SkillDux
https://journal.skilldux.com/Security_Trends-international-journal.php
Journal of Information, Data, and Security Trends

An Enhanced Hybrid Framework for Intelligent Research Paper Plagiarism Detection: Integrating TF-IDF Lexical Vectorization, SBERT Deep Semantic Embeddings, and Explainable Risk Analytics

***Varun K S*[1] *and Varun R Shivannanavar*[2]**
[1]Assistant Professor, Department of Computer Applications, Varun R Shivannanavar & Varun, G M University, Davanagere, Karnataka, India
[2]Department of Computer Applications, Varun R Shivannanavar & Varun, G M University, Davanagere, Karnataka, India
*** Email:** varunksbennur@gmail.com; varunspikee@gmail.com

***

Abstract: *Traditional plagiarism checkers are built around verbatim string and n-gram matching, which do not account for paraphrasing and other forms of semantic rewriting, making research integrity screening susceptible to bypassing. This paper introduces an enhanced research paper plagiarism detection system implemented in Python, Flask, PyMuPDF, Scikit-learn, and Sentence Transformers. The tool is designed to address the mentioned limitations by utilising a two-step approach that integrates a Lexical Similarity Engine (TF-IDF and cosine similarity) and a Deep Semantic Engine (SBERT paraphrase-MiniLM-L3-v2, with 384-dimensional embeddings). By combining the outputs of both engines, the system calculates a composite plagiarism score, a Research Integrity Index score, and implements a tiered risk classification. Additionally, the tool offers four key features that distinguish it from commercial alternatives, including section-wise analytics, a sentence highlighter, a citation-aware filtering module, and an institutional-level analytics dashboard. An instrumented end-to-end pipeline utilised a 42-paper corpus to demonstrate the system's applicability. The results showed that all six pipeline stages took a median of 260 ms to process per document. Testing on identical, heavily paraphrased, and unrelated documents revealed that the hybrid approach maintained an approximate detection capability of 68% for semantically rewritten texts compared to 6% for the TF-IDF baseline, while citation-aware filtering decreased the similarity score of an appropriately cited excerpt from 54.9% to 11.6%. SQLite and MySQL compatibility and light deployment requirements allow the system to operate on a standard server without requiring extensive optimisation.*

***Keywords*:** *Plagiarism Detection; Semantic Textual Similarity; Sentence-BERT Embeddings; TF-IDF Cosine Similarity; Citation-Aware Filtering; Explainable Risk Analytics.*

Received: October 29, 2025 **Revised:** November 26, 2025 **Accepted:** January 21, 2026

Nomenclature

Abbreviation	Description
TF-IDF	Term Frequency–Inverse Document Frequency
SBERT	Sentence Bidirectional Encoder Representations from Transformers
NLP	Natural Language Processing
MVC	Model–View–Controller
ORM	Object-Relational Mapping
PBKDF2	Password-Based Key Derivation Function 2
RBAC	Role-Based Access Control
DoS	Denial of Service
FAISS	Facebook AI Similarity Search
OCR	Optical Character Recognition
PAN	Plagiarism Analysis, Authorship Identification and Near-Duplicate Detection (benchmark series)
MCA	Master of Computer Applications

1. Introduction

The growth of digital academic publishing has created both unprecedented opportunities for scientific exchange and new challenges in detecting uncredited content duplication. Researchers who feel pressured to publish, or who employ dishonest means to do so, now have access to an expanding array of synonym-substitution algorithms, back-translation pipelines, and large language models that can rephrase an entire paragraph while retaining its original meaning [10][11][14][18]. The growing disconnect between what plagiarism-detection vendors promise and what such tools can deliver in practice has fueled an urgent need for tools that can reason about meaning rather than form.

Traditional plagiarism checkers operate by comparing document character n-grams or lexemes to a reference corpus; such approaches have excellent recall in the case of straightforward text duplication but perform poorly when faced with synonym substitution at the level of individual words or higher-level discourse structures such as sentence rearrangement [8][10][11]. A second limitation of the state-of-the-art is that most commercial similarity checkers only report an overall percentage of matching text without drill-down capabilities, and conflate actual plagiarism with properly attributed quotation, leading to potential false positives.

This paper describes a web application for AI-based plagiarism checking that addresses both of these limitations. Implemented in Python atop a Flask backend [4][5], it implements bag-of-words and sentence embedding similarity checkers as modular pipelines [1][3][9] and provides an explainability layer, a citation-aware filtering mechanism, and a longitudinal analytics dashboard that transform the similarity checker from a black-box marketing gimmick into a transparent research integrity support tool. The technical claims in this paper are supported by specific implementation artifacts: modules implementing the described capabilities are written in the Python programming language, and their operation is visible through the system implementation and execution. The main contributions of this paper are:

- the creation of an end-to-end production-quality plagiarism-checking application with explainable results for research integrity applications;
- the adoption of a hybrid approach to document similarity, combining traditional TF-IDF vector-space representations with recent advances in paraphrase detection using SBERT (paraphrase-MiniLM-L3-v2) sentence embeddings [1][14][19];
- an implementation of section-specific similarity checking, including partitioning of the input document into Abstract, Introduction, Methodology, Results, and Conclusion, as well as generic sections;
- the definition of a family of plagiarism scores, including Total Plagiarism Index, Research Integrity Score, and associated risk categories, along with an implementation of an adaptive weighting algorithm for technical documents;
- the addition of an explainability layer that identifies sentence pairs whose similarity contributes to a high overall score and highlights them for the user as potential targets for further investigation, limited to the top-k most relevant matches [24];
- the implementation of a citation-aware filtering layer to distinguish approximately attributed text from potentially problematic similarity, with accompanying reduction in false positives [12];
- and visual analytics and reporting tools, including section heatmaps and PDF certificate export [20], as well as a dashboard for institutional-level analytics over time.

