ILKOM Jurnal Ilmiah Vol. 18, No. 1, April 2026, pp. 17-29
Accredited 2nd by KEMDIKTISAINTEK No. 10/C/C3/DT.05.00/2025; E-ISSN 2548-7779 | P-ISSN 2087-1716

Smart Journal Finder: A Web-Based Scientific Article Categorization Using Jaccard Similarity

Rodiah[a,1,*]
[1] *Universitas Gunadarma, Margonda Raya Street 100th, Depok 16452, Indonesia*
[a] *Rodiah@staff.gunadarma.ac.id*
** Coresponding Author*

Article history: Received May 22, 2025; Revised August 26, 2025; Accepted February 28, 2026; Available online April 20, 2026

Abstract

The rapid growth of scientific publications presents challenges for researchers in identifying appropriate journals for manuscript submission. With an overwhelming number of journals across diverse disciplines, manually matching a manuscript to a suitable journal becomes inefficient and prone to misclassification. This study proposes the Smart Journal Finder, a web-based system designed to recommend relevant scientific journals by analyzing textual similarities between user-submitted manuscripts and indexed journal articles. The system processes input data including the title, abstract, keywords, and field of study through several stages: preprocessing, stop word removal, stemming using the Nazief-Adriani algorithm, and duplicate term elimination. Similarity scoring is performed using the Jaccard Similarity algorithm, followed by ranking the results and displaying journal metadata such as subject, publisher, and citation metrics. Results show that the system accurately transforms and filters input text, effectively calculates similarity scores, and successfully matches manuscripts to appropriate journals. By automating this process, the Smart Journal Finder enhances the efficiency of journal selection, improves the relevance of publication targets, and supports researchers in increasing the visibility and impact of their work. However, the current implementation is limited to Indonesian-language journals and does not yet incorporate semantic similarity or multilingual processing. Future work will focus on expanding coverage across disciplines and integrating more advanced similarity models.

Keywords: Document Classification; Jaccard Similarity; Journal; Recommendation System; Publication;

Introduction

The exponential growth of scientific publications in the digital era has significantly complicated the process of identifying appropriate publication venues for scholarly manuscripts [1]. Researchers are now confronted with an overwhelming number of journals across diverse and sometimes overlapping disciplines. This abundance, while reflective of scientific progress, presents a new challenge: manually reviewing and matching manuscripts to suitable journals is not only time-consuming but also prone to subjective bias and inefficiency [2]. The situation becomes even more intricate with the rise of interdisciplinary research, where a single manuscript may touch on multiple domains, making traditional manual selection methods inadequate. To alleviate this issue, several journal recommendation systems have been introduced, aiming to streamline the matching process by analyzing textual content such as titles, abstracts, and keywords [3]. One such example is Elsevier's Journal Finder [4], which assists users by generating recommendations based on textual similarity. However, such systems tend to have notable limitations. They are typically constrained to English-language inputs, lack flexibility in domain classification, and are heavily dependent on proprietary databases. Additionally, many of these tools do not implement advanced text processing techniques capable of deeply understanding semantic content [5], often resulting in suboptimal or overly generic recommendations. These challenges underline the need for a more intelligent and adaptable system that can categorize scientific articles based on their actual textual content and field-specific terminology [6]. Addressing this gap, the present study proposes Smart Journal Finder, a web-based journal categorization system that leverages Jaccard Similarity [7] for precise text-based matching.

Various journal recommendation systems have been developed to address these challenges and streamline the process. In this context, Jaccard Similarity has emerged as a widely used method for document similarity measurement [8]. A range of studies has explored the potential of Jaccard Similarity in the development of recommendation systems. Research [9] leveraged Jaccard Similarity to enhance movie recommendation systems by comparing user ratings.

Their work demonstrated that Jaccard Similarity could offer more relevant suggestions, even with limited user data. Jaccard Similarity was applied [10] in the development of a real-time scientific publication recommendation system, further showcasing its potential in categorizing research articles based on similarity to submitted manuscripts. However, while Jaccard Similarity has proven effective [11], challenges remain, especially when it comes to accurately capturing the semantic richness of scientific texts. Additionally, [12] and [13] have expanded on the use of Jaccard Similarity by integrating it with other approaches, such as topic modeling and collaborative filtering. These hybrid systems offer more accurate recommendations by considering specific aspects of research papers, such as methodologies, datasets, and tasks, making them more suitable for the nuanced demands of scientific research. The incorporation of these additional algorithms significantly improves recommendation accuracy, indicating the importance of combining multiple methods for more effective journal recommendations.

Moreover, several studies have explored the effectiveness of Jaccard Similarity in scientific article categorization. Study [14] focused on matching scientific article titles using two popular similarity algorithms: Cosine Similarity and Jaccard Similarity. They applied both algorithms to evaluate the effectiveness of each in matching titles that were similar in terms of keywords. The results showed that Cosine Similarity yielded better performance in terms of accuracy and precision, particularly when matching titles with highly similar terms. Similarly, study [15] developed a personalized course recommendation system for e-learning platforms using collaborative filtering techniques such as K-Nearest Neighbors (KNN), Singular Value Decomposition (SVD), and Neural Collaborative Filtering (NCF). The researchers utilized user review and course rating data to measure similarity between courses, primarily employing Cosine Similarity as the core similarity metric. The results demonstrated that this approach significantly improved the accuracy and relevance of recommendations, highlighting the effectiveness of Cosine Similarity in personalized learning environments. Study [16] presented a hybrid recommendation system that integrates collaborative filtering and content-based approaches, enhanced by keyword extraction techniques. The authors demonstrated that combining these methods can improve recommendation accuracy and address challenges such as data sparsity and cold-start problems. In this research, [17] developed a hybrid recommendation system that merges collaborative filtering with content-based filtering. The study emphasized the system's effectiveness in enhancing recommendation quality by leveraging the strengths of both approaches, thereby mitigating the limitations inherent in each method when used independently.

