
Optimizing Scientific Document Similarity Detection Using Jaro-Winkler and Machine Learning

Firman Santosa¹, Detri Amelia Chandra², Sri Wahyudi³, Ridwan⁴, Rahmat⁵

^{1, 4, 5} Computer Science, Department of Computer Science, Universitas Rokania, Indonesia

^{2, 3} Information Technology, Department of Computer Science, Universitas Rokania, Indonesia

E-mail: firman.snts@gmail.com, detriamelia@rokania.ac.id, sriwahyudi.sl@gmail.com,

ridwan.psp2018@gmail.com, 2305016@rokania.ac.id

Correspondence Email: firman.snts@gmail.com

Submitted: 14-04-2026; **Accepted:** 12-05-2026; **Published:** 15-05-2026;

DOI: <https://doi.org/10.20527/cetj.v6i1.18697>

Abstrak

Penelitian ini bertujuan mengembangkan dan mengevaluasi prototipe sistem deteksi kesamaan dokumen ilmiah berbasis algoritma Jaro-Winkler yang terintegrasi dengan sinkronisasi metadata Open Journal Systems (OJS) dan disiapkan sebagai fondasi model hybrid berbasis machine learning. Penelitian menggunakan pendekatan terapan dengan orientasi pengembangan sistem melalui tahapan analisis kebutuhan, perancangan arsitektur, implementasi, pengujian fungsional, dan analisis keluaran similaritas. Data uji terdiri atas 162 artikel referensi hasil sinkronisasi OJS dan 30 dokumen uji berupa judul dan abstrak; 27 pengujian selesai diproses dan 3 berada pada status pending saat dokumentasi sistem dilakukan. Hasil pengujian black-box menunjukkan bahwa fungsi utama, mulai dari input dokumen, pra-pemrosesan teks, penghapusan ignored words, tokenisasi, perhitungan Jaro-Winkler, agregasi skor per artikel, penyimpanan hasil, riwayat pemeriksaan, hingga pencetakan laporan, berjalan sesuai skenario. Uji similaritas menunjukkan sistem mampu menghasilkan skor terukur pada rentang 0,7106–0,8297 untuk sampel dokumen yang terdokumentasi, termasuk hasil akhir 0,8097 pada salah satu pengujian. Temuan ini menunjukkan bahwa JARWIN dapat digunakan sebagai alat skrining awal yang cepat, terdokumentasi, dan dapat ditelusuri, tetapi belum menggantikan penilaian editorial karena masih berfokus pada kemiripan leksikal. Struktur data hasil pemeriksaan dapat dimanfaatkan sebagai dataset awal untuk pengembangan model hybrid machine learning yang lebih peka terhadap parafrasa dan kemiripan semantik.

Kata Kunci: integrasi OJS; Jaro-Winkler; kemiripan dokumen; machine learning; similarity checker

Abstract

This study aims to develop and evaluate a scientific document similarity detection prototype based on the Jaro-Winkler algorithm, integrated with Open Journal Systems (OJS) metadata synchronization, and prepared as a foundation for future hybrid machine-learning models. This applied research followed a system-development orientation consisting of requirements analysis, architecture design, implementation, functional testing, and similarity-output analysis. The testing data comprised 162 OJS-synchronized reference articles and 30 test submissions containing titles and abstracts; 27 submissions were fully processed and 3 remained pending at the time of system documentation. Black-box testing showed that the main functions, including document input, text preprocessing, ignored-word removal, tokenization, Jaro-Winkler computation, article-level score aggregation, result storage, checking history, and printable reporting, worked according to the expected scenarios. The similarity-output test produced measurable scores ranging from 0.7106 to 0.8297 in the documented samples, including a final score of 0.8097 in one submission. These findings indicate that JARWIN can serve as a fast, documented, and traceable early-screening tool, although it does not replace editorial judgment because it still focuses on lexical similarity. The stored similarity records can be used as an initial dataset for developing hybrid machine-learning models that are more sensitive to paraphrase and semantic similarity.

Keywords: document similarity; Jaro-Winkler; machine learning; OJS integration; similarity checker

How to cite: Santosa, F., Chandra, D. A., Wahyudi, S., Ridwan, & Rahmat. (2026). Optimizing scientific document similarity detection using Jaro-Winkler and machine learning. *Computing and Education Technology Journal (CETJ)*, 6(1), 225–236. <https://doi.org/10.20527/cetj.v6i1.18697>

INTRODUCTION

Scientific publishing increasingly depends on digital repositories, institutional e-journals, and Open Journal Systems (OJS)-based workflows. This transformation improves publication access, but it also increases the need for rapid and traceable screening of manuscript similarity before editorial decisions are made. In journal management, early detection is important because similarities in titles and abstracts often become the first indicator of topic duplication, potential text reuse, or insufficient novelty. Without an initial screening mechanism, editors and reviewers must rely heavily on manual inspection, which is time-consuming, subjective, and difficult to audit consistently.