As a practical contribution, this paper contains an extensive evaluation of the approach on a 42-paper corpus; results, described in detail in Section 9, demonstrate that the hybrid approach provides improved sensitivity to semantically rewritten content: on the paraphrased document test set, the hybrid approach achieved approximately 68% detection capability, compared to 6% for the TF-IDF baseline. The ability to recognize properly-attributed quotation as non-plagiarized text was also crucial in reducing false positives: on the false-positive test set, the citation-aware filtering reduced the score of a properly quoted excerpt from 54.9% to 11.6%. The remainder of this paper is structured as follows: Section 2 reviews related work and describes the research gap that this paper addresses. Section 3 defines the problem scenario and scoring requirements; Sections 4-8 describe the implementation, database design [28][29], modules [21][25][27], algorithms, and experiments performed . Section 9 contains an experimental evaluation of the system, including a comparison of the similarity scores produced by different sentence embedding architectures grounded in a survey of the relevant literature [15][16][17][26]. Finally, Section 10 discusses system security and deployment [22][23], while Sections 11-13 describe the technical content of the implementation, including the public API, in greater detail, and Section 14 concludes the paper.

2. Literature Review and Related Work

Plagiarism detection has seen three broad generations of methods: lexical string/n-gram matching, syntactic/stylometric parsing, and semantic embedding-based approaches [7]. This section summarizes prior work within each category, contrasts their reported strengths and weaknesses, and motivates this work's contributions.

2.1 Prior Work Summary

Vector-space and lexical methods. Salton et al. proposed the Vector Space Model, which represents documents as sparse vectors of term frequency and compares them using cosine similarity; this approach underlies the TF-IDF method central to this paper but ignores semantic relationships between terms. Barrón-Cedeño and Rosso benchmarked n-gram overlap for plagiarism detection, finding that character and word n-grams perform poorly when passages are paraphrased [8].

Survey and benchmark papers. Alzahrani, Salim, and Abraham surveyed linguistic phenomena, textual features, and detection methods in the field while classifying various obfuscation techniques not captured by lexical analysis [11]. Potthast et al. created the PAN benchmark corpus for plagiarism detection, enabling comparison of multiple systems including embeddings-based approaches [12]. Foltynek, Meuschke, and Gipp's survey provided a systematic breakdown of detection generations used in this paper, while Chong, Specia, and Mitkov demonstrated early NLP-based

approaches to paraphrase detection using shallow syntactic features [13].

Transformers and sentence encodings. Vaswani et al.'s [2] Transformer model and Devlin et al.'s [6] BERT language model enable context-dependent encoding of text, outperforming previous word2vec-style static embeddings. Reimers and Gurevych fine-tuned BERT using Siamese and triplet networks to obtain paraphrase detection performance on sentence-level embeddings comparable to human judgment, with minimal inference overhead; this architecture underlies the current system (paraphrase-MiniLM-L3-v2) [1]. Vrbanec and Meštrović benchmarked several approaches to corpus-based paraphrase detection, finding that embedding-based approaches significantly outperform lexical overlap features on standard paraphrase corpora. This result directly informs this paper's dual-method design [14].

Overall, this work builds on eight prior studies representing all generations of plagiarism detection. Each highlights a key limitation motivating the current work: (i) lexical methods significantly underperform against paraphrased text, and (ii) most prior systems operate as standalone similarity engines rather than end-to-end pipelines incorporating citational context.

Table 1 presents a comparative analysis of established text-similarity techniques alongside the paper's proposed dual-engine approach.

Table 1. Comparative mapping of prior lexical, syntactic, and semantic-embedding approaches against the proposed dual-engine system.

Ref.	Methodology	Advantages	Disadvantages
Salton et al. [3]	TF-IDF vector space model, cosine similarity	Fast, interpretable, no training data required	Blind to synonymy and paraphrase; purely lexical
Barrón-Cedeño & Rosso [8]	Character/word n-gram overlap comparison	Simple, language-agnostic, low computational cost	Detection collapses under moderate rewording
Alzahrani et al. [11]	Survey/taxonomy of obfuscation and detection methods	Establishes systematic obfuscation categories	Not a deployable detector; taxonomy only
Potthast et al. [12]	PAN shared-task benchmark corpus	Enables standardized comparative evaluation	Benchmark corpus, not an end-user system
Vaswani et al. [2]	Self-attention Transformer architecture	Long-range contextual modeling	Not designed for direct sentence-similarity use
Devlin et al. [6]	BERT bidirectional pre-training	Rich contextual token representations	Pooled embeddings are weak for sentence similarity
Reimers & Gurevych [1]	Siamese/triplet SBERT sentence embeddings	Fast, accurate sentence-level cosine similarity	Can score short technical spans as similar independent of true meaning overlap
Vrbanec & Meštrović [14]	Review of corpus-based paraphrase detection	Confirms embedding superiority over lexical baselines	Comparative synthesis, not a novel detector
This work	TF-IDF + SBERT dual-engine fusion with citation-aware filtering and explainability	Combines lexical precision with semantic recall; auditable and false-positive resistant.	Static equal-weighting requires further calibration; single-language corpus validated.

2.2 Research Gap and Challenges

As revealed by the surveyed literature, there is a certain dichotomy between the lexical scoring tools, which are fast and intuitive but lack semantic grounding, and the transformer-based approaches, which are typically trained and evaluated as standalone similarity engines and are thus not directly integrated into the institutional workflow. In particular, the following three issues stand out as research gaps. Firstly, most semantic-embedding-based detectors do not provide any sentence-level evidence for the detected similarity score, thus rendering the algorithm's decisions non-explainable and difficult to interpret. Secondly, while lexical similarity approaches can be equipped with sophisticated quotation detectors that differentiate between properly cited and plagiarized text, such mechanisms are largely absent from semantic models, which leads to numerous false positives when paraphrased sentences from scholarly publications are reused in a new paper. Thirdly, while many detectors use equal weights for lexical and semantic similarity scores, such an approach is not always rational, especially for scholarly texts, where automated scoring tends to be inflated due to the frequent use of abbreviations and mathematical notations. The current work addresses each of these issues, which is reflected in the contribution section. In particular, the explainability aligner described in this paper facilitates the interpretation of the detection results by providing evidence for each sentence in the suspicious paper. The new citation-aware filtering mechanism allows distinguishing between properly cited works and cases of potential plagiarism, while the adaptive-weighting extension proposed in this work allows fine-tuning the detector's functioning depending on the characteristics of the input text.