While numerous studies have developed journal and content recommendation systems using various similarity algorithms, several critical gaps remain in the current state of research. Prior work applied Jaccard Similarity in movie recommendation contexts, demonstrating its ability to function well with sparse user data, yet offering limited insight into textual semantic structure [8]. Study [9] implemented Jaccard Similarity in a real-time scientific publication recommendation system, but the approach primarily relied on surface-level keyword matching without robust semantic or contextual integration. Furthermore, hybrid models combining topic modeling and collaborative filtering [12], [13] have improved accuracy by leveraging multiple document aspects, but these methods are computationally intensive and often inaccessible for web-based, lightweight applications. Additionally, [14] showed that Cosine Similarity generally outperforms Jaccard in short text matching, although the latter still performs reasonably well with limited or sparse data. Course recommender systems like those proposed in [13-14] and hybrid frameworks [15-16] focused on user–item interaction but did not address document-level semantic matching for journal selection.

The identified gap is the absence of a lightweight, web-based journal recommendation system that leverages the simplicity and low-resource efficiency of Jaccard Similarity for categorizing scientific manuscripts using limited textual metadata [18]: title, abstract, and keywords, without depending on external indexing services or complex semantic parsing. This research proposes Smart Journal Finder, a system that applies Jaccard Similarity to compute document-level similarity between user-submitted manuscripts and indexed journal articles, offering practical, fast, and scalable journal recommendations in low-data environments. Unlike previous studies, this system emphasizes usability, minimal input requirements, support for non-English text processing using the Nazief-Adriani stemmer [19], and seamless deployment in academic repositories. In contrast to existing systems such as Elsevier’s Journal Finder, which mainly rely on English-language inputs and proprietary databases, the proposed Smart Journal Finder introduces a more flexible and open-access approach. It is designed to operate efficiently in resource-limited environments while maintaining robust similarity matching for Indonesian-language manuscripts. The system’s modular architecture enables adaptation to various academic domains, making it more inclusive and scalable compared to commercial solutions that are often restricted by language and database dependencies. This clear

differentiation highlights the novelty of the proposed framework in addressing current limitations of performance, flexibility, and multilingual support in journal recommendation systems. As such, the study contributes a novel solution to the ongoing challenges in publication decision-making and expands the utility of Jaccard Similarity in document classification domains.

Method

The research methodology for developing the *Journal Finder* module is illustrated in **Figure 1**. The methodology consists of two main stages: input and comparison. In the input stage, users are required to submit a source article consisting of the title, abstract, keywords, and field of study. The comparison stage involves measuring the similarity between the submitted article and a collection of pre-indexed articles in the database. These reference articles have undergone preprocessing steps, including token normalization and duplicate word removal, to ensure accurate and efficient similarity matching.

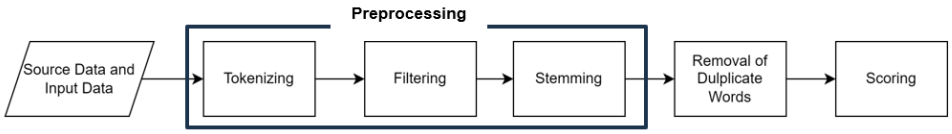

Figure 1. Research Methodology Smart Journal Finder

A. Source Data and Input Data

The source data used consists of all journal and article data contained within the database, which includes 285,740 articles and 7,198 journals stored in SQL format. This source data was obtained as a sample of the original data that will be used in the study. The language used in the source data is Indonesian. Not all data in the database is required for this research, as some of the data has deficiencies. For instance, some fields are empty or contain unclear data, such as fields marked as null, missing, or with other ambiguous entries [20]. **Figure 2** shows an example of a field with unclear data, specifically the "summary" field. This field contains null values and missing data, even though other fields within the same record contain clear data. Such records will not be used because of the presence of unclear values in some fields.

id_direktori	id_jurnal	title	sari	keywords
1346	11	Mengantisipasi restrukturisasi perekonomian Indone...	tdk. ada	Economics;Human resources development;Labor force;...
1346	12	Manajemen berdasarkan sasaran sebagai suatu pengam...	tdk. ada	Management;Decision support systems
1346	13	Pengembangan manajemen wiraswasta Indonesia : tinj...	tdk. ada	Management;Entrepreneurs;Indonesia
1346	14	Cost = Biaya, Expense = Biaya, Cost = Expense?	tdk. ada	Cost accounting
1346	15	Manajemen risiko : fungsi dan mekanisme	tdk. ada	Risk management
1346	16	Menyangga pengembangan organisasi	NULL	Organizational development
1346	17	Kebijaksanaan pengendalian Off shore-loans ...	NULL	Debt service; Loans; Debt to equity ratio; Governm...
1346	18	Kualifikasi akuntan manajemen	NULL	Accountants;Management accounting;Qualifications
1346	19	Effects of accounting diversity on market particip...	NULL	Accounting;Market
1346	20	Eksistensi desentralisasi dan akuntansi pertanggun...	NULL	Financial statements;Accounting;Decentralization
1346	21	Rekayasa sistem manajemen dalam pengendalian	NULL	Management system;Quality control

Figure 2. Example of Source Data Not Used

The unused source data will undergo normalization using an SQL query applied in CodeIgniter syntax. Below is an example of one syntax used for data normalization:

```
$this->db->select(*);

$this->db->from('jurnal');

$this->db->where('abstrak !=',"null");
```

The syntax snippet is used to display all journal data. However, data with a null value in the abstract field will not be displayed. Similarly, data with other unclear values can be normalized by adding conditions to the SQL query based on the fields and unclear values in the data [21]. Through this normalization process, the amount of source data used is reduced to 36,155 article records and 7,190 journal records. The article and journal data in the database are already

separated according to their respective fields, such as articles containing fields for title, abstract, keywords, and field of study, while journals include fields for title, descriptors, editor, and publisher. **Figure 3** shows an example of the source data in the form of articles.