Previous studies show that lexical and string-based similarity techniques remain useful for identifying surface-level text closeness in large datasets, particularly when they are combined with preprocessing procedures such as case folding, punctuation cleaning, tokenization, and stopword filtering (Baloi et al., 2024; Novyantika & Isa, 2023). The Jaro-Winkler algorithm is especially relevant for short scientific texts because it gives higher weight to common prefixes and is sensitive to minor character-level variations (Jaro, 1989; Winkler, 1990). These characteristics make it suitable for comparing article titles, keywords, and concise abstract phrases that frequently contain repeated terminology. In a closely related Indonesian journal study, Santosa (2022) also applied the Jaro-Winkler algorithm to identify similarity levels in scientific article abstracts, supporting its relevance as a lightweight method for early academic text screening.

However, similarity detection based only on lexical matching has limitations. It may identify near-duplicate phrases effectively, but it can be less robust when the same idea is rewritten using different terms, sentence structures, or semantic expressions. For this reason, recent research increasingly explores hybrid approaches that combine string similarity, semantic features, and machine learning to improve the reliability of text comparison (Ahmad et al., 2022; Rudwan & Fonou-Dombeu, 2023; Reimers & Gurevych, 2019). Machine-learning-based approaches can support semantic representation, but they require structured datasets, stable preprocessing, stored comparison results, and a clear evaluation pipeline before they can be deployed operationally.

Although string metrics such as Jaro-Winkler, TF-IDF, and edit-distance variants have been applied in duplicate detection and short-text comparison, most prior works still emphasize metric performance or preprocessing quality rather than deployable workflows for journal operations. Santosa (2022) demonstrated the usefulness of Jaro-Winkler for identifying similarity levels in scientific article abstracts, while Baloi et al. (2024) and Pitchandi and Balakrishnan (2023) emphasized scalable string-similarity and clustering approaches. However, these studies do not fully address the operational needs of OJS-based journals: automated ingestion of reference articles, persistent token-level evidence, aggregated scoring, administrative history, and report generation within one auditable system. Machine-learning and semantic approaches such as BERT and Sentence-BERT provide richer representations (Devlin et al., 2019; Reimers & Gurevych, 2019), but they require structured datasets and reliable baseline outputs before supervised or hybrid models can be trained. Therefore, the research gap addressed in this study is the absence of an institutional prototype that bridges lexical similarity computation, OJS reference-data management, and data readiness for future machine-learning optimization.

The novelty of this research lies in the development of JARWIN (Jaro-Winkler Similarity Checker), a web-based prototype that integrates a lightweight Jaro-Winkler similarity engine, OJS-based dataset synchronization, token-level result storage, aggregated final similarity indicators, and a modular architecture that can be extended toward machine-learning-based hybrid models. This combination provides both practical value for journal managers and technical value for future similarity-model development.

Based on this gap and novelty, this study has two interconnected objectives. First, it develops JARWIN as a web-based scientific document similarity screening prototype that integrates text preprocessing, Jaro-Winkler computation, OJS-based reference-data synchronization, result storage, and administrative reporting. Second, it evaluates the prototype through black-box functional testing and similarity-output analysis to determine whether the system can produce traceable similarity indicators for scientific titles and abstracts and provide structured data for future hybrid machine-learning models.

METHOD

This research is an applied study with a system-development orientation. The design follows a research and development approach in which the main output is an executable prototype rather than only a conceptual model. The study focuses on developing JARWIN as an early-stage scientific document similarity screening system for journal management environments.

The study was conducted from March to December 2025 at Universitas Rokania. The research subjects were the operational actors represented in the system workflow, namely journal administrators, checker operators, and system administrators. The research objects were scientific document metadata, particularly article titles and abstracts, together with reference journal articles synchronized from Open Journal Systems (OJS). The empirical testing dataset consisted of 162 OJS-synchronized reference articles and 30 submitted document records containing titles and abstracts. The 30 submissions were used as prototype-testing cases rather than as a large-scale benchmark corpus. At the time of documentation, 27 submissions had completed similarity processing and 3 remained pending, which allowed the study to observe both completed and in-process workflow states. Technical objects also included database structures, preprocessing rules, similarity-score records, OJS configuration, and system logs produced during application testing.

The research instruments consist of a requirements checklist, document-observation sheet, preprocessing specification, database-design documentation, functional testing scenarios, and output-validation checklist. These instruments were used to ensure that the system requirements, implementation artefacts, and testing results could be traced to the research objectives.

Data collection was carried out through document analysis, observation of the prototype workflow, OJS metadata synchronization through REST API, and functional testing documentation. The collected data included article titles and abstracts, ignored-word configurations, tokenization outputs, Jaro-Winkler comparison scores, aggregated similarity values, black-box scenario results, checking-status records, and administrative report outputs.

The data analysis technique was descriptive, functional, and output-oriented. Descriptive analysis was used to interpret system requirements, workflow suitability, and implementation implications. Functional analysis was conducted using black-box testing to verify whether input handling, preprocessing, similarity computation, reference-data management, reporting, and baseline security features worked according to the expected outcomes. Similarity-output analysis was performed by examining token-level Jaro-Winkler scores, aggregating them into journal-level averages, identifying the highest final score as the system similarity indicator, and comparing representative output records against the interpretation categories used in the application interface.