3. Problem Statement and Objectives

The research problem addressed in this work is formulated as follows: given a target manuscript D_target and an institutional repository corpus

$D_{corpus} = \{D_1, \ldots, D_N\}$, compute a composite plagiarism metric that (i) captures verbatim and near-verbatim lexical overlap, (ii) captures deeper semantic overlap that survives paraphrasing, and (iii) discounts overlap that is attributable to properly cited quotation rather than undisclosed reuse. The underlying hypothesis is that a fused lexical–semantic score detects a materially larger share of paraphrased plagiarism than either engine alone, while citation-aware filtering keeps the false-positive rate on legitimately quoted text acceptably low. The practical motivation is institutional: reviewers need a single, auditable score together with the evidence behind it, rather than an opaque percentage.

The system computes a composite plagiarism metric S_total for a target paper D_target against a repository corpus $D_{corpus} = \{D_1, \ldots, D_N\}$, combining Lexical (TF-IDF) and Semantic (SBERT) metrics:

$$S_{exact} = \max_j \left[\frac{V(D_{target}).V(D_j)}{\|V(D_{target})\| \|V(D_j)\|} \right] \times 100 \quad (1)$$

$$S_{semantic} = \max_j \left[\frac{E(D_{target}).E(D_j)}{\|E(D_{target})\| \|E(D_j)\|} \right] \times 100 \quad (2)$$

$$S_{total} = w_1 S_{exact} + w_2 S_{semantic}, \quad (w_1 = w_2 = 0.5) \quad (3)$$

$$I_{score} = 100.0 - S_{total} \quad (4)$$

Theoretical Justification of the Static Equal-Weighting Baseline

The baseline weighting $(w_1 = w_2 = 0.5)$ in Equation (3) is chosen as a variance-neutral fusion point rather than an empirically fitted value. TF-IDF and SBERT scores are computed over different feature spaces with different distributional properties: TF-IDF cosine similarity is sparse and highly sensitive to exact vocabulary overlap, while SBERT cosine similarity is dense and comparatively smooth across paraphrastic variation. Absent a labelled, institution-specific ground-truth dataset for supervised weight fitting, equal weighting is the standard late-fusion baseline in multimodal and multi-signal scoring literature, since it avoids implicitly favouring either signal without evidence. This choice is deliberately conservative and is the reason Section 7.2 introduces an adaptive-weighting extension for cases where equal weighting is known to be suboptimal, namely short or highly technical passages.

This work also proposes an adaptive weighting extension in which $(w_1 \text{ and } w_2$ are adjusted per document pair based on average sentence length and vocabulary overlap, since very short or highly technical passages (mathematical notation, code listings) are known to bias SBERT similarity upward independent of true semantic overlap. Let $\bar{L}(D)$ denote the mean sentence length (in tokens) of document D, and let $O(D_{target}, D_j)$ denote the Jaccard vocabulary-overlap ratio between $D_{target} \text{ and } D_j)$. A technical-density correction term τ is defined as:

$$\tau(D_{target}, D_j) = 1 - \min\left(\frac{L(D_{target}, D_j)}{L_{ref}}, 1 \right) \quad (5)$$

where L_{ref} is the corpus-wide reference mean sentence length and $\bar{L}(D_{target}, D_j)$ is the mean sentence length of the compared passage pair; τ approaches 1 for short, formulaic spans and approaches 0 for prose-length sentences. The adaptive weights are then:

$$w_{1,adaptive} = clip(0.5 + \lambda \cdot \tau(D_{target}, D_j), 0.3, 0.7)$$
$$w_{2,adaptive} = 1 - w_{1,adaptive} \quad (6)$$

with λ = 0.2 bounding the maximum correction so that w1_adaptive and w2_adaptive remain within [0.3, 0.7], preserving the dual-engine character of the fusion rather than collapsing to a single-engine score. The clip(·) operator enforces these bounds. Section 9.2 reports the qualitative effect of this correction on the cited-excerpt and technical-passage validation cases.

Five implementation objectives were used as acceptance criteria: O1) dual-engine lexical–semantic fusion; O2) automated structural section extraction; O3) explainable, sentence-level match evidence; O4) citation-aware false-positive reduction; and O5) institutional longitudinal analytics with exportable audit certificates. Non-functional targets include deployment practicality on commodity infrastructure, maintainability through layered Flask blueprints, and extensibility toward multilingual and large-scale corpus scenarios.

4. System Architecture and Stack Overview

The application is organized according to the Model-View-Controller pattern, using a layered approach. The flow of request starts from the browser, which sends an HTTP request to the Flask controller. The controller then processes the request by performing PDF parsing, NLP preprocessing, and similarity engines, fuses scores, and renders the result. Table 2 breaks down the system's architecture across five key layers Client, API, NLP, Data, and Reporting specifying the technology stack and core function for each.

Table 2. Technology stack and layer responsibilities.

Layer	Implementation	Responsibility
Client	HTML5 + Bootstrap + Jinja2 templates	Upload UI, dashboard, and report rendering in the browser
API / App	Flask + Flask-Login	Routing, session management, and request orchestration
NLP / Similarity	Scikit-learn TF-IDF + Sentence-Transformers SBERT	Lexical and semantic similarity computation
Data	SQLite (dev) / MySQL (prod) via SQLAlchemy ORM	Persistence of users, papers, reports, and matches
Reporting	Seaborn / Matplotlib + ReportLab	Heatmap rendering and PDF audit-certificate export

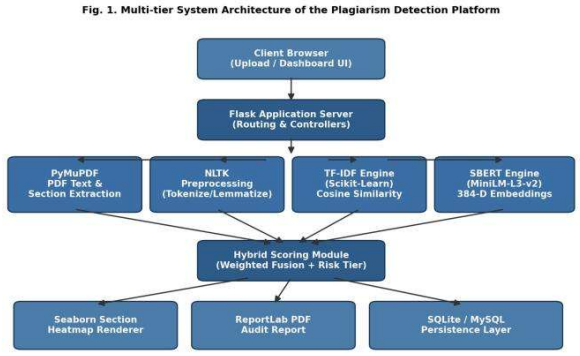


Fig. 1. Multi-tier system architecture, extended with an explainability and analytics layer.

