the first approach to compute Sentence-BERT, MPNet, and TF-IDF in parallel, enabling early-stage matching decisions through a threshold-based mechanism; (2) Hybrid similarity computation: A three-stage matching strategy is designed—direct matching $\rightarrow$ candidate merging $\rightarrow$ BERT-based verification—to balance efficiency and accuracy; (3) Dynamic block merging algorithm: Non-contiguous plagiarized segments are handled via adjustable gap parameters (`max_susp_gap`/`max_src_gap`), allowing more flexible detection.

For future work, we plan to optimize the current method from the following aspects: (1) Parameter tuning: Currently, most parameters are heuristically set. We plan to adopt more advanced parameter optimization algorithms, such as genetic algorithms, to achieve global optimization. This will enable more precise adaptation to the characteristics of different corpora, thereby improving detection performance; (2) Incorporation of linguistic features: To better handle paraphrasing and semantic shifts, we aim to incorporate linguistically informed techniques, such as semantic role labeling and dependency parsing. These methods can help capture deeper semantic similarities between texts and reduce false positives; (3) Utilization of contextual information: The current detection approach primarily focuses on sentence-level similarity and overlooks document-level context. We will explore incorporating contextual information, taking into account sentence positioning and discourse context to further enhance detection accuracy.

In conclusion, although this study has made progress in plagiarism detection, there is still room for performance improvement. With the proposed enhancements, we believe our approach can be further refined to become more competitive for future real-world applications.

Acknowledgments

This work is supported by the National Social Science Foundation of China (Grant No. 22BTQ101).

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT-4 for the following activities: content drafting, grammar and spelling check, paraphrasing, and rewriting. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] A. Barrón-Cedeño, M. Vila, M. A. Martí, P. Rosso, Plagiarism meets paraphrasing: Insights for the next generation of automatic plagiarism checkers, Computational Linguistics (2013).

[2] S. M. Alzahrani, N. Salim, A. Abraham, Understanding plagiarism: Linguistic patterns, textual features, and detection methods, IEEE Transactions on Systems, Man, and Cybernetics (2012).

[3] M. Potthast, et al., Overview of the pan 2023 shared tasks on digital text forensics, in: Working Notes of CLEF, 2023.

[4] PAN@CLEF Lab, Generated plagiarism detection - pan at clef 2025, https://pan.webis.de/clef25/pan25-web/generated-plagiarism-detection, 2025. Accessed: May 28, 2025.

[5] M. Sanchez-Perez, G. Sidorov, A. Gelbukh, A winning approach to text alignment for text reuse detection at pan 2014, in: Working Notes for CLEF, 2014.

[6] Y. Palkovskii, A. Belov, Developing high-resolution universal multi-type n-gram plagiarism detector, in: Working Notes for CLEF, 2014.

[7] D. Glinos, A hybrid architecture for plagiarism detection, in: Working Notes for CLEF, 2014.

[8] P. Shrestha, S. Maharjan, T. Solorio, Machine translation evaluation metric for text alignment, in: Working Notes for CLEF, 2014.

[9] D. Rodríguez Torrejón, J. Martín Ramos, Coremo 2.3 plagiarism detector text alignment module, in: Working Notes for CLEF, 2014.

[10] Explosion AI, spacy: Industrial-strength natural language processing in python (version 3.8.5), https://spacy.io, 2024.

[11] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, 2019.

[12] J. Johnson, M. Douze, H. Jégou, Billion-scale similarity search with gpus, IEEE Transactions on Big Data (2019). FAISS Library: https://github.com/facebookresearch/faiss.

[13] K. Song, X. Tan, T. Qin, J. Lu, T.-Y. Liu, Mpnet: Masked and permuted pre-training for language understanding, in: Advances in Neural Information Processing Systems (NeurIPS), 2020.

[14] J. Bevendorff, D. Dementieva, M. Fröbe, B. Gipp, A. Greiner-Petter, J. Karlgren, M. Mayerl, P. Nakov, A. Panchenko, M. Potthast, A. Shelmanov, E. Stamatatos, B. Stein, Y. Wang, M. Wiegmann, E. Zangerle, Overview of pan 2025: Voight-kampff generative ai detection, multilingual text detoxification, multi-author writing style analysis, and generative plagiarism detection, in: CLEF 2025, Lecture Notes in Computer Science, Springer, 2025.

[15] M. Fröbe, M. Wiegmann, N. Kolyada, B. Grahm, T. Elstner, F. Loebe, M. Hagen, B. Stein, M. Potthast, Continuous integration for reproducible shared tasks with tira.io, in: Advances in Information Retrieval. 45th European Conference on IR Research (ECIR 2023), Springer, 2023, pp. 236–241.

[16] A. Greiner-Petter, M. Fröbe, J. P. Wahle, T. Ruas, B. Gipp, A. Aizawa, M. Potthast, Overview of the generative plagiarism detection task at pan 2025, in: Working Notes of CLEF 2025 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, CEUR-WS.org, 2025.