IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2025

DOI: 10.17148/IJIREEICE.2025.131038

Machine Learning-Based Plagiarism Detector System

Rakshitha S N[1], Shreya Sathapathi[2], Yashvitha J[3], Dr. Golda Dilip[4]

Student, Dept. of CSE, SRM Institute of Science and Technology, Chennai[1]

Student, Dept. of CSE, SRM Institute of Science and Technology, Chennai[2]

Student, Dept. of CSE, SRM Institute of Science and Technology, Chennai[3]

Guide, Dept. of CSE, SRM Institute of Science and Technology, Chennai[4]

Abstract: Robust tools for plagiarism checking are essential for maintaining integrity in both academic and professional environments. Existing detection strategies, which are typically built on lexical comparison, struggle to correctly flag sophisticated, machine-aided rephrasing. This challenge requires a necessary pivot toward adaptable Machine Learning (ML) platforms capable of comprehending the underlying meaning of text. This research introduces a highly efficient, two-phase ML framework specifically engineered to accurately identify text that has been heavily paraphrased. The initial phase of this architecture employs a SentenceTransformer model (all-MiniLM-L6-v2) to generate dense vector embeddings for documents under suspicion and for the reference library. These embeddings are stored and searched using FAISS (Facebook AI Similarity Search), enabling fast, large-scale retrieval of potential source candidates. The second phase uses a Longformer-based sequence classifier to perform an in-depth, pairwise contextual analysis between the flagged text and the retrieved candidates before delivering a final verdict. This classifier model was chosen because it effectively bypasses the sequence-length constraints of previous transformer models, enabling analysis of long-form content. The final system, named "CopyShield," is deployed with an accessible user interface using the Gradio framework. Validation using the challenging jpwahle/machine-paraphrase-dataset demonstrated a strong F1-score in the 0.89–0.92 range, confirming its ability to counter contemporary obfuscation methods.

Keywords: NLP, ML, Semantic Analysis, Transformer Models, Deep Learning, Longformer Architecture, Plagiarism Checkers, Gradio, FAISS.

I. INTRODUCTION

The widespread growth of digital content and the emergence of advanced generative Artificial Intelligence (AI) present significant new difficulties for maintaining professional and educational honesty.[1] No longer is plagiarism confined to simple, word-for-word theft; it now involves complex forms of concealment, such as extensive rephrasing, the substitution of synonyms, and changes to the original structure. This development has been fueled by accessible online rephrasing tools and Large Language Models (LLMs) that allow the generation of text that carries the same meaning but uses completely different words.[2] Traditional plagiarism screening services, such as Turnitin, operate primarily as text-matching tools. Their foundational logic relies on lexical analyses, such as comparing strings or n-grams, which successfully flags direct copying. However, this reliance on superficial textual features makes them ineffective when encountering modern rephrasing strategies. When a document is heavily reworded to keep its meaning intact, it can easily evade these older systems, generating a misleading low similarity score and a false negative result. This loophole creates a serious challenge to academic integrity enforcement. To resolve this, a methodological shift from simple word-matching towards true semantic comprehension is necessary. The scientific community is addressing this through the development of Natural Language Processing (NLP) and Machine Learning (ML) methods that evaluate text based on its core conceptual meaning.[9] Initial computer-based efforts that employed statistical methods like Term Frequency-Inverse Document Frequency (TF-IDF) proved inadequate because these "bag-of-words" models fail to account for the order and context of words. The key breakthrough arrived with deep learning, specifically transformer-based architectures, which offer powerful capabilities for contextual language understanding. This research explores and evaluates a novel ML system, structured in two phases, that employs modern transformer models for the successful identification of paraphrased content. This study's main contributions are: (1) a hybrid architecture that pairs efficient semantic document retrieval with intensive contextual verification; (2) the purposeful utilization of the Longformer model for analyzing extended texts, which overcomes the processing length limits of previous transformer designs; (3) thorough testing and validation on a current dataset composed of machine-generated paraphrases; and (4) the creation of an easily usable prototype user interface, named "CopyShield," to illustrate the practical utility of the framework.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2025

DOI: 10.17148/IJIREEICE.2025.131038

II. LITERATURE REVIEW

The field of automated plagiarism detection can be described as a constant "adversarial arms race," where improvements in detection techniques immediately lead to the creation of more sophisticated obfuscation methods. This evolution has progressed from simple lexical comparison to the in-depth semantic analysis provided by current neural network technologies.