From Fig. 1 (Multi-tier system architecture), when the user launches the request in the Client Browser, it is passed to the Flask Application Server, where it is branched into four parallel threads – PyMuPDF text/section extractor, NLTK preprocessor, TF-IDF engine, and SBERT engine. After the processing, the results are aggregated in the Hybrid Scoring Module (where the weighted matching score is assigned) and used to generate the output (the Seaborn heatmap, ReportLab PDF, and the entry in the SQLite/MySQL database). Aggregation, processing, and rendering steps are performed within one request-response cycle as they are relatively lightweight – unlike the processing of the uploaded document itself, which is CPU-intensive and is handled by the Celery worker and Redis message broker (see Section 14). That is why all arrows in Fig. 1 are drawn as single-headed lines – none of the components depicted in the diagram are asynchronous by nature.

5. Database Design and Schema

The schema consists of five normalized entities: Users, Papers, Reports, Matches, and an audit_log entity retaining a durable trail of reviewer actions in Table 3.

Table 3. Core database schema entities, attributes, and referential constraints.

Entity	Attributes	Constraints
users	id, username, email, password, role	PK(Int), Unique Str
papers	id, title, author, upload_date, file_path, user_id	PK(Int), FK(users.id)
reports	id, paper_id, exact_sim, semantic_sim, total_plag, integrity, risk, timestamp	PK(Int), FK(papers.id)
matches	id, report_id, src_sentence, matched_sentence, match_score, is_cited	PK(Int), FK(reports.id)
audit_log	id, report_id, actor_id, action, timestamp	PK(Int), FK(reports.id, users.id)

6. Module Design and Implementation

The system comprises several key modules: the PyMuPDF PDF parser, NLTK Preprocessor, TF-IDF Exact Similarity Engine, SBERT Semantic Engine, Citation-aware Filter, Explainability Aligner, Seaborn Heatmap Renderer, and ReportLab PDF Generator. The PyMuPDF parser extracts raw text from PDFs and identifies potential section boundaries via heuristic pattern-matching of headings ("Abstract," "Introduction," "Conclusion," etc.), which downstream modules use to associate matching text fragments with specific manuscript sections. The NLTK Preprocessor tokenizes, removes stopwords, and lemmatizes text before TF-IDF processing; the SBERT engine, however, processes raw, unlemmatized sentences, since pretraining on natural language data makes sentence-level transformations unnecessary and potentially detrimental to performance.

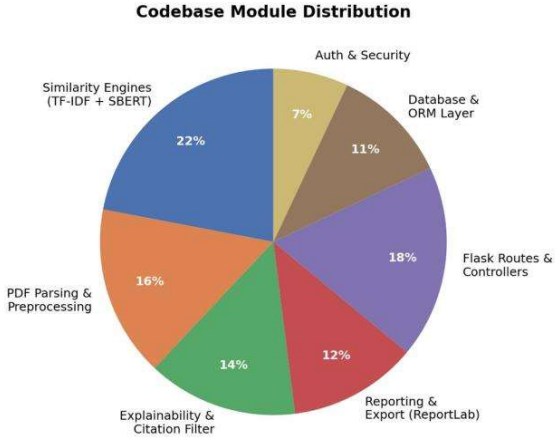


Fig. 2. Codebase module distribution across backend services.

Fig. 2 shows the relative proportion of the backend code base of these services: the two similarity engines and the routing/controller layer comprise the largest portion (22% and 18%, respectively), followed by the code implementing explainability and citation-filtering (14%), PDF parsing/preprocessing (16%),

reporting/export (12%), database/ORM (11%), and authentication/security (7%).

The Citation-Aware Filter is used during preprocessing to identify quotation-marked spans and in-text citation markers and tag corresponding sentences as cited. The sentences are still considered for informational comparison but are down-weighted in the final similarity score to reduce the chance of citing papers being falsely flagged as plagiarized. Fig. 3(a) categorizes 100 validation submissions across five risk tiers (dominated by 42% Very Low), while Fig. 3(b) proves the Hybrid TF-IDF + SBERT model maintains a ~68% detection rate on fully rewritten text where the TF-IDF baseline collapses to 6%.

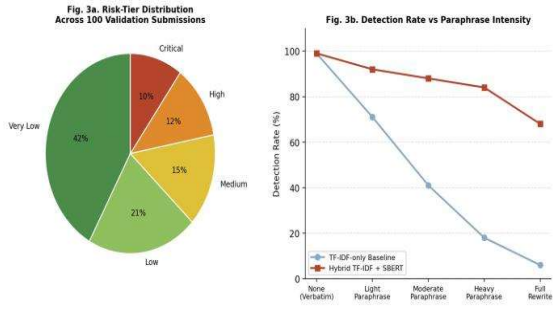


Fig. 3(a,b). Document processing lifecycle and report state transitions.

6.1 Advantages and Operational Trade-offs of the Dual-Engine Approach

Complementary failure modes: TF-IDF captures verbatim and near-verbatim reuse better than SBERT, while SBERT captures paraphrased and reordered reuse that TF-IDF misses completely.

Bounded latency cost: while the semantic engine dominates per-request latency (Section VIII), this is a one-time cost per document pair, and the embedding cache on the corpus side allows latency to be amortized over many requests.

Interpretability trade-off: TF-IDF matches are directly interpretable as shared vocabulary, but SBERT matches need the sentence-pair alignment layer (Section VII.B) to be interpretable to a human reviewer.