The development procedure follows a staged workflow from requirements analysis to report finalization, as summarized in Table 1 and illustrated in Figure 1. Each stage produces concrete outputs so that the research process remains traceable from problem identification to prototype evaluation.

Table 1. Research Stages and Deliverables for JARWIN Development

Stage	Key activities	Deliverables
Requirements analysis	Review the proposal, research objects, and similarity-checker requirements	Defined system requirements and development scope
Data collection	Collect journal articles, metadata, and reference data structures	Initial journal-article dataset and technical system information
System design	Design the architecture, processing workflow, modules, and database	JARWIN application blueprint
Implementation	Implement the checker module, admin dashboard,	Executable prototype application

	similarity results, and OJS integration	
Testing and evaluation	Validate workflows, core functions, and security/development notes	Implementation findings and improvement recommendations
Report finalization	Integrate development outputs into the formal research report	Final research report

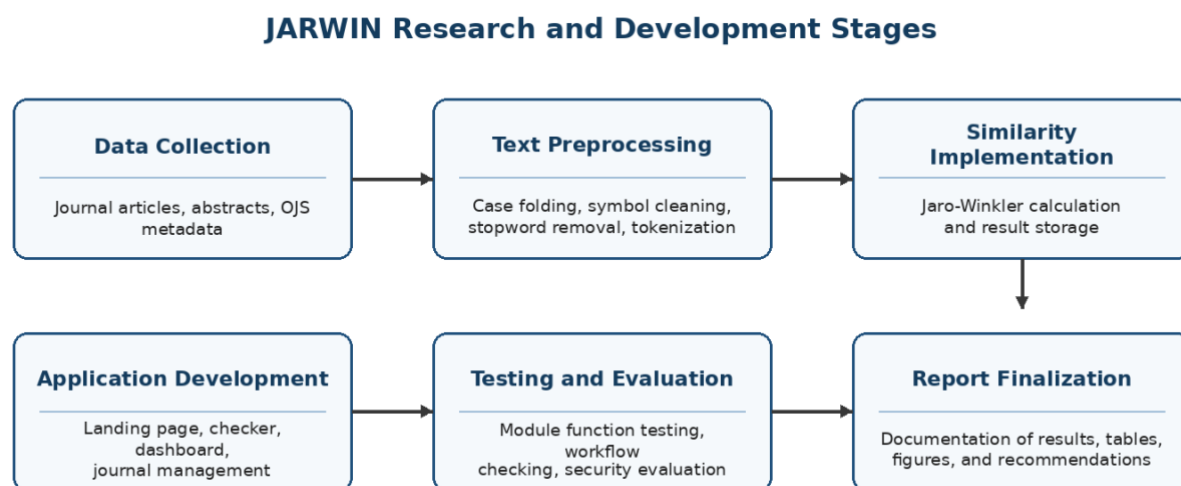


Figure 1. Research and Development Stages of JARWIN

The text-processing procedure consists of case folding, symbol and punctuation cleaning, ignored-word removal, and tokenization. After preprocessing, each token in a submitted title or abstract is compared with tokens in the reference corpus using the Jaro-Winkler metric. The system records token-level scores in detail, aggregates the scores into journal-level averages, and stores the highest aggregated value as the final similarity indicator for each submission. This procedure allows the result to be reviewed from the general score down to the specific token comparison.

Functional testing was performed using scenario-based black-box testing. Each scenario verified whether the submitted data could be stored, preprocessed, compared, summarized, and displayed correctly. The testing also examined administrative workflows, OJS synchronization readiness, report generation, and baseline security controls such as authentication, CAPTCHA, input sanitization, and access restriction. The test was considered successful when the observed output matched the expected output and the system produced a stored record that could be traced through the dashboard and report interface. The testing aspects and expected outcomes are summarized in Table 2.

Table 2. Functional testing aspects and expected outcomes

Testing aspect	Verification indicator	Expected outcome
Submission input	Title and abstract can be saved and processed	Data are stored correctly in the system
Preprocessing	Text is cleaned, normalized, and tokenized	Tokens are consistently prepared for comparison
Similarity engine	Jaro-Winkler produces token-level scores and journal-level summaries	Similarity can be computed and recorded

Journal data management	Reference articles can be managed and synchronized	Reference corpus remains available
Result reporting	Similarity summaries and detailed comparisons can be displayed	Users obtain traceable information
Baseline security	Authentication, input sanitization, and access restrictions are checked	System is ready for incremental hardening

RESULT AND DISCUSSION

System requirements and development implications

The requirements analysis identified that the system must provide fast screening for titles and abstracts, store results for re-checking, and support continuously updated reference articles. These requirements motivated an architecture with three primary layers: submission input, similarity processing, and administrative data management. Table 3 summarizes the key requirements and their implementation implications.

