



Comparative Evaluation of String-Similarity Algorithms for Keyword Classification in Mobile Library OCR Systems

Damar Galih Aji Pradana ¹, Sritrusta Sukaridhoto ^{2*}, Nirwana Haidar Hari ¹

¹ Department of Informatics and Computer Engineering, Politeknik Elektronika Negeri Surabaya,
Surabaya, Indonesia

^{2,*} Department of Creative Multimedia Technology, Politeknik Elektronika Negeri Surabaya, Surabaya,
Indonesia

*Corresponding Author: dhoto@pens.ac.id

Article Info	Abstract
Article History: Submitted 27 Jul 2026 First Revised 27 Sep 2026 Accepted 28 Sep 2026 First Available Online 30 Sep 2026 Publication Date 30 Sep 2026 Keyword: Fuzzy String Matching; Jaro-Winkler; Levenshtein Distance; TF-IDF; Keyword Classification.	Automated keyword classification is a critical step in OCR-based library metadata systems, where OCR-extracted synopsis terms must ultimately be mapped to standardized subject keywords despite character-level recognition noise. While prior mobile OCR pipelines for library metadata extraction have adopted Levenshtein-distance-based fuzzy matching for this task, no systematic comparison exists to justify this choice against alternative string-similarity algorithms. This study isolates the keyword-classification sub-task and empirically compares three algorithms — Levenshtein-based similarity ratio, Jaro-Winkler similarity, and TF-IDF with cosine similarity — using noisy keyword-level queries as a controlled proxy for OCR-corrupted terms, under simulated OCR-noise conditions (clean, light, and heavy character-level corruption). Evaluation uses a curated 20-keyword library subject database and 900 test queries per algorithm across five random seeds. Jaro-Winkler achieves the highest overall classification accuracy (97.8%) and the fastest average matching time (22.68 microseconds per query), compared with Levenshtein-based matching (96.4% accuracy, 34.45 microseconds) and TF-IDF with cosine similarity (74.2% accuracy, 581.01 microseconds). A paired t-test across seeds confirmed that Jaro-Winkler's accuracy advantage over TF-IDF was statistically significant, while its advantage over Levenshtein was not and instead rests on its markedly lower and more stable matching latency. The gap against TF-IDF widened sharply under heavy noise (95.3% vs. 34.0% accuracy), showing that TF-IDF-based matching, despite its popularity for document-level text similarity, is

poorly suited to short-keyword matching under character-level noise. These findings, based on a 20-keyword benchmark, offer preliminary algorithm-selection guidance for developers of OCR-based metadata classification pipelines, pending validation on larger, real-world OCR-noise data.

Published by the Indonesian Society of Applied Science (ISAS).
This article is an open access article distributed under the terms of the Creative Commons Attribution (CC BY 4.0) license.



1. Introduction

Digital library modernization increasingly relies on Optical Character Recognition (OCR) to automate the extraction of bibliographic metadata, including titles, authors, publishers, and subject keywords, from scanned book covers and synopses [1]. A necessary step within this pipeline is keyword classification: mapping noisy, OCR-extracted synopsis text onto a standardized set of subject keywords so that books can be indexed, searched, and cataloged consistently. Because OCR output is imperfect, especially under adverse acquisition conditions such as glare, curvature, or low-end camera hardware, this classification step must tolerate character-level noise rather than relying on exact string matches [2], [3].

Existing mobile OCR-based library metadata systems have adopted Levenshtein-distance-based fuzzy string matching — typically via libraries such as FuzzyWuzzy — for this keyword classification task, owing to its simplicity, transparency, and freedom from training-data requirements [1]. However, this choice has generally been made on the basis of implementation convenience rather than systematic empirical comparison against alternative string-similarity algorithms with demonstrated strengths in related domains, such as record linkage and information retrieval [4], [5].

Two prominent alternatives are well established in the literature. Jaro-Winkler similarity, developed for record linkage and widely applied to name and short-string matching [13], [22], applies a prefix-weighting bonus that rewards strings sharing a common initial substring — a property plausibly well suited to keyword and name matching. TF-IDF (term frequency-inverse document frequency) weighting combined with cosine similarity is a foundational technique in information retrieval and document classification, widely used for matching text against a reference vocabulary [6], [7]. Despite the maturity of both techniques, no existing study has systematically compared them against Levenshtein-based matching specifically for the short-keyword, OCR-noise-tolerant classification task found in mobile library metadata systems [8], [9].