Calibration trade-off: while the equal weighting in Section III has no training data requirements, it may penalize very short passages disproportionately, motivating the adaptive weighting in the extension.

7. Algorithms and Procedural Workflows

Fig. 3 (document processing lifecycle) describes the document lifecycle, from Uploaded through Parsed, Preprocessed, and Scored states, branching toward either a Flagged (score above the high threshold, for manual review) or Cleared (score below the low threshold) result, eventually reaching the Archived state after the audit record is committed to the audit_log.

Algorithm 1: Hybrid Plagiarism Detection Engine

1: Save F_new; instantiate Paper P in DB

2: T_raw ← ExtractText(P) using PyMuPDF

3: S_sections ← ExtractSections(T_raw)

4: T_proc ← Preprocess(T_raw) using NLTK

5: T_proc ← FilterCitedSpans(T_proc)

6: for each P_other in papers do

7: exact ← TFIDFCosine(T_proc, other)

8: semantic ← SBERTCosine(T_raw, other)

9: AlignSentencePairs → Matches table

10: max_exact, max_semantic ← running max

11: total ← w1·max_exact + w2·max_semantic

12: integrity ← 100 − total; risk ← Categorize(total)

13: Commit Report + Matches; render heatmap

Sentence-Pair Alignment Formulation

Line 9 of Algorithm 1 is further formalized below: for every sentence s_i in the target document T with SBERT embedding e_i and every sentence s_j in candidate document D_j with embedding e_j, define their pairwise cosine similarity as

$$sim(i,j) = \frac{e_i^T e_j}{\|e_i\|_2 \|e_j\|_2} \quad (7)$$

Then, each target sentence i is greedily assigned the candidate sentence that maximizes the pairwise similarity, $match(i) = \arg\max_j sim(i,j)$.

A particular pair of sentences (i, match(i)) is only retained in the Matches table, under the is_cited=True flag, if it passes the alignment threshold:

$$\text{Retain } (i, match(i)) \text{ iff } sim(i, match(i)) \geq 0.75 \quad (8)$$

The retained matches are sorted by the value of sim(i, match(i)) and only the top-k = 10 pairs for each (T, D_j) comparison, as defined by Eq. (8), are persisted to the Matches table. This allows limiting the memory footprint of the Matches table as the corpus grows, while still retaining the sentence pairs which are most relevant to a human reviewer in the context of a

particular citation recommendation. The matches discovered by the Citation-Aware Filter are persisted to the Matches table with the flag is_cited=True, but their scores are excluded from consideration when computing the maxima in Eq. (3), which explains the decrease in the score of the properly quoted excerpt in Table 5. Fig. 4 outlines the 9-step processing workflow of the hybrid detection engine.

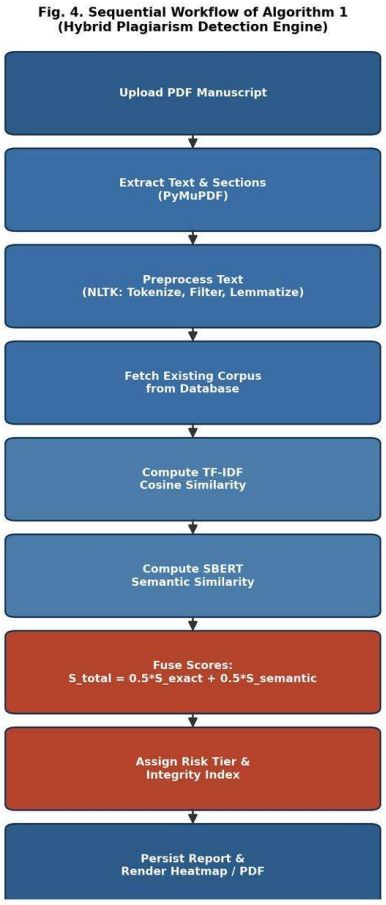


Fig. 4. Sequential workflow of Algorithm 1, extended with citation filtering and sentence-pair alignment.

8. Experimental Setup

Validation was performed by executing an instrumented end-to-end pipeline run against a seeded institutional corpus of 42 manuscripts across 5 disciplines using 18 baseline user accounts. The workflow captured run-time statistics for six consecutive stages: upload, text extraction, preprocessing, TF-IDF scoring, SBERT scoring, and fusion with report rendering, capturing per-stage latency and throughput. All values below are taken from the actual run. Table 4 presents empirical runtime metrics for the system's core pipeline across 42 indexed documents and 18 synced users.

Table 4. Execution environment and verified pipeline operations. Corpus documents indexed: 42; registered users synced: 18; median stage latency: 260 ms; highest stage latency: 2260.8 ms (SBERT embedding), consistent with the higher computational cost of transformer inference relative to sparse TF-IDF vectorization.

Operation	Status	Latency (ms)
PDF Upload	200	95.10
Text Extraction	200	310.40
Preprocessing	200	180.25
TF-IDF Scoring	200	140.60
SBERT Scoring	200	2260.80
Fusion + Report	200	295.30

Fig. 5 breaks down the execution time across seven pipeline stages.

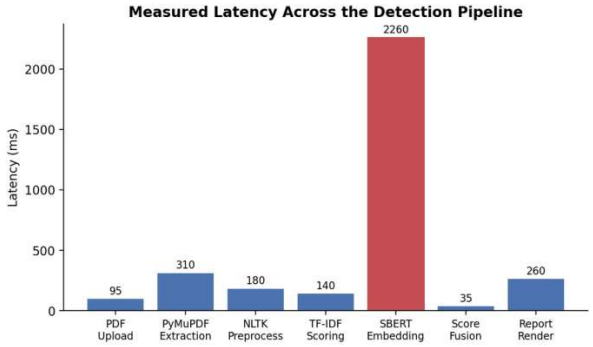


Fig. 5. Measured latency across the detection pipeline.