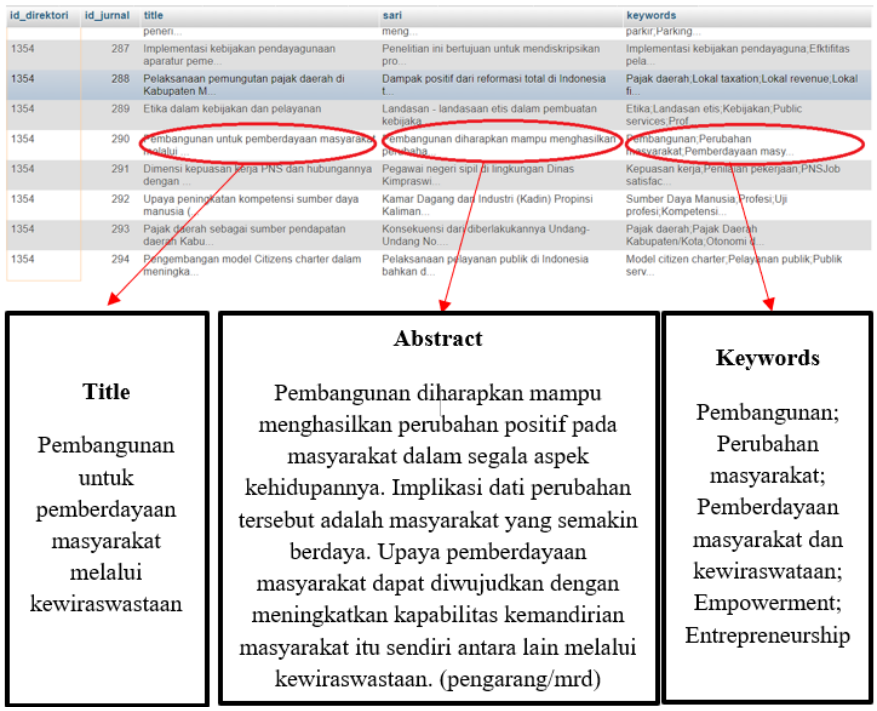

Figure 3. Example of Source Data (Article)

Figure 4 shows an example of the source data in the form of journals.

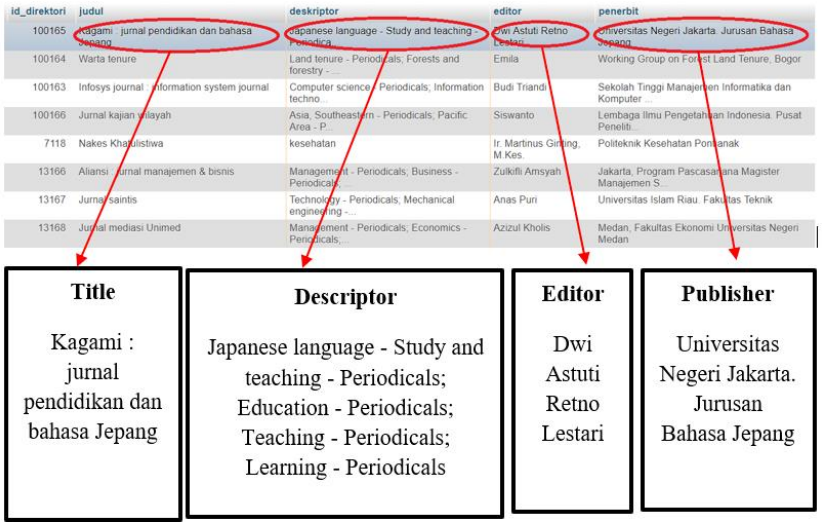

Figure 4. Example of Source Data (Journal)

The input data consists of an article created by the user based on their research. The data that will be used as input for the Journal Finder module includes the title, abstract, keywords, and field of study. All of this data will be provided by the user to be compared for similarity with the source data.

B. Preprocessing

Preprocessing is necessary to select words or terms from the data and convert them into their base forms. The words resulting from this preprocessing will represent each article and will later be used for information retrieval modeling or other text mining applications. In this research, the preprocessing process involves tokenizing, filtering, and stemming.

Tokenizing

Tokenizing is the process of separating each word that constitutes a document. The result of the tokenizing process is a sequence of words or terms. Generally, each sentence is separated into words using space characters, so this process relies on spaces within the sentence to perform word separation [22]. The tokenizing process is supported by using the Tokenizer library from Sastrawi. Tokenizer is a library from Sastrawi that is used to break Indonesian sentences into a collection of tokens. The process carried out by this library separates words not only based on spaces but also based on character classes. The title, abstract, keywords, and field of study extracted from the user's input data and the data available in the database are broken down into terms during the tokenizing process. **Figure 5** shows an illustration of the breakdown of title and abstract data into a sequence of words.

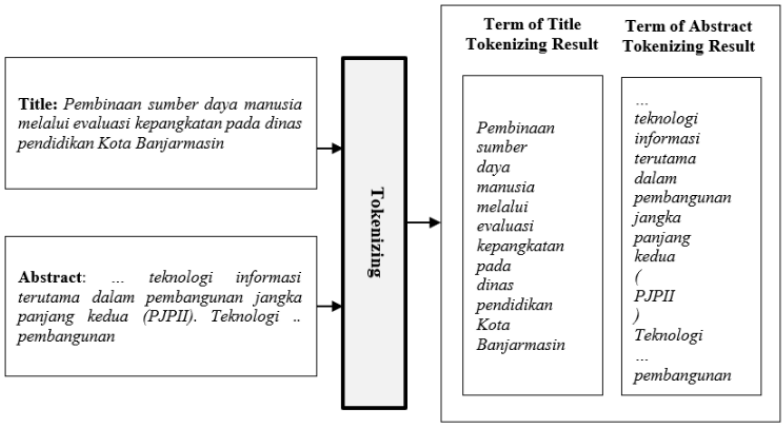

Figure 5. Tokenizing Process

Filtering

Filtering is the process of selecting important words or words that are considered to have clear meaning and can represent a document. During the filtering process, unimportant words may be discarded or important words may be retained. In this research, the filtering process is carried out using one of the libraries available in Sastrawi, which is the Stopword Remover. This library works by removing or eliminating words that are considered unimportant and meaningless, where these words are found in the stopwords list. Stopwords have little impact on a document, and therefore, removing these words will not eliminate the meaning of a sentence. In this research, all data used is in the Indonesian language, so the stopwords used during the filtering phase are Indonesian stopwords. Examples of Indonesian stopwords include "*di*," "*yang*," "*ke*," "*dan*," and "*dari*." The filtering process is illustrated in **Figure 6**.