Table 3. System requirements and implementation implications

Aspect	Key finding	Implementation implication
Screening target	Titles and abstracts are prioritized for early screening	Similarity engine is optimized for concise but critical scientific text
Reference data	Journal articles must be available and easy to update	OJS synchronization module is included
System management	Managers require a practical administrative dashboard	Users, journals, submissions, and ignored-words modules are implemented
Result traceability	Screening results must be reviewable in detail	Token-level and aggregated results are stored per submission (final result)
Future research direction	A path for machine learning integration is required	Data structures and workflow are modular for extensibility

Prototype testing data and discussion

The prototype was evaluated using operational data generated from the application workflow. The reference corpus consisted of 162 article metadata records synchronized from OJS, while the testing records consisted of 30 submitted title-and-abstract documents. This dataset size is sufficient for prototype validation because the focus of the study is functional readiness, traceability, and output consistency rather than a large-scale benchmark comparison. The testing summary is presented in Table 4.

Table 4. Prototype testing dataset and black-box summary

Testing component	Data or scenario	Observed result
Reference corpus	162 OJS-synchronized article metadata records	Available for comparison in the journal table
Submission records	30 submitted title-and-abstract documents	27 records completed and 3 records pending
Black-box scenarios	12 functional scenarios covering input, preprocessing, similarity processing, OJS synchronization, reporting, and security baseline	12 scenarios produced outputs that matched the expected results

Security baseline	Authentication, math CAPTCHA, input sanitization, and URL parameter protection	Baseline controls worked and require further hardening
Reporting	History view and printable similarity report	Reports were generated and could be traced to submission records

The similarity-output analysis was conducted by examining representative records shown in the dashboard and printable report. The application uses a 0.0000–1.0000 score scale and displays the value as a percentage. As shown in Table 5, the documented records produced high lexical similarity scores, which confirms that the system can detect strong surface-level similarity among short scientific metadata. The result should be interpreted as an early-warning indicator, not as a final plagiarism decision, because Jaro-Winkler primarily measures lexical closeness.

Table 5. Representative similarity-output records from JARWIN testing

No.	Output record context	Jaro-Winkler score	Percentage	Interpretation
1	Printable final report sample	0.8097	80.97%	High lexical similarity
2	Recent completed check	0.8265	82.65%	High lexical similarity
3	Recent completed check	0.8213	82.13%	High lexical similarity
4	Recent completed check	0.8250	82.50%	High lexical similarity
5	Reference comparison sample	0.7106	71.06%	High lexical similarity
6	Reference comparison sample	0.7514	75.14%	High lexical similarity
7	Reference comparison sample	0.7285	72.85%	High lexical similarity
8	Reference comparison sample	0.7402	74.02%	High lexical similarity

These results strengthen the practical contribution of the prototype. The functional test shows that JARWIN can connect input, preprocessing, calculation, storage, and reporting into one auditable workflow. This addresses the operational gap identified in the introduction, where similarity studies often stop at algorithmic comparison and do not provide an institutional workflow for journal managers. The high similarity values in Table 5 are consistent with the strength of Jaro-Winkler in identifying character-level and prefix-based closeness in short texts (Jaro, 1989; Winkler, 1990; Santosa, 2022). However, the findings also confirm the limitation of lexical matching: paraphrased texts with different wording may not be fully captured. Therefore, the stored token-level and aggregated scores are important because they can become baseline features for hybrid models involving semantic representation and machine-learning classifiers, as suggested in prior studies on lexical-semantic similarity and sentence embeddings (Foltýnek et al., 2020; Reimers & Gurevych, 2019; Rudwan & Fonou-Dombeu, 2023).

System architecture and technology stack

JARWIN is implemented as a lightweight web application using CodeIgniter 3 with a MySQL/MariaDB backend. The architecture separates user interaction, processing modules, database storage, and external integration with OJS. This separation simplifies maintenance and supports gradual extension toward hybrid similarity models. Figure 2 depicts the overall architecture, while Table 6 details the technology components and selection rationale.



Figure 2. JARWIN system architecture

Table 6. Technology stack and selection rationale

Component	Implementation	Rationale
Framework	CodeIgniter 3.1.10	Lightweight, stable, and suitable for rapid prototyping
Programming language	PHP	Easy to deploy on campus servers and supports CI integration
Database	MySQL / MariaDB	Supports storage of journal data, submissions, and similarity outputs
UI stack	Bootstrap / Limitless Admin Theme, jQuery, DataTables	Accelerates dashboard and data-table implementation
External integration	Open Journal Systems (OJS) melalui REST API	Facilitates reference-article synchronization
Core libraries	Custom preprocessing, Jaro-Winkler, sanitasi input, auto numbering	Addresses similarity-checker specific requirements

Functional modules

The system provides both public-facing and administrative functions. Public modules support similarity checking, while administrative modules manage users, journals, submissions, and configuration (e.g., ignored words), and provide reporting. Table 7 lists the main modules and their outputs.

Table 7. Main system modules and outputs

Module	Primary function	Output
Landing page / checker	Accepts title/abstract input, shows statistics, and routes to similarity processing	New submission and initial screening output
Processing module	Performs preprocessing, Jaro-Winkler matching, and final score computation	Per-journal similarity scores and final result
Admin dashboard	Provides data overview and access to management modules	Operational control
Journal management	Stores, displays, deletes, and synchronizes reference journal articles	Up-to-date reference corpus
Submission management	Shows screening history and results	Traceable screening history

Reporting / printing

Generates overall or per-
submission reports

Similarity report documents

Database design and similarity computation workflow

To ensure traceability, the database schema stores submissions, journal references, aggregated results, and token-level similarity records. After preprocessing, each token in a submission is compared against tokens in reference articles using Jaro-Winkler; the system aggregates scores per reference journal and stores the highest aggregated score as the final similarity indicator. Figure 3 illustrates the workflow, and Table 8 summarizes the core database tables. Figure 4 shows the relationship among key tables.