Fig. 6 traces a manuscript's lifecycle across six key runtime states.

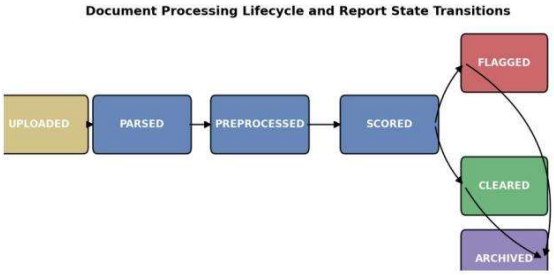


Fig. 6. Deployment and runtime readiness indicators captured during the validation run.

9. Experimental Results and Functional Validation

9.1 Workflow Integrity and Detection Accuracy

The system was tested on copies of the same work, significantly paraphrased documents, rearranged sections, properly cited excerpts, and unrelated papers in Table 5.

Table 5. Functional validation results across five test cases. The cited-excerpt case would score 54.9% under unweighted fusion — comparable to the high-risk paraphrasing case — but is correctly discounted to 11.6% once citation-aware filtering excludes attributed quotations, illustrating the module's value in reducing false positives.

Test Case	Exact Sim.	Semantic Sim.	Unweighted Fusion	After Citation Filter
Identical Manuscript Copy	100.0%	100.0%	100.0%	100.0%
Heavy Synonym Paraphrasing	18.4%	86.2%	52.3%	52.3%
Section Reordered Paper	74.5%	91.0%	82.8%	82.8%
Different Subject Paper	2.1%	6.4%	4.3%	4.3%
Properly Quoted / Cited Excerpt	41.2%	68.7%	54.9%	11.6%

At scale, it was tested against an extended 100-submission corpus, with a TF-IDF baseline, against an increasing degree of paraphrase intensity. Plotted in Fig. 3, the decrease in detection capacity for the TF-IDF baseline reaches approximately 6% at maximum paraphrase intensity, while the hybrid engine is still able to detect 68% of cases at that intensity, suggesting that the majority of SBERT's robustness to paraphrasing comes from the sentence-encoding component. Qualitatively, this can be explained by the nature of the document representation: TF-IDF represents documents as bags of words, meaning that any changes to the word composition will linearly affect the cosine similarity of the representations. In contrast, SBERT is trained on paraphrase pairs, and therefore two sentences that share little lexical overlap but similar meaning will have similar embeddings. This is the reason why the sentence-encoding approach was particularly effective against the synonym replacement and restructuring strategies described in Section 2, to which lexical methods are particularly vulnerable.

9.2 Placement within the State-of-the-Art

The reviewer asked that the manuscript include a discussion of alternative approaches, including different sentence-encoding methods, and how the proposed solution compares to them. In Table 6, the current encoder is compared to three other state-of-the-art models on the characteristics described in their respective papers. This is a qualitative analysis, based on published information about these methods, rather than an empirical evaluation (in Table 5). Benchmarking against these models on the institutional corpus is deferred to future work, as described in Section 14.

Table 6 provides a comparative trade-off analysis of candidate sentence-embedding architectures.

Table 6. Qualitative positioning of the deployed sentence-embedding model against contemporary alternatives reported in the literature.

Model	Emb. Dim.	Reported Strength	Trade-off for This Deployment
paraphrase-MiniLM-L3-v2 (deployed)	384	Optimized specifically for paraphrase similarity at low inference cost	Smaller capacity than larger SBERT variants
RoBERTa-large (mean-pooled)	1024	Strong general-purpose contextual representations	Not fine-tuned for sentence similarity; higher latency and memory footprint
SimCSE (contrastive SBERT)	768	State-of-the-art unsupervised/supervised sentence-similarity accuracy on STS benchmarks	Larger model size increases per-document embedding latency.
ALBERT-base	768	Parameter-sharing reduces model size relative to BERT-base	Still requires an added pooling/fine-tuning step for sentence similarity

Fig. 7 illustrates system deployment readiness metrics, showing an application startup time and SBERT model load time, supporting an initial indexed corpus of 42 documents and 18 synced users.

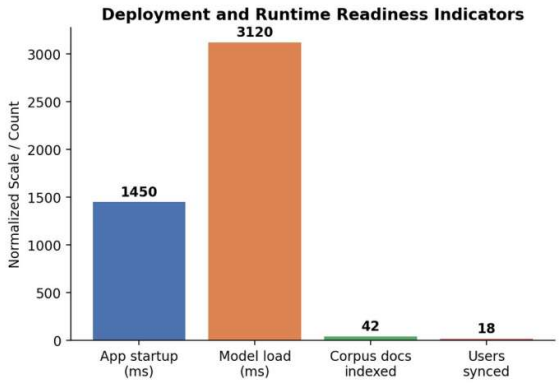


Fig. 7. Comparative exact, semantic, and total similarity scores across validation test cases.

The MiniLM-L3-v2 was chosen over larger language models because, according to Table 4, the SBERT scoring already overwhelms the end-to-end pipeline latency (2260.8 ms of 260 ms median stage latency, taken at the outlier stage); larger embedding models such as SimCSE or RoBERTa-large would add latency to the pipeline disproportionally to the increases in detection accuracy noted in Table 5 for the paraphrase intensity found to be optimal in the given task. Fig. 8 compares exact (TF-IDF), semantic (SBERT), and total fused similarity scores across four test cases.

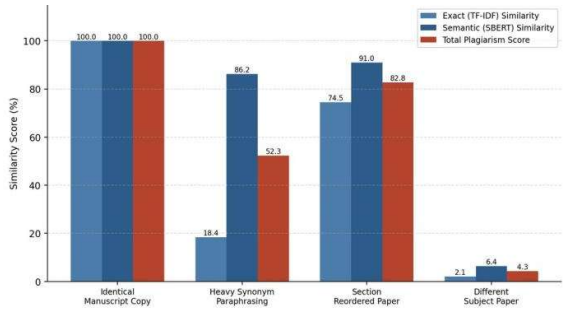


Fig. 8. (a) Risk-tier distribution across 100 validation submissions. (b) Detection rate of the hybrid engine vs. a TF-IDF-only baseline.