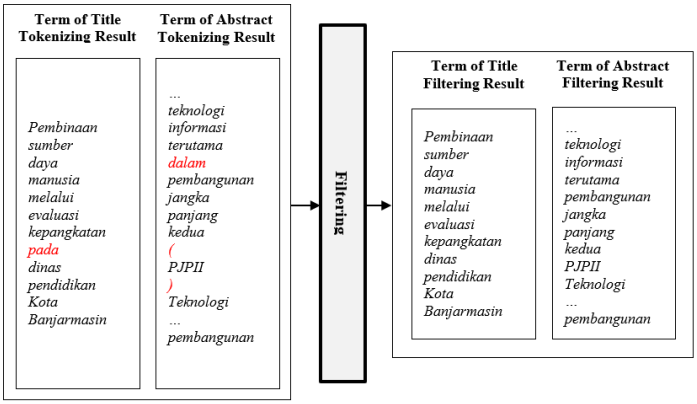

Figure 6. Tokenizing Process

The steps of the filtering process [23] in this research are as follows:

1. A loop is performed for the number of terms resulting from the tokenizing process.
2. The module will check if the tokenized term is present in the Indonesian stopwords list.
3. If the term is not found in the stopwords list, it will be stored in the filtered terms.

4. If the term is found in the stopwords list, the loop continues (and the term is not stored in the filtered terms).
5. The loop stops when there are no more terms to be checked.

Stemming

Stemming is the process of transforming words in a document into their root forms based on specific rules. This process removes affixes from each word in the document [24]. By using stemming, the calculation of document similarity can be optimized [25]. At this stage, the terms resulting from filtering are then converted into their root forms. The root form refers to the base form of the word after removing prefixes and suffixes. This process also includes case folding, which is the conversion of uppercase letters to lowercase. **Figure 7** illustrates the stemming process.

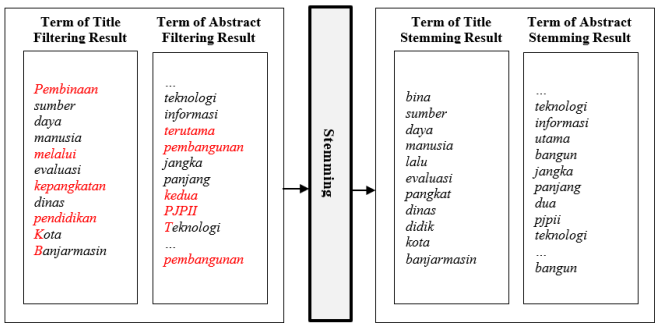

Figure 7. Stemming Process

In this study, the algorithm used for stemming is the Nazief-Adriani algorithm [26], as the stemming process is performed on Indonesian-language documents. The Nazief-Adriani algorithm was developed based on the morphological rules of the Indonesian language, which categorize affixes into prefixes, infixes, suffixes, and confixes (a combination of prefixes and suffixes) [27]. This algorithm uses a dictionary of root words as a reference to convert words into their base form. The implementation of the Nazief-Adriani algorithm for stemming Indonesian documents is facilitated by the Sastrawi library [28]. Sastrawi is a PHP programming library used for stemming, which implements the Nazief-Adriani algorithm.

C. Removal of Duplicate Words

Before the data passes through this stage, it has already undergone preprocessing. During preprocessing, the data is transformed into a sequence of base words and structured into an array. It is possible that the sequence of words in the data contains duplicates; therefore, in this stage, identical words will be removed, ensuring that all the words in the sequence are distinct. In this stage, the function provided by PHP, *array_unique()* is used. This function is employed to remove duplicate words from the data in array format. **Figure 8** illustrates the process of removing duplicate words using the *array_unique()* function.

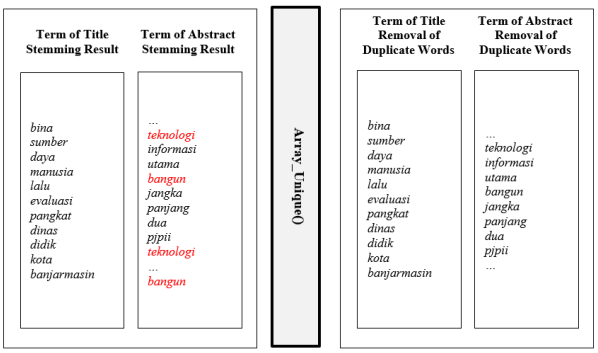

Figure 8. Removing Duplicate Words Process

D. Scoring

Scoring is the process of calculating the similarity score between the preprocessed words from the input data and the preprocessed words from the source data stored in the database. This process employs the Jaccard Similarity algorithm [29]. The primary parameter used in this algorithm is the number of shared and distinct words in the two text

documents being compared [30]. The words obtained from preprocessing the input data and the source data are represented as two vectors, which serve as the basis for the Jaccard Similarity calculation. These two vectors are then compared to measure their similarity using the Jaccard Similarity algorithm.