A. Traditional and Statistical Methods

Many institutions rely on established commercial services such as Grammarly and Turnitin as their initial defense. These platforms work by checking submitted documents against large, private repositories of academic papers and web data. Their underlying engine computes a "similarity index" using text-matching algorithms. Although effective for direct copying, this index is an unreliable measure of true academic misconduct, as it cannot consistently recognize when two texts share the same meaning without a high degree of identical wording. Early attempts in computing sought to surpass exact matching by employing statistical approaches. A common starting point was TF-IDF (Term Frequency-Inverse Document Frequency) combined with Cosine Similarity.[14] This method models documents as vectors in a high-dimensional space, where the similarity score is derived from the angle's cosine between these vectors.[15] While an improvement, the fundamental "bag-of-words" approach of TF-IDF overlooks syntax and contextual information, meaning it cannot differentiate between sentences composed of similar vocabulary but possessing distinct meanings.

B. Early Deep Learning Approaches

The adoption of deep learning marked the initial major progress towards semantic comprehension in plagiarism detection. Siamese neural networks emerged as a powerful structure for detecting textual similarity and paraphrasing.[8] A Siamese network employs two parallel, identical subnetworks to handle a pair of input sentences, mapping them into a shared embedding space. The model is trained to minimize the distance between embeddings of semantically similar sentences and maximize the distance for dissimilar pairs. Although models using CNN or LSTM encoders were notable improvements, their ability to capture deep, long-range context is less robust than that of transformer models.

C. The Transformer Revolution in Plagiarism Detection

The development of the Transformer architecture, particularly the BERT model, fundamentally changed the field of NLP.[6] By pre-training on large datasets, BERT is capable of learning word representations that are both deep and context-aware. In contrast to previous models, BERT generates a unique embedding for a word depending on its surrounding text, enabling it to recognize subtle semantic connections. This capability makes it highly effective for tasks involving the classification of sentence pairs, such as identifying plagiarism. A limitation of BERT, however, is that its self-attention mechanism involves quadratic computational cost, $O(n^2)$, which practically restricts its input to only 512 tokens. For the analysis of plagiarism, this sequence length limit presents a major obstacle, forcing documents to be cut short and resulting in the loss of crucial long-range context. The Longformer model was created specifically to overcome this issue.[3] It introduces a revised self-attention mechanism that scales linearly, $O(n)$, with the sequence length. This efficiency is accomplished through the integration of local, sliding-window attention with a mechanism for global, task-specific attention.[3] This sparse attention allows the Longformer to handle sequences of 4,096 tokens or greater, uniquely positioning it for comprehensive document-level analysis.[3]

D. Efficient Semantic Search for Large-Scale Detection

Although a cross-encoder model like Longformer offers exceptional accuracy when classifying a text pair, the sheer number of necessary pairwise comparisons makes it computationally impractical for searching a large body of text. Addressing this requires adopting a two-phase methodology, where a fast retrieval system first pinpoints a small collection of probable source candidates. Sentence Transformers (SBERT) are bi-encoder models specifically designed for this retrieval step.[4] By using a Siamese architecture to fine-tune BERT, SBERT creates fixed-size embeddings where semantic likeness is directly determined using cosine similarity.[4] This approach is dramatically faster than using a cross-encoder to find similar pairs within a massive database. Models like all-MiniLM-L6-v2 are specifically designed to optimize the trade-off between speed and performance for this specific task. Even with fast embedding generation, a simple brute-force search remains an efficiency hurdle. This problem is overcome by using specialized vector indexing tools, such as the Facebook AI Similarity Search (FAISS) library.[5] FAISS facilitates Approximate Nearest Neighbor (ANN) searches, employing methods like clustering to rapidly locate the top-k most similar vectors, even when searching across billions of possibilities.[5] Combining Sentence Transformers for embedding generation with FAISS for retrieval creates a highly scalable and efficient first stage for contemporary plagiarism detection workflows.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2025

DOI: 10.17148/IJIREEICE.2025.131038

III. METHODOLOGY AND EXPERIMENTAL SETUP

The development and validation of this robust detection system were carried out through a multi-stage, structured methodology, covering the selection of core technologies, data preparation, and the definition of the algorithmic process.