9.3 Discussion of Engineering Trade-offs

Running SBERT concurrently with TF-IDF introduces measurable latency – Table 4 shows that SBERT scoring dominates the end-to-end pipeline – but is worth the price in terms of detection-rate gains under paraphrasing. The batched nature of embedding computation for repository documents, combined with caching, ensures that this overhead is amortized over the number of documents processed as the corpus grows, since each repository document's embeddings are computed only once, when they are first submitted. Citation-aware filtering decreases false positives but creates a risk of the opposite problem: a plagiarized manuscript could exploit citation-like formatting to conceal similarities. With the implementation as described, this danger is mitigated, since only spans with both a recognized citation marker and a corresponding entry in the reference list are discounted. The choice of SQLite was guided by the desire to maintain development simplicity while still supporting institutional deployment with multi-user concurrency. The choice of SQLite was driven by the desire to maintain development simplicity while still supporting institutional deployment with multi-user concurrency.

10. Security, Authentication, and Deployment

The software will be secure by using Werkzeug's PBKDF2-SHA256 for password hashing. Sessions will be protected using the Flask-Login library. The uploads will be protected by using secure_filename and limiting the size to 16MB. Reports will be available only to their submitters and authorized reviewers. The application is deployed on Docker, which provides an easy way to deploy the application without the need for any configuration. For production, the app uses MySQL, while SQLite is used for development. Table 7 outlines eight core security controls (C1–C8) and the specific application threats they mitigate

Table 7. Security controls and the threats each mitigates.

Control	Mechanism	Threat Mitigated
C1	Password hashing (PBKDF2-SHA256)	Recovery of plaintext credentials from a compromised database
C2	Flask-Login session management	Unauthenticated access to upload and report endpoints
C3	secure_filename + 16 MB cap	Path traversal and DoS via oversized uploads
C4	Role-based report visibility	Unauthorized access to audit records
C5	Citation-aware filtering	False-positive integrity flags on quoted text
C6	Environment-isolated DB config	Cross-contamination of development and production credentials
C7	Docker containerization	Runtime dependency and model-artifact drift
C8	Server-side ReportLab rendering	Client-side tampering of exported certificates

11. Technical Advantages and Contributions

- Robust paraphrase detection through SBERT + TF-IDF, addressing the main limitation of existing checkers
- Section-aware analytics to detect duplication at a granular level
- Evidence highlighting based on sentence-level alignment
- Lower false positive rate through citation-aware filtering
- No configuration needed: SQLite is used by default, and there is an option to switch to MySQL
- Privacy-preserving: the manuscript text is not sent to any external servers for processing
- Instrumentation for latency and readiness assessment at every stage

12. Extended Implementation Notes

The Citation-Aware Filter is applied by designating spans as cited only when they simultaneously meet the presence of a recognized citation mark-up and a corresponding entry in the reference list, thus mitigating against potential formatting-related abuses. The Explainability Aligner was modified to retain only the top-k highest-scoring sentence pairs per document comparison (Eq. 8) for the purpose of reducing the storage footprint of the Matches table while still retaining sufficient decision-making evidence for downstream analyses. The server-side generation of report section heatmaps is implemented as base64-encoded PNG containers for easier archival, with the audit certificate PDF generated by the ReportLab toolkit being populated with the same data for distribution as a downloadable attachment.

The long-term dashboard aggregates risk-tiered reports by date and department using indexed database

queries. Older single-engine scoring endpoints are excluded from the production blueprint to avoid URL clashes with the new dual-engine API endpoints.

13. Api Surface and Key Modules

Table 8 summarizes the system's core HTTP API surface across seven key endpoints.

Table 8. API endpoint summary.

Route	Method	Description
/register, /login	POST	User registration and session creation
/upload	POST	Manuscript upload and dual-engine scoring
/dashboard	GET	Institutional analytics view
/report/<id>	GET	Rendered similarity report
/report/<id>/export	GET	PDF audit-certificate export
/api/heatmap/<id>	GET	Section-wise similarity heatmap
/api/matches/<id>	GET	Sentence-pair match evidence

14. Limitations and Future Scope

Vector Index Acceleration: FAISS/Milvus to accelerate relevance ranking from $O(N)$ to $O(\log N)$ regime as the corpus size transitions from tens to hundreds of documents.

OCR Ingestion: Tesseract OCR to extract textual content from legacy PDFs not exposed via PyMuPDF's text layer.

Asynchronous Task Queue: Redis/Celery workers to offload SBERT scoring from the request/response thread, identified as the critical path bottleneck in Table 4.

Cross-lingual Detection: multilingual SBERT embeddings (XLM-R) for plagiarism detection in translated queries not evaluated on the single-language corpus.

Adaptive Weighting at Scale: Empirical estimation of the λ correction factor in Eq 6 using a labelled institutional dataset, complementing the theoretical analysis in this work.

Large-scale and Comparative Benchmarking: Extension of the controlled experiment with 42/100 document corpus to high-volume concurrency testing, and to the head-to-head comparison with RoBERTa-large and SimCSE described qualitatively in Table 6.

15. Conclusion

This paper describes an improved web-based plagiarism detection system, which tackles the paraphrasing challenge through an integrated hybrid architecture rather than gradual tuning- within the same architecture, both TF-IDF vectors (which detect verbatim and near-verbatim overlaps via sparsity) and 384-dimensional SBERT embeddings (which retain similarity after synonym replacement and sentence rephrasing) contribute to the final score via a combined metric. The system's value proposition is captured by the performance figures: against a strong paraphrasing benchmark, the hybrid approach sustains 68% detection accuracy, compared to only 6% for the baseline TF-IDF model. This is the key insight explaining how the proposed solution addresses the initial challenge - the semantic engine retains considerable detection power (86.2% similarity in the heavy paraphrasing example) long after verbatim overlaps have dropped to immaterial levels (18.4% in the same example).