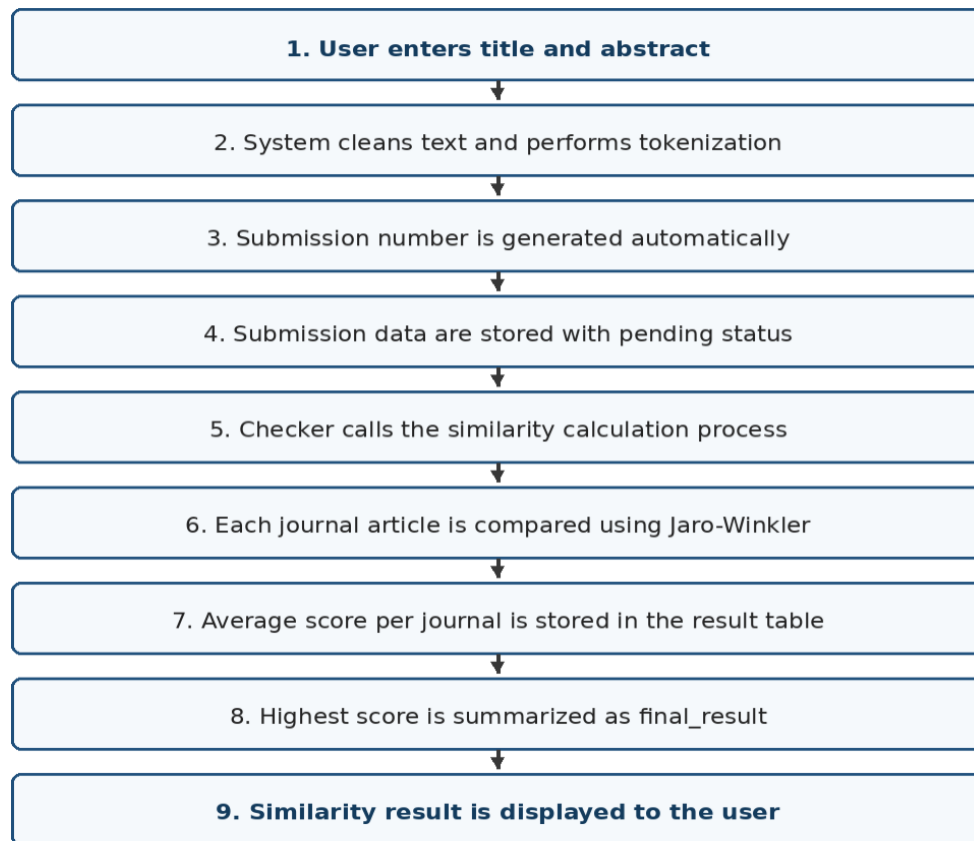


Figure 3. Similarity checking workflow in JARWIN

Table 8. Core database tables supporting similarity screening

Table	Function	Primary key	Notes
users	Stores system users	id_user	Used for authentication and admin identity
journal	Stores reference journal articles	id_journal	Includes title, abstract, author, and journal path relations
submissions	Stores screened submissions	no_submission	Central record for similarity screening
result	Stores aggregated average scores per journal	id_result	Links submissions to reference articles
jaro	Stores token-level similarity details	id_jaro	Basis for detailed similarity inspection
ojs / ojspath	Stores journal source configuration	id_ojs / id_path	Supports OJS synchronization