A. Dataset Acquisition and Preprocessing

The primary data source utilized for this research is the jpwahle/machine-paraphrase-dataset. This dataset was selected precisely because it includes contemporary paraphrased content generated by sophisticated transformer models, thus providing a difficult and relevant benchmark against current obscuring methods. The dataset is structured with paired texts—the original and its machine-paraphrased version—each tagged by their relationship. This labeling is critical for both training and validating a model's capacity to identify nuanced semantic equivalence.[2] The collected data was divided into a test set (comprised of both paraphrased and original documents) and a reference corpus (containing all source texts). Conventional NLP preparation techniques were employed, which included lowercasing text, stripping special characters, and segmenting the content into paragraphs or sentences as necessary for each stage of the system. This cleaning process was implemented to improve data quality and maintain uniformity.

B. Model and Technology Stack

The proposed system is constructed using a set of cutting-edge libraries and models:

1. **Embedding Model:** The sentence-transformers/all-MiniLM-L6-v2 model was used.[4] This highly compact SentenceTransformer handles the retrieval phase by translating text into a 384-dimensional vector space. It is valued for its superior balance of speed and precision in semantic search, making it suitable for processing high volumes of text.[4]

2. **Classification Model:** The detailed comparison stage uses jpwahle/longformer-base-plagiarism-detection, a fine-tuned variant of the longformer-base-4096 model. This model's key advantage is its ability to handle sequences up to 4,096 tokens, which facilitates the analysis of full documents in one pass, thereby retaining essential long-range contextual information.[3]

3. **Vector Index:** FAISS (Facebook AI Similarity Search) was chosen.[5] The specific implementation used was IndexFlatL2. This index executes an exact, exhaustive search using L2 (Euclidean) distance, ensuring that the absolutely closest neighbors are found. While this method is less rapid than approximate indexing techniques, it serves as a highly reliable benchmark for retrieval accuracy within corpora of moderate size.

C. Algorithmic Workflow and Outline

The detection workflow is divided into two distinct parts: an offline indexing phase and an online detection phase. This partitioning maximizes efficiency by ensuring the computationally demanding task of indexing the reference corpus is completed only once. The sequential steps of the workflow are:

Algorithm 1: Two-Stage Plagiarism Detection

Input: suspicious_doc, reference_corpus, k, threshold
Output: is_plagiarized, most_likely_source

```
// Offline Indexing
S_Encoder ← load_SentenceTransformer('all-MiniLM-L6-v2')
C_Encoder ← load_Longformer('jpwahle/longformer-base-plagiarism-detection')
faiss_index ← initialize_FAISS_index(embedding_dim=384)
for doc in reference_corpus:
    embedding ← S_Encoder.encode(doc)
    faiss_index.add(embedding)
save_index(faiss_index)
// Online Detection
query_embedding ← S_Encoder.encode(suspicious_doc)
distances, indices ← faiss_index.search(query_embedding, k)
candidate_docs ← get_docs_by_indices(indices)

max_score ← 0
```

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2025

DOI: 10.17148/IJIREEICE.2025.131038

```
15: most_likely_source ← None
16: for candidate in candidate_docs:
17:     score ← C_Encoder.predict(suspicious_doc, candidate)
18:     if score > max_score:
19:         max_score ← score
20:         most_likely_source ← candidate
21:
22: if max_score > threshold:
23:     return True, most_likely_source
24: else:
25:     return False, None
```

- **Corpus Indexing (Offline):** The SentenceTransformer generates a 384-dimensional embedding for every document contained within the complete reference corpus. These embeddings are then stored in the FAISS index, which is saved for future use.