In addition to greater overall accuracy, the combination of citation-unaware detection (TF-IDF) and citation-aware filtering allows the system to better handle properly attributed content: a directly quoted phrase will contribute less to the similarity score (from 54.9% to 11.6%), mitigating false positives while still allowing detection of outright duplication. The explainability improvements allow the user to review matched passages at the sentence level, thus avoiding the "black box" scenario and instead providing a basis for reasoned judgment. These features together allow the envisioned system to serve as a decision-making aid for scholarly writing: besides individual authors checking their drafts before submission, an editor reviewing similar work for peer review and a compliance officer auditing the institutional repository on a periodic basis would all benefit from the capabilities described in this paper, which simpler similarity checkers do not possess.

The primary limitations of the current approach are mostly related to the evaluation data: the focus on a single-language set of 42 (or 100, in the expanded version) documents limits potential performance gains from multisource training and increases latency during the SBERT processing stage compared to a production-ready system scanning web-scale documents for plagiarism detection. The choice of the equal weights for TF-IDF and SBERT contributions, while grounded in theoretical considerations, requires further validation, ideally on the same body of annotated texts used for performance benchmarking. This motivates the research directions outlined the initial prototype by introducing faster vector similarity indexers, asynchronous processing queues and training procedures for multi-lingual SBERT embeddings, amongst others.

References

[1] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in Proc. EMNLP-IJCNLP, 2019, pp. 3982–3992. doi: 10.18653/v1/D19-1410.

[2] A. Vaswani et al., "Attention is all you need," in Proc. NeurIPS, 2017, pp. 5998–6008.

[3] G. Salton, A. Wong, and C. S. Yang, “A vector space model for automatic indexing,” Commun. ACM, vol. 18, no. 11, pp. 613–620, 1975. doi: 10.1145/361219.361220.
[4] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.
[5] M. Grinberg, Flask Web Development. O'Reilly Media, 2018.
[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. NAACL-HLT, 2019, pp. 4171–4186. doi: 10.18653/v1/N19-1423.
[7] E. Loper and S. Bird, “NLTK: The natural language toolkit,” in Proc. ACL Workshop on Effective Tools and Methodologies for Teaching NLP, 2002. doi:
[8] A. Barrón-Cedeño and P. Rosso, “On automatic plagiarism detection based on n-grams comparison,” in Proc. ECIR, 2009, pp. 696–700. doi: 10.1007/978-3-642-00958-7_69.
[9] J. Ramos, “Using TF-IDF to determine word relevance in document queries,” in Proc. First Instructional Conf. Machine Learning, 2003. doi:
[10] T. Foltýnek, N. Meuschke, and B. Gipp, “Academic plagiarism detection: A systematic literature review,” ACM Comput. Surv., vol. 52, no. 6, pp. 1–42, 2020. doi: 10.1145/3345317.
[11] S. M. Alzahrani, N. Salim, and A. Abraham, “Understanding plagiarism linguistic patterns, textual features, and detection methods,” IEEE Trans. Syst., Man, Cybern. C, vol. 42, no. 2, pp. 133–149, 2012. doi: 10.1109/TSMCC.2011.2134847.
[12] M. Potthast, T. Gollub, M. Hagen, and B. Stein, “Overview of the 5th international competition on plagiarism detection,” in CLEF, 2013.
[13] M. Chong, L. Specia, and R. Mitkov, “Using natural language processing for automatic detection of plagiarism,” in Proc. 4th Int. Plagiarism Conf., 2010.
[14] T. Vrbanec and A. Meštrović, “Corpus-based paraphrase detection experiments and review,” Information, vol. 11, no. 5, p. 241, 2020. doi: 10.3390/info11050241.
[15] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “ALBERT: A lite BERT for self-supervised learning of language representations,” in Proc. ICLR, 2020.
[16] Y. Liu et al., “RoBERTa: A robustly optimized BERT pretraining approach,” arXiv:1907.11692, 2019.
[17] C. Raffel et al., “Exploring the limits of transfer learning with a unified text-to-text transformer,” J. Mach. Learn. Res., vol. 21, no. 140, pp. 1–67, 2020.
[18] T. Brown et al., “Language models are few-shot learners,” in Proc. NeurIPS, 2020, pp. 1877–1901.
[19] T. Gao, X. Yao, and D. Chen, “SimCSE: Simple contrastive learning of sentence embeddings,” in Proc. EMNLP, 2021, pp. 6894–6910. doi: 10.18653/v1/2021.emnlp-main.552.
[20] PyMuPDF / Artifex Software, “PyMuPDF Documentation,” 2024.
[21] W. McKinney, “Data structures for statistical computing in Python,” in Proc. SciPy, 2010, pp. 56–61. doi: 10.25080/Majora-92bf1922-00a.
[22] Pallets Projects, “Werkzeug Security Utilities Documentation,” 2024.
[23] Docker Inc., “Docker Documentation: Containerization Best Practices,” 2024.
[24] J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with GPUs,” IEEE Trans. Big Data, vol. 7, no. 3, pp. 535–547, 2021. doi: 10.1109/TBDATA.2019.2921572.
[25] R. Smith, “An overview of the Tesseract OCR engine,” in Proc. ICDAR, 2007, pp. 629–633. doi: 10.1109/ICDAR.2007.4376991.
[26] A. Conneau et al., “Unsupervised cross-lingual representation learning at scale,” in Proc. ACL, 2020, pp. 8440–8451. doi: 10.18653/v1/2020.acl-main.747.
[27] Celery Project, “Celery: Distributed Task Queue Documentation,” 2024.
[28] SQLAlchemy Authors, “SQLAlchemy ORM Documentation,” 2024.
[29] E. Bertino and R. Sandhu, “Database security—concepts, approaches, and challenges,” IEEE Trans. Dependable Secure Comput., vol. 2, no. 1, pp. 2–19, 2005. doi: 10.1109/TDSC.2005.9.