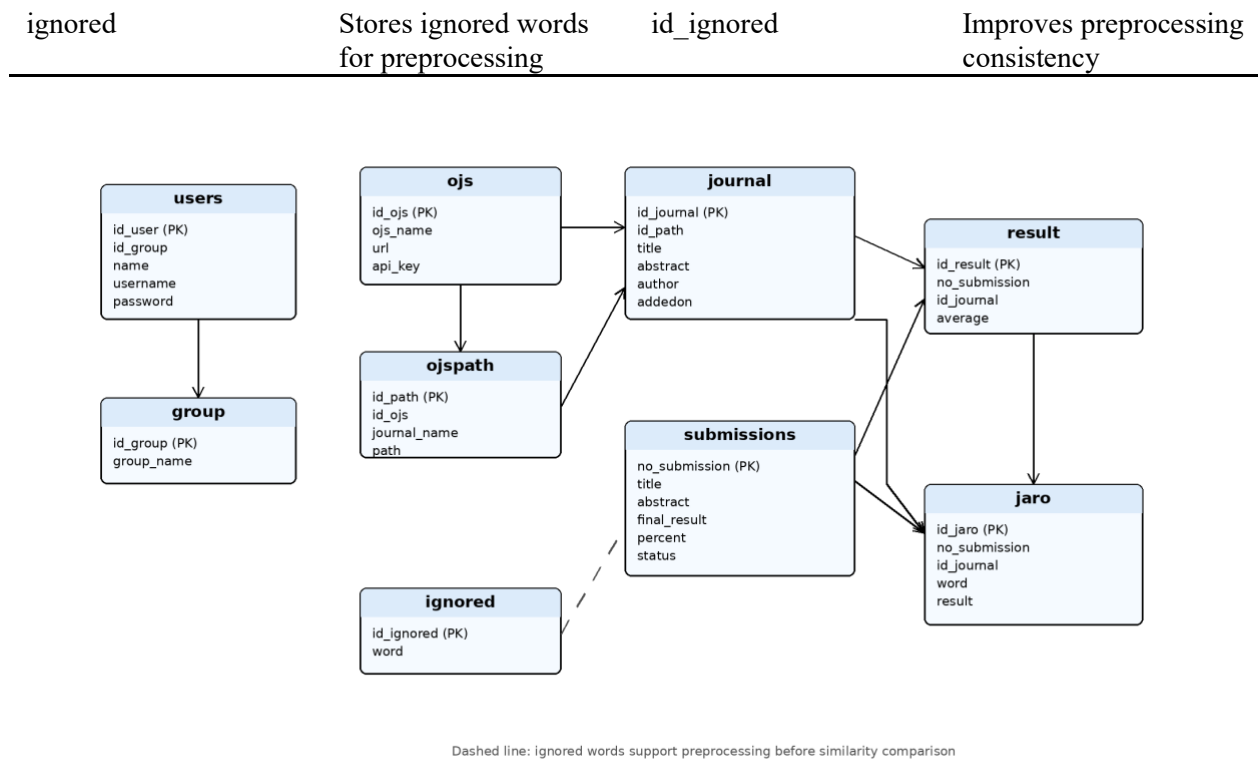


Figure 4. Relationships among key database tables in JARWIN

Security baseline and external integration

The prototype implements session-based authentication, a math CAPTCHA, input sanitization, and basic URL parameter protection. OJS integration via API enables automatic enrichment of reference articles, reducing manual data entry and improving operational feasibility. Table 9 summarizes the current security and integration implementation and highlights recommended hardening actions for the next development phase.

Table 9. Baseline security and external integration summary

Aspect	Current implementation	Development notes
Authentication	Session-based authentication and math CAPTCHA	Strengthen role-based access control
Input security	Input sanitization and XSS filtering are available	Standardize protections across all endpoints
Password management	Passwords stored using a legacy mechanism	Migrate to modern password hashing
Request protection	Basic protection exists for some processes	Enable CSRF protection and audit all forms
Journal data integration	Synchronize articles from OJS via API	Strength because reference corpus is easier to enrich
Reporting	Printable reports and submission traceability are available	Improve output format to be more informative

The evaluation demonstrates prototype feasibility, but it does not yet represent a full accuracy benchmark because the test submissions were not labeled by expert reviewers into duplicate, paraphrased, or original categories. A supervised accuracy evaluation should therefore be conducted in future work using an expert-labeled corpus and metrics such as precision, recall, F1-score, and receiver-operating characteristics. This limitation is important to avoid overclaiming the capability of a lexical similarity method while still recognizing its usefulness as a fast screening layer.

Application interface evidence

To complement the implementation findings, the final article provides prepared spaces for application screenshots. These screenshots are intended to document the operational evidence of JARWIN, including the public checking page, administrative dashboard, reference-data management, similarity results, detailed comparison outputs, and printable reports.