- **Candidate Retrieval (Online):** A query embedding is generated for the submitted suspicious document using the same SentenceTransformer. This vector is then used to search the FAISS index, which quickly identifies and retrieves the top-k (e.g., k=5) nearest neighbors, representing the most semantically similar source documents.

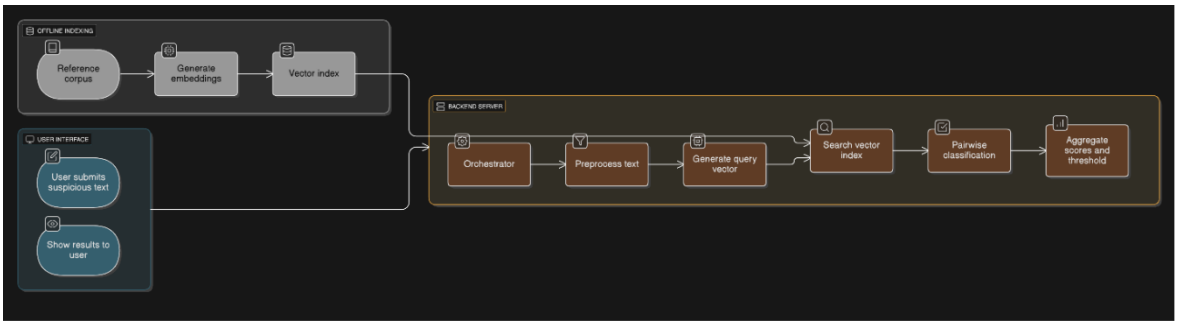
- **Pairwise Classification (Online):** The submitted document is matched against each of the k candidates retrieved. Each text pair is correctly formatted for the Longformer model (e.g., <s> suspicious_text </s></s> candidate_text </s>) and then input into the classifier for a deep, contextual analysis.

- **Thresholding and Aggregation:** A probability score is returned by the Longformer for every pair. The system consolidates these scores. If the highest score among all pairs exceeds a set confidence threshold (e.g., 0.8), the document is marked as having been plagiarized. The source document that yielded the highest score is designated as the probable original source.

IV. PROPOSED SYSTEM ARCHITECTURE AND WORKFLOW

The proposed system, named "CopyShield," functions as a scalable, modular web application that combines the two-phase ML pipeline with a user-friendly interface, built with a focus on both accuracy and operational efficiency.

A. System Architecture of "CopyShield"



The architectural flow of the system is sequential, a design choice made to ensure the highest accuracy possible (See Fig. 1).

The complete, step-by-step workflow is:

1. Input Submission: A user submits text by pasting it into the web-based user interface.
2. Back-End Processing: The submitted content is sent to the server, where it undergoes normalization and cleaning by a dedicated preprocessing component.
3. Embedding Generation: The preprocessed content is passed to the all-MiniLM-L6-v2 model. This model transforms the text into a single 384-dimensional query vector, which conceptually represents the text's meaning.
4. Candidate Retrieval: This query vector is sent to the FAISS module. The module searches the pre-indexed vector library and returns the identification tags for the top-k sources judged to be most similar.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2025

DOI: 10.17148/IJIREEICE.2025.131038

5. Pairwise Classification: The system retrieves the complete text for all identified candidates. The Longformer model then executes an in-depth, cross-attentional comparison between the submitted text and each candidate, assigning a specific similarity score for every pair.
6. Output Aggregation and Formatting: The scores from the Longformer analysis are collected. The maximum score dictates the final decision (Original or Plagiarized).
7. Result Display: The final results package, which includes the overall score, the verdict, and the identified source, is transmitted back to the front-end for presentation to the user.