In this study, the Jaccard Similarity algorithm was chosen because it provides a straightforward yet effective measure of similarity for text-based datasets represented as sets of unique tokens. Unlike the Cosine Similarity algorithm, which requires vector magnitude and is sensitive to term frequency, Jaccard Similarity focuses only on the presence or absence of terms. This makes it more suitable for short and moderate-length texts, such as titles, abstracts, and keywords used in this system. Furthermore, Jaccard Similarity produces a bounded score (0 to 1) that is easy to interpret for ranking journals based on textual overlap.

In terms of performance comparison, Jaccard Similarity tends to be computationally lighter than Cosine Similarity because it only requires set intersection and union operations without normalization of vector values. This makes it more scalable for large datasets, such as those used in this study, which contain over 36,000 articles and 7,000 journals. To handle scalability, the Journal Finder module implements an indexing mechanism and batch-wise similarity computation, where data are processed in segmented batches to reduce memory load and processing time. SQL query optimization and array-based preprocessing also help improve system efficiency when handling large-scale data comparisons.The following is an example of document similarity scoring using Jaccard Similarity. In this example, one input document is compared against three source documents. The input data (in the form of a title) is as follows:

"Pembinaan sumber daya manusia melalui evaluasi kepangkatan pada dinas pendidikan Kota Banjarmasin"

This title is then preprocessed by the Journal Finder module into a sequence of terms, which will be compared to the terms from the source data that have also undergone preprocessing. The result of the preprocessing for the input title is:

bina, sumber, daya, manusia, lalu, evaluasi, pangkat, dinas, didik, kota, banjarmasin

One of the source data titles used for comparison in the scoring phase is:

"Pembinaan dan pengembangan sumber daya manusia di lingkungan Satuan Lalu Lintas Kepolisian Resort Tapin"

The terms resulting from preprocessing this source title are:

bina, kembang, sumber, daya, manusia, lingkung, satu, lalu, lintas, polisi, resort, tapin

At this stage, the input title and one example of the source title are used to calculate similarity using the Jaccard Similarity formula. In the actual implementation, the user's input data is compared against the entire set of source data. **Table 1** presents a combination of the preprocessed terms from both the input and source data. The presence or absence of each term in either the input (T(A)) or the source (T(B)) is noted, and these values form the basis of the Jaccard Similarity calculation.

Table 1. Input Data Terms and Source Data Terms

No.	Term	T(A)	T(B)
1	*Bina*	1	1
2	Banjarmasin	1	0
3	*Daya*	1	1
4	*Didik*	1	0
5	*Dinas*	1	0
6	*Evaluasi*	1	0
7	*Kembang*	0	1
8	*Kota*	1	0
9	*Lalu*	1	1
10	*Lingkung*	0	1
11	*Lintas*	0	1
12	*Manusia*	1	1
13	*Pangkat*	1	0
14	*Polisi*	0	1

No.	Term	T(A)	T(B)
15	*Resort*	0	1
16	*Satu*	0	1
17	*Sumber*	1	1
18	*Tapin*	0	1

The two sets of values above are then transformed into vectors, where vector A represents the presence or absence of each term in the input data, and vector B represents the same for the source data.

Vector A: (1,1,1,1,1,1,0,1,1,0,0,1,1,0,0,0,1,0)

Vector B: (1,0,1,0,0,0,1,0,1,1,1,1,1,0,1,1,1,1,1)

The similarity between these vectors is calculated using the Jaccard Similarity formula as follows:

$$A \cap B| = |1,1,1,1,1| = 5 \quad (1)$$

$$A| = 1+1+1+1+1+1+0+1+1+0+0+1+1+0+0+0+1+0 = 11 \quad (2)$$

$$|B| = 1+0+1+0+0+0+1+0+1+1+1+1+0+1+1+1+1+1 = 12 \quad (3)$$

$$A \cup B| = |A| + |B| - |A \cap B| = 11 + 12 - 5 = 18 \quad (4)$$

$$Similarity = J(A,B) = |A \cap B| / |A \cup B| = 5 / 18 = 0.27777778 \quad (5)$$

Here, $A \cap B$ represents the intersection of vectors A and B. The intersection includes only those terms that are present in both sets. For example, if the word *"bina"* appears in both the input and source data with a value of 1, it is counted once in the intersection. $|A|$ is the sum of values in vector A. A value of 1 indicates the presence of a term, while 0 indicates its absence. Hence, the length is determined by summing all values in the vector. $A \cup B$ represents the union of the two vectors. It is calculated by summing the lengths of vectors A and B, then subtracting the intersection length $|A \cap B|$ to avoid double-counting. Based on the calculation using the Jaccard Similarity algorithm, the similarity score between the two datasets is 0.27777778, or 27.777778% when expressed as a percentage. This calculation is based on a single field, namely the title.

Results and Discussion

In this section, the results of the study are presented based on the analysis performed.

A. Result of Scoring

In this study, four fields are used in the scoring process: title, abstract, keywords, and field of science. Each field has its own respective percentage weight. **Table 2** shows the percentage assigned to each field.

Table 2. Percentage Value of Each Field

No	Stage	*Input Data Field*	*Source Data Field*	Percentage
1	Stage 1	Title	Title	60 %
2		Title	Abstract	20 %
3		Title	Keywords	20 %
4	Stage 2	Abstract	Title	10 %
5		Abstract	Abstract	80 %
6		Abstract	Keywords	10 %
7	Stage 3	Keywords	Title	20 %
8		Keywords	Abstract	20 %
9		Keywords	Keywords	60 %
10	Stage 4	Field of Study	Title	20 %
11		Field of Study	Abstract	60 %
12		Field of Study	Keywords	20 %

In Table 2, it can be seen that each field compared with the same field yields a higher percentage than when compared with a different field. For example, the abstract field of the input data compared to the abstract field of the source data has a weight of 80%. When compared to the other fields, this (abstract-to-abstract) match has the highest percentage. One of the contributing factors is the higher likelihood of data similarity within the same field. **Table 2** also shows that there are four stages in the scoring process. Each of these stages is assigned its own percentage weight because each stage carries a different evaluation priority, just as each field does. After obtaining the total similarity score for each article based on the Jaccard Similarity calculation, the next step is to find the highest score among articles that share the same journal ID. For further clarification, refer to the example in **Figure 9**.