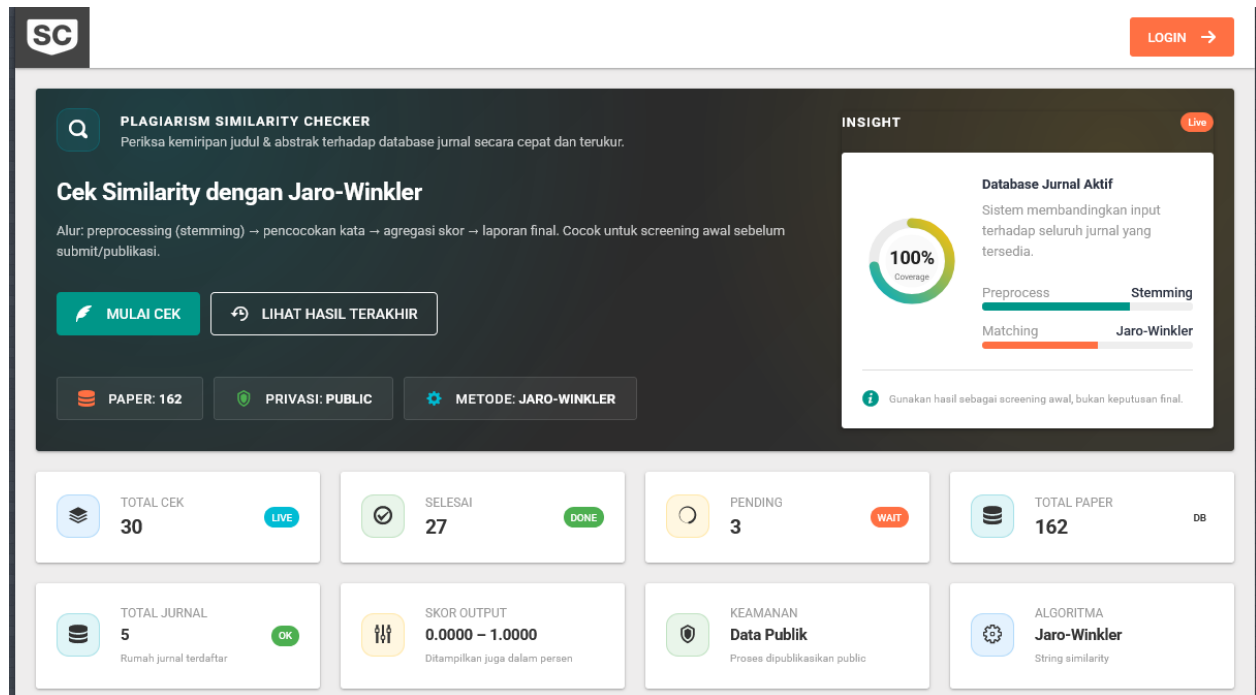


Figure 5. Public landing page and document similarity checking form

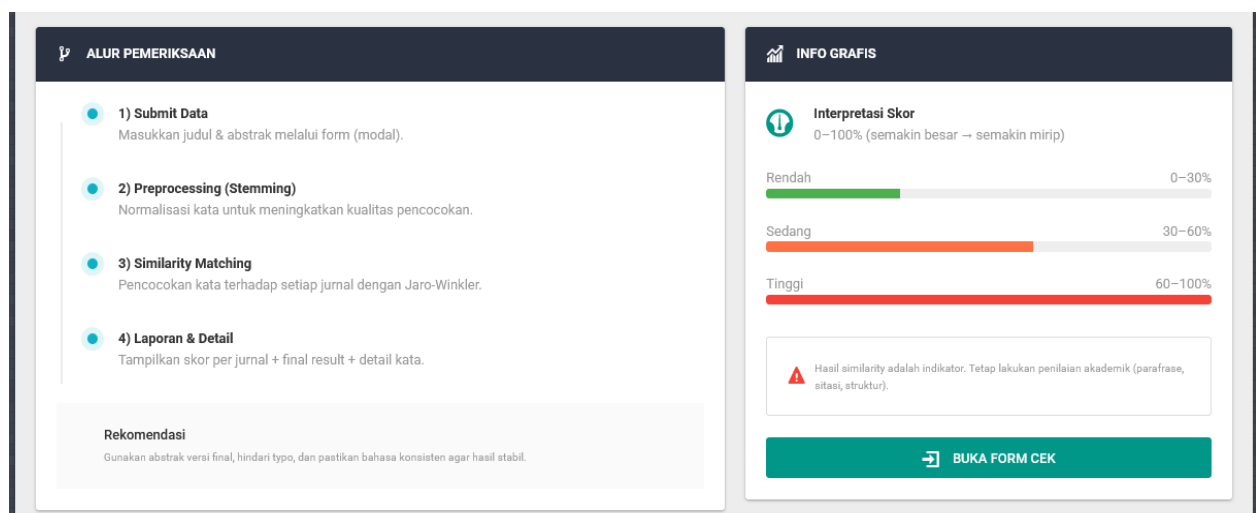


Figure 6. Administrative dashboard showing data summary and main navigation

159.	Web Based Leave Application for Teachers in Rural Multi-Island Regency	0.7106	71.06%	DETAIL →
160.	Implementasi Business Intelligence Untuk Analisa Dan Visualisasi Data Rating Amazon Sales Menggunakan Platform Data Studio	0.7514	75.14%	DETAIL →
161.	Implementation of a Web-Based Information Management System for Masjid Al-Mukhlisin	0.7285	72.85%	DETAIL →
162.	Influence of Teachers' Computing Skills on the Utilization of PMM at SDN 014 Ujung Tanjung, Benai District, Kuantan Singingi Regency	0.7402	74.02%	DETAIL →

FINAL RESULT

80.97%

Jaro Winkler: 0.8097

[PRINT RESULT](#)

Figure 7. Similarity checking result page showing aggregated Jaro-Winkler scores



SIMILARITY CHECKER
Similarity / Plagiarism Screening Report
Generated: 2026-04-29 07:24:37

SIMILARITY REPORT
#260413.SUB.00001
STATUS: DONE

Submission Metadata

Submitted at: 2026-04-13 15:14:20
Method: Jaro-Winkler Similarity
Scale: 0.0000 – 1.0000 (shown as %)

Final Similarity Score

80.97%
Jaro-Winkler: 0.8097
Highest similarity among compared journals.



Scan to verify

SUBMISSION DETAIL

TITLE

Figure 8. Printable similarity report and submission history interface

CONCLUSION

This study answers the first research objective by successfully developing JARWIN as a web-based scientific document similarity screening prototype. The system integrates text preprocessing, ignored-word removal, tokenization, Jaro-Winkler computation, OJS-based reference-data synchronization, token-level result storage, aggregated journal-level scoring, checking history, and printable reporting. These components show that the proposed prototype is not only an algorithmic implementation but also an operational workflow that can support journal managers in conducting early similarity screening.