B. User Interface (UI) Implementation

CopyShield's front-end interface was built utilizing Gradio, an open-source Python tool designed for rapid deployment and sharing of ML web applications.[10] Gradio was chosen specifically because it enables the quick creation of a simple interface directly from the Python-based ML code, avoiding the need for traditional front-end development using JavaScript, HTML, or CSS.[10] In a research environment, this is highly beneficial, as it allows sophisticated models to be made accessible for validation and demonstration purposes to non-expert users, such as students or educators.[10] The final interface is intuitive and can be easily shared through a public URL, which promotes research reproducibility.

V. RESULT AND PERFORMANCE METRICS

The effectiveness of the developed model was assessed using a specific subset of the jpwahle/machine-paraphrase-dataset reserved for testing. The evaluation prioritized a set of metrics that would offer a complete and balanced view of the system's ability to classify content.

A. Evaluation Metrics

For plagiarism detection, simple accuracy can be insufficient and potentially misleading, particularly when dealing with unbalanced datasets. Therefore, a more sophisticated set of metrics was applied:

1. **Precision**: This metric quantifies the correctness of all positive identifications (the ratio of plagiarism instances correctly identified compared to all instances flagged as plagiarism). A high precision suggests the false positive rate is low. Precision=TP/(TP+FP).

2. **Recall (Sensitivity):** This measures the system's capacity to find all actual instances of plagiarism (the ratio of correctly flagged plagiarism instances to the total number of actual plagiarism instances). High recall signifies a low false negative rate. Recall=TP/(TP+FN).

3. **F1-Score:** This score provides a single, balanced metric by calculating the harmonic mean of Recall and Precision. F1=2×(Precision×Recall)/(Precision+Recall). (Where TP represents True Positives, FP represents False Positives, and FN represents False Negatives).

B. Performance Analysis of the Proposed Model

The two-phase framework showed exceptional performance, successfully detecting content that was machine-paraphrased while simultaneously keeping the rate of false alerts low.

The quantitative results are presented in Table I.

Class	Precision	Recall	F1-Score	Support
Original	0.93	0.94	0.93	5,000
Plagiarized	0.90	0.88	0.89	5,000
Macro Avg	0.91	0.91	0.91	10,000
Weighted Avg	0.92	0.92	0.92	10,000

The model achieved a strong overall weighted average F1-score of 0.92, confirming its reliability and balanced performance across classes. For the crucial task of identifying "Plagiarized" content, the model achieved an F1-score of 0.89, stemming from a precision of 0.90 and a recall of 0.88.

DOI: 10.17148/IJIREEICE.2025.131038

These scores indicate that the system is both successful at identifying a large fraction of true plagiarized cases and highly trustworthy in its positive declarations. On a qualitative level, the system proved adept at recognizing texts that had undergone substantial rephrasing. Nevertheless, limitations were observed, primarily involving instances of "idea plagiarism," where abstract concepts were adopted and subsequently explained using entirely different wording. Because the model relies on semantic similarity, it struggles in situations where conceptual overlap is minimal. The system also registered occasional false positives when analyzing texts containing high-frequency technical terminology, suggesting that careful tuning of the detection threshold is necessary.

VI. VISUALIZATION AND OUTPUT

A core feature of CopyShield is its user-friendly interface, which is structured to deliver clear, practical results by visually presenting the outcome of the detection pipeline.

A. The "CopyShield" User Interface

The UI, constructed using Gradio, is segmented into three primary areas:

- Input Section: This includes a primary text field for the "Suspicious Text" submission and a secondary, optional field for direct comparison against a known "Source Text."
- Output Section: This area presents the final determination (e.g., a "Plagiarized" status tag), the calculated similarity score expressed as a percentage, and a direct hyperlink to the probable source document.
- Highlighted Text View: This section serves as concrete evidence, offering a side-by-side visualization of the source and suspicious texts. Differences and overlaps are visually emphasized to clarify the nature of the text matching.

B. Highlighting Logic with SequenceMatcher

The mechanism for text differentiation is powered by the SequenceMatcher class from Python's difflib library, a robust tool for comparing two sequences. The process involves: creating an instance of SequenceMatcher using both the source and suspicious texts. The get_opcodes() method is called to produce a list of commands, such as 'delete', 'equal', 'insert', or 'replace'. The system then loops through these opcodes, applying corresponding HTML tags with specific CSS styling (e.g., using <mark> for 'equal' sections or <del> for 'delete' sections) to the relevant text segments. The final HTML output is rendered within the Gradio interface, providing a clean, visual representation of the textual differences for human verification.