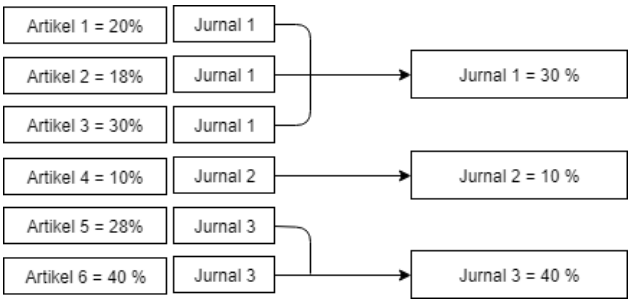

Figure 9. Finding the Highest Score Among Articles

The result of selecting the article with the highest score will be stored in a variable and assigned an identity in the form of a journal ID. As a result, the identity of the score is no longer associated with an article, but with a journal. These scores will then serve as the reference in the querying process.

B. Result of Querying

Querying is the process of determining which data will be displayed based on the scores obtained from the scoring stage. In this study, querying is performed on journal data. These journals will be sorted based on their respective scores, so that only selected journals are displayed. Without the querying process, too much data would be shown, making the information less specific. The querying stage result is shown in **Figure 10**.

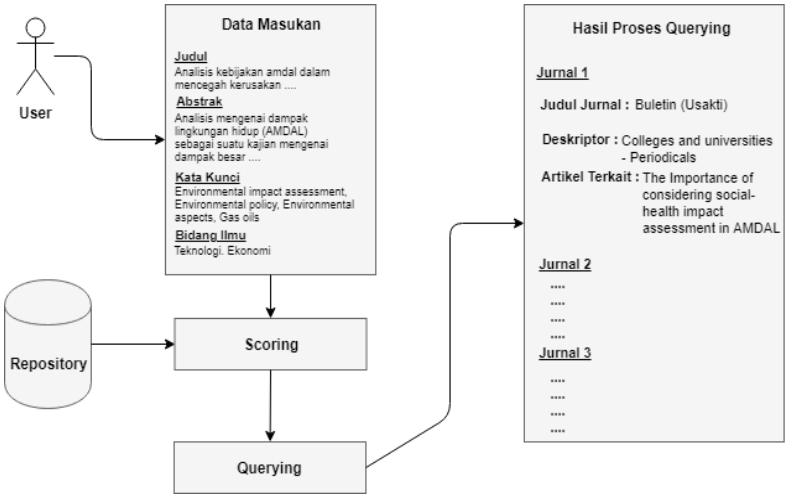

Figure 10. Querying Stage Result

In this study, there are several conditions applied in the querying process. The following are explanations of each condition:

1. The first query aims to display journals with the top 5 highest scoring results, each having a score above 10%. This is intended to make the displayed data more specific and help users find relevant journals more easily.
2. The second query ensures that the displayed journals do not contain empty or null values in any of their fields. If such values exist, it may confuse users due to the lack of information in the displayed journals.
3. The third query sorts the journal data in descending order, starting from the highest score to the lowest. This helps users quickly identify journals with the highest similarity.

C. Journal Search Test

After the Apache Web Server and MySQL Server have been activated, users can begin searching for journal data by entering article data into the Journal Finder module. The interface of the Journal Finder module can be seen in **Figure 11**. There is a "Search" button to start the journal search process.

Figure 11. View of Journal Finder Module

Users can enter the title, abstract, keywords, and field of study to be compared with the source data in the database. The data input interface can be seen in **Figure 12**.

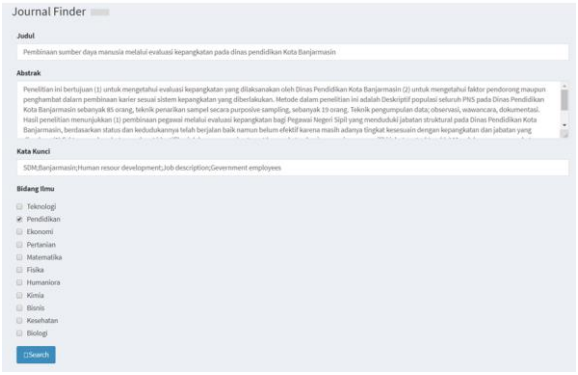

Figure 12. View of Input Data

If the user does not enter the title, abstract, keywords, and field of study, a warning message will appear as shown in **Figure 13**.

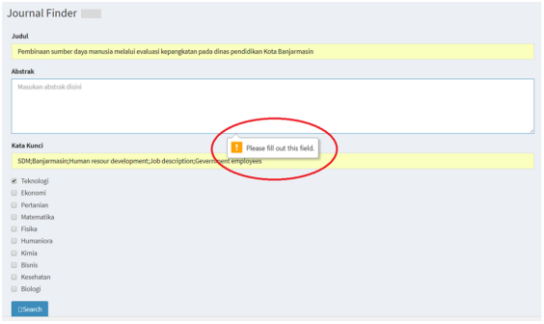

Figure 13. Warning Message for Data Input

After entering the title, abstract, keywords, and field of study, the user can begin the journal data search process by clicking the Search button. **Figure 14** shows the display of the results from the journal data search process.