This study answers the second research objective by evaluating the prototype through black-box functional testing and similarity-output analysis. The evaluation used 162 OJS-synchronized reference articles and 30 test submissions, consisting of 27 completed checks and 3 pending records. The black-box test showed that 12 functional scenarios produced outputs consistent with the expected results. The documented similarity outputs ranged from 0.7106 to 0.8297, including a final score of 0.8097 in the printable report sample. These findings indicate that JARWIN can generate measurable and traceable similarity indicators for scientific titles and abstracts.

The scientific contribution of this research lies in bridging lexical similarity computation with journal-data management and data readiness for future machine-learning optimization. Practically, higher-education institutions and journal managers can use JARWIN as a faster and more auditable early-screening tool before manuscripts proceed to deeper editorial or reviewer evaluation. Conceptually, the system confirms that lightweight lexical methods remain useful as a baseline layer, especially when the output is stored in a structured form that can later be combined with semantic features.

Future work should improve system security through stronger role-based access control, modern password hashing, complete CSRF protection, and endpoint-level security auditing. Further research should also evaluate the system using an expert-labeled corpus, compare Jaro-Winkler with other lexical and

semantic similarity metrics, and implement machine-learning models to detect paraphrased or semantically similar text that cannot be fully captured by lexical matching alone.

REFERENCES

- Ahmad, T., Ahamad, M., Ahmed, S. U., Ahmad, N., & Ahmad, N. (2022). Short question-answers assessment using lexical and semantic similarity-based features. *Journal of Discrete Mathematical Sciences and Cryptography*, 25(7), 2057–2067.
- Alian, M., & Awajan, A. (2021). Arabic sentence similarity based on similarity features and machine learning. *Soft Computing*, 25(15), 10089–10101.
- Baloi, A., Belean, B., Turcu, F., & Peptenatu, D. (2024). GPU-based similarity metrics computation and machine learning approaches for string similarity evaluation in large datasets. *Soft Computing*, 28(4), 3465–3477.
- Basile, A., Crupi, R., Grasso, M., Mercanti, A., Regoli, D., Scarsi, S., et al. (2024). Disambiguation of company names via deep recurrent networks. *Expert Systems with Applications*, 238, 122035.
- Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python*. O'Reilly Media.
- Christen, P. (2012). *Data matching: Concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer.
- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019* (pp. 4171–4186). Association for Computational Linguistics.
- Draschner, C. F., Jabeen, H., & Lehmann, J. (2022). SimE4KG: Distributed and explainable multi-modal semantic similarity estimation for knowledge graphs. In *2022 IEEE Fifth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)* (pp. 1–8). IEEE.
- Foltýnek, T., Meuschke, N., & Gipp, B. (2020). Academic plagiarism detection: A systematic literature review. *ACM Computing Surveys*, 52(6), 1–42.
- Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406), 414–420.
- Kazemian, H., & Shrestha, S. (2023). Comparisons of machine learning techniques for detecting fraudulent criminal identities. *Expert Systems with Applications*, 229, 120591.
- Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text classification algorithms: A survey. *Information*, 10(4), 150.
- Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.
- Molloy, C., Banks, J., Ding, S. H. H., Alaca, F., Charland, P., & Walenstein, A. (2025). Mecha: A neural-symbolic open-set homogeneous decision fusion approach for zero-day malware similarity detection. *IEEE Transactions on Software Engineering*, 51(2), 621–637.
- Novyantika, R. D., & Isa, S. M. (2023). Improve data text quality by applying text pre-processing method (case study). *International Journal of Engineering Trends and Technology*, 71(1), 94–108.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Pitchandi, P., & Balakrishnan, M. (2023). Document clustering analysis with aid of adaptive Jaro Winkler with Jellyfish search clustering algorithm. *Advances in Engineering Software*, 175, 103322.
- Potthast, M., Hagen, M., Göring, S., Rosso, P., & Stein, B. (2014). Overview of the 6th International Competition on Plagiarism Detection. In *CLEF 2014 Evaluation Labs and Workshop: Working Notes Papers*.
- Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of EMNLP-IJCNLP 2019* (pp. 3982–3992). Association for Computational Linguistics.
- Rudwan, M. S. M., & Fonou-Dombeu, J. V. (2023). Hybridizing fuzzy string matching and machine learning for improved ontology alignment. *Future Internet*, 15(7), 229.
- Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11), 613–620.
- Santosa, F. (2022). Accuracy in identifying similarity levels in scientific articles using the Jaro Winkler algorithm. *Jurnal Informasi dan Teknologi*, 4(3), 142–147. <https://doi.org/10.37034/jidt.v4i3.217>
- Singh, P. N., & Gowdar, T. P. (2021). Searching string in big-data: A better approach by applied machine learning. *SN Computer Science*, 2(3), 192.
- Sudarma, M. (2025). Integrating LSTM and hybrid methods for automatic Balinese script transcription. *Journal of Information Systems Engineering and Management*, 10(9s), 709–720.
- Winkler, W. E. (1990). String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. *Proceedings of the Section on Survey Research Methods, American Statistical Association*, 354–359.