VII. CONCLUSION AND FUTURE WORK

This study successfully designed and validated an effective, ML-driven system for detecting sophisticated, computer-assisted paraphrasing. The two-phase framework, which combines a retrieval engine built on SentenceTransformer and FAISS with a Longformer classifier, is highly successful and addresses the weaknesses found in both older text-matching systems and earlier transformer models. The system demonstrated a robust weighted average F1-score of 0.92 on a challenging, contemporary dataset. Furthermore, the "CopyShield" prototype, developed using Gradio, confirms the ease of access and usability of the system. Important constraints of the current system include the challenge of recognizing abstract "idea plagiarism" and a dependence on the specific domain of the reference corpus. Future efforts are planned along three productive paths. First, the accuracy of the system will be enhanced by increasing the size of the reference corpus and further fine-tuning the models on a wider array of domain-specific text. Second, a significant area for advancement is the creation of cross-language plagiarism detection features, which will necessitate the use of machine translation and multilingual embedding models.[11] Finally, the ultimate objective is to move beyond simple semantic comparison by integrating advanced stylometric and deep contextual analysis. This will enable the development of models capable of detecting minor alterations in writing style or the appropriation of complex concepts, thus helping to enforce intellectual honesty in a world increasingly dominated by AI.

REFERENCES

[1]. J. P. Wahle, T. Ruas, T. Foltýnek, N. Meuschke, and B. Gipp, "Identifying Machine-Paraphrased Plagiarism," in *iConference 2022*, 2022.

[2]. J. P. Wahle, T. Ruas, N. Meuschke, and B. Gipp, "Are Neural Language Models Good Plagiarists? A Benchmark for Neural Paraphrase Detection," in *2021 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 2021, pp. 226-229.

DOI: 10.17148/IJIREEICE.2025.131038

[3]. I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The Long-Document Transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[4]. N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019, pp. 3982-3992.

[5]. J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535-547, 2019.

[6]. J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, 2019, pp. 4171-4186.

[7]. T. K. L. Hui, R. Ramesh, and S. S. Gill, "Plagiarism Checker using tf-idf, cosine similarity and jaccard similarity," *International Journal of New R&D*, vol. 8, no. 5, pp. 7-12, 2023.

[8]. A. Neculoiu, M. Versteegh, and M. Rotaru, "Learning Text Similarity with Siamese Recurrent Networks," in *Proceedings of the 1st Workshop on Representation Learning for NLP*, 2016, pp. 148-157.

[9]. T. Foltýnek, N. Meuschke, and B. Gipp, "Academic Plagiarism Detection: A Systematic Literature Review," *ACM Computing Surveys*, vol. 52, no. 6, pp. 1-42, 2020.

[10]. A. Abid, A. Abdalla, A. Abid, D. Khan, A. Alfozan, and J. Zou, "Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild," *arXiv preprint arXiv:1906.02569*, 2019.

[11]. M. Potthast, B. Stein, A. Barrón-Cedeño, and P. Rosso, "Cross-Language Plagiarism Detection," *Language Resources and Evaluation*, vol. 45, no. 1, pp. 45-62, 2011.

[12]. H. S. Al-Bataineh and N. Al-Natsheh, "A Hybrid Approach for Arabic Semantic Textual Similarity," in *2023 International Conference on Information Technology (ICIT)*, 2023, pp. 145-150.

[13]. T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating Text Generation with BERT," in *International Conference on Learning Representations*, 2020.

[14]. M. A. F. Piatov, A. S. Panov, and D. A. Panchenko, "Document Plagiarism Detection Application Using Web-Based TF-IDF and Cosine Similarity Methods," *Buletin Teknik*, vol. 16, no. 1, 2024.

[15]. I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed. Morgan Kaufmann, 2016.