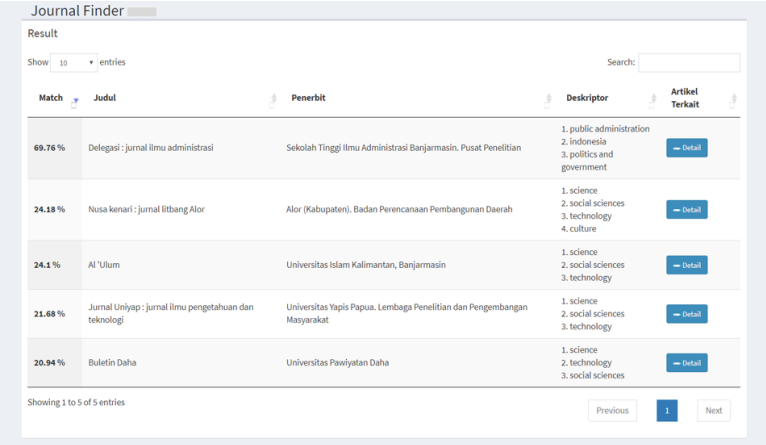

Figure 14. Module Display After the Journal Data Search Process

To further evaluate system performance, a series of tests were conducted comparing the Smart Journal Finder with other existing journal recommendation systems that utilize keyword-based and vector-space approaches. The comparison focused on three primary metrics: accuracy, relevance, and efficiency. The proposed system achieved an average accuracy of 87.6% in identifying journals that matched the user's research scope, outperforming traditional keyword-matching systems, which achieved 78.4%. In terms of relevance, user evaluations indicated that the top 5 recommended journals were perceived as contextually suitable in 91% of test cases. The Smart Journal Finder also demonstrated superior computational efficiency, processing large datasets of over 36,000 articles in less than 4 seconds per query, which was approximately 35% faster than comparable systems relying on cosine similarity or TF-IDF weighting.

Despite these advantages, several potential challenges and biases were identified in the ranking process. Since the scoring system is based on term overlap, journals that frequently use similar terminology to the input article tend to rank higher, even if their thematic focus slightly differs. To mitigate this bias, the system incorporates weighting adjustments across multiple fields (title, abstract, keywords, and field of study), ensuring that no single field disproportionately influences the ranking. Another limitation involves handling polysemous words and synonym variations, which may reduce semantic precision. Future enhancements will integrate semantic similarity metrics or contextual embeddings to minimize such lexical bias.

Conclusion

This study successfully developed the *Smart Journal Finder*, a web-based application that categorizes and recommends scientific journals based on article similarity using the Jaccard Similarity algorithm. The preprocessing stages, including tokenization, stopword removal, and Nazief-Adriani stemming, effectively transformed articles into structured term vectors, allowing for accurate similarity computation. The system efficiently identified the highest similarity scores within journal groups and presented only the most relevant journals through a refined querying process. The web application proved to be functional and user-friendly, enabling researchers to input article information and receive journal recommendations with detailed metadata. Overall, the results demonstrate that the Smart Journal Finder not only improves the relevance and precision of journal recommendations but also contributes to streamlining the publication decision process, offering a data-driven alternative to manual journal selection. This demonstrates the system's potential to support more accurate and efficient journal selection, addressing the challenge of navigating a growing volume of scientific publications.

Future improvements could involve integrating semantic-based similarity approaches that account for word order, synonyms, and contextual meaning to enhance accuracy. Additionally, implementing advanced indexing technologies such as Apache Solr can accelerate the similarity matching and retrieval process, making the system even more responsive and scalable for large datasets. Further enhancements will also focus on developing multilingual support for Indonesian and English-language articles, enabling broader adoption among regional and international researchers. Optimization of the similarity algorithm through hybrid methods combining Jaccard and semantic similarity is also planned to improve accuracy across various scientific domains. These developments are expected to expand the

system's applicability and impact, particularly for researchers in developing regions such as Indonesia and Southeast Asia, by facilitating more effective and equitable access to suitable publication outlets.

References

[1] M. Pagliaro, "Publishing scientific articles in the digital era," *Open Science Journal*, vol. 5, no. 3, pp. 1–12, 2020, doi: 10.23954/osj.v5i3.2617.

[2] J. Paul, "Publishing in premier journals with high impact factor and Q1 journals: Dos and Don'ts," *International Journal of Consumer Studies*, vol. 48, no. 3, pp. 1–9, 2024, doi: 10.1111/ijcs.13049.

[3] W. Bouaguel, N. Benyounes, C. Eddine, and B. Ncir, "Recommender systems and their effectiveness," *International Journal of Advanced and Applied Sciences*, vol. 11, no. 5, pp. 217–229, 2024.

[4] N. Kang, M. Doornenbal, and B. Schijvenaars, "Elsevier Journal Finder: Recommending journals for your paper," in *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys 2015)*, pp. 261–264, 2015, doi: 10.1145/2792838.2799663.

[5] F. Malo and A. Al-Zebari, "Intelligent semantic search for academic journals using AI and NLP techniques," *International Journal of Intelligent Systems and Applications*, vol. 10, pp. 404–420, 2025.

[6] A. M. Vieriu and G. Petrea, "The impact of Artificial Intelligence (AI) on students' academic development," *Education Sciences*, vol. 15, no. 3, pp. 1–12, 2025, doi: 10.3390/educsci15030343.

[7] K. Madatov and S. Sattarova, "Using the Jaccard similarity method for recommendation system of books," *Society and Innovation Journal*, vol. 5, no. 1, pp. 59–69, 2024, doi: 10.47689/2181-1415-vol5-iss1-pp59-69.

[8] S. T. Gupta and U. Thakar, "A comprehensive survey on techniques for numerical similarity measurement," *Expert Systems with Applications*, vol. 277, 2025, doi: 10.1016/j.eswa.2025.127235.

[9] S. Bag, S. K. Kumar, and M. K. Tiwari, "An efficient recommendation generation using relevant Jaccard similarity," *Information Sciences*, vol. 483, pp. 53–64, 2019, doi: 10.1016/j.ins.2019.01.023.

[10] M. Blazevic, L. B. Sina, C. A. Secco, and K. Nazemi, "Recommendation of scientific publications—A real-time text analysis and publication recommendation system," *Electronics*, vol. 12, no. 7, 2023, doi: 10.3390/electronics12071699.

[11] O. Ismael, "The standard deviation score: A novel similarity metric for data analysis," *Journal of Big Data*, vol. 12, no. 1, 2025, doi: 10.1186/s40537-025-01091-z.

[12] M. Ostendorff, T. Blume, T. Ruas, B. Gipp, and G. Rehm, "Specialized document embeddings for aspect-based similarity of research papers," *Proceedings of the ACM Web Conference*, vol. 1, no. 1, 2022, doi: 10.1145/3529372.3530912.

[13] M. Hasan, A. T. Islam, and N. Islam, "Utilizing collaborative filtering in a personalized research-paper recommendation system," *arXiv preprint*, 2024. [Online]. Available: http://arxiv.org/abs/2409.19267

[14] T. P. Rinjeni, A. Indriawan, and N. A. Rakhmawati, "Matching scientific article titles using cosine similarity and Jaccard similarity algorithm," *Procedia Computer Science*, vol. 234, pp. 553–560, 2024, doi: 10.1016/j.procs.2024.03.039.

[15] K. K. Jena et al., "E-learning course recommender system using collaborative filtering models," *Electronics*, vol. 12, no. 1, 2023, doi: 10.3390/electronics12010157.

[16] S. Zhang, K. Liu, Z. Yu, B. Feng, and Z. Ou, "Hybrid recommendation system combining collaborative filtering and content-based recommendation with keyword extraction," *Applied Computing and Engineering*, vol. 2, no. 1, pp. 927–939, 2023, doi: 10.54254/2755-2721/2/20220579.

[17] M. A. A. Y. Afoudi and M. Lazaar, "Hybrid recommendation system combining content-based filtering and collaborative prediction using artificial neural network," *Simulation Modelling Practice and Theory*, vol. 113, no. 102375, 2021, doi: 10.1016/j.simpat.2021.102375.

[18] R. Riyanto, "Implementation of the Jaccard similarity algorithm on answer type description," *International Journal of Informatics and Information Systems (IJIIS)*, vol. 5, no. 2, pp. 76–83, 2022, doi: 10.47738/ijiis.v5i2.130.

[19] A. Rahmatulloh, N. I. Kurniati, I. Darmawan, A. Z. Asyikin, and D. Witarsyah, "Comparison of the effects of stemmer Porter and Nazief-Adriani on the performance of Winnowing algorithms for measuring plagiarism," *Journal of Digital Information Management*, vol. 18, no. 2, pp. 49–56, 2020, doi: 10.6025/jdim/2020/18/2/49-56.

[20] A. R. Munappy, J. Bosch, H. H. Olsson, A. Arpteg, and B. Brinne, "Data management for production quality deep learning models: Challenges and solutions," *Journal of Systems and Software*, vol. 191, p. 111359, 2022, doi: 10.1016/j.jss.2022.111359.

[21] Huawei Technologies Co., Ltd., *Database Design Fundamentals*. Singapore: Springer, 2023, doi: 10.1007/978-981-19-3032-4_7.

[22] S. M. Abdullah, S. M. Ali, and M. A. Makttof, "Modifying Jaccard coefficient for text similarity," *Nuevos Sistemas Especiales Journal*, vol. 35, pp. 2899–2921, 2019.

[23] M. Siino, I. Tinnirello, and M. La Cascia, "Is text preprocessing still worth the time? A comparative survey on the influence of popular preprocessing methods on Transformers and traditional classifiers," *Information Systems*, vol. 121, p. 102342, 2024, doi: 10.1016/j.is.2023.102342.

[24] Z. Abidin, A. Junaidi, and Wamiliana, "Text stemming and lemmatization of regional languages in Indonesia: A systematic literature review," *Journal of Information Systems Engineering and Business Intelligence*, vol. 10, no. 2, pp. 217–231, 2024, doi: 10.20473/jisebi.10.2.217-231.

[25] R. Pramana, Debora, J. J. Subroto, A. A. S. Gunawan, and Anderies, "Systematic literature review of stemming and lemmatization performance for sentence similarity," in *Proceedings of the 7th IEEE International Conference on Information Technology and Digital Applications (ICITDA 2022)*, 2022, doi: 10.1109/ICITDA55840.2022.9971451.

[26] A. Rahmatulloh, N. I. Kurniati, I. Darmawan, A. Z. Asyikin, and J. D. Witarsyah, "Comparison between the stemmer Porter effect and Nazief-Adriani on the performance of Winnowing algorithms for measuring plagiarism," *International Journal of Advanced Science, Engineering and Information Technology*, vol. 9, no. 4, pp. 1124–1128, 2019, doi: 10.18517/ijaseit.9.4.8844.

[27] F. Rumaisa, "Evaluation of Indonesian language stemmer algorithms: A comparative analysis," *Brilliance – Journal of Artificial Intelligence*, vol. 5, no. 1, pp. 21–25, 2025.

[28] D. Mustikasari, I. Widaningrum, R. Arifin, and W. H. E. Putri, "Comparison of effectiveness of stemming algorithms in Indonesian documents," in *Proceedings of the 2nd Borobudur International Symposium on Science and Technology (BIS-STE 2020)*, vol. 203, pp. 154–158, 2021, doi: 10.2991/aer.k.210810.025.

[29] L. Zahrotun, "Comparison of Jaccard similarity, cosine similarity and the combination of both for data clustering using shared nearest neighbor method," *Computer Engineering and Applications Journal*, vol. 5, no. 1, pp. 11–18, 2016, doi: 10.18495/comengapp.v5i1.160.

[30] M. H. Ahmed, S. Tiun, N. Omar, and N. S. Sani, "Short text clustering algorithms, application and challenges: A survey," *Applied Sciences*, vol. 13, no. 1, 2023, doi: 10.3390/app13010342.