This gap matters because the three algorithms differ substantially in their sensitivity to character-level noise, their computational cost, and their suitability for short strings (typically one to three words) as opposed to full-length documents. A mobile OCR pipeline, constrained by on-device compute resources, would benefit from evidence-based algorithm selection rather than convention.

This study addresses the gap by empirically comparing Levenshtein-based similarity ratio, Jaro-Winkler similarity, and TF-IDF with cosine similarity for keyword classification, under a controlled, noise-stratified evaluation protocol that simulates three levels of OCR-style character corruption (clean, light, and heavy), using a curated library-subject keyword database as the classification target.

The contribution of this paper is threefold: (1) it presents, to the authors' knowledge, the first systematic comparison of these three string-similarity families specifically for OCR-derived keyword classification in the mobile library-metadata setting, where prior comparative studies have focused on plagiarism detection [12] or general record linkage rather than short-keyword classification under OCR noise; (2) it introduces a reproducible, noise-stratified evaluation protocol that can be reused to benchmark future string-similarity algorithms under simulated OCR degradation; and (3) it provides evidence-based algorithm-selection guidance, showing that Jaro-Winkler similarity outperforms the Levenshtein-based approach commonly deployed in existing mobile OCR library systems [1] in both classification accuracy and matching speed, while TF-IDF with cosine similarity is shown to be poorly suited to this short-keyword task despite its popularity in document-level text classification [10].

This comparison has direct practical value for vocational ICT education and industry practice. Polytechnics and vocational higher-education institutions frequently develop and maintain their own library digitization systems as both a teaching resource for software engineering and information systems curricula and a working production system for their own library operations, making algorithm-selection decisions that are typically made ad hoc, without access to comparative performance data. Industry developers of OCR-based cataloging tools for small and resource-constrained libraries — a common deployment context in vocational institutions — face similar decisions under tight computational and budget constraints, where the choice of matching algorithm directly affects both classification reliability and on-device processing cost. By quantifying the accuracy-latency trade-offs among three widely available, open-source string-similarity implementations, this study offers vocational ICT practitioners and system developers concrete, evidence-based guidance that can be applied directly during system design, without requiring them to conduct their own comparative benchmarking [11], [16].

2. Methods

Figure 1 illustrates the overall experimental pipeline: keyword database, noise injection, parallel evaluation of the three candidate algorithms, and the resulting evaluation metrics, detailed in the following subsections.

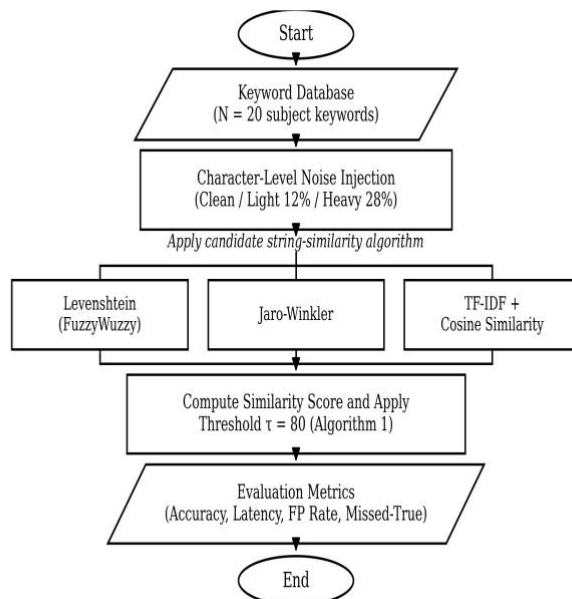


Fig. 1. Experimental Methodology Pipeline

2.1. System Context

The keyword classification task studied here is drawn from a Flutter-based mobile OCR pipeline for book metadata extraction, in which Google ML Kit performs on-device text recognition on a book's cover, verso page, and synopsis, and a string-similarity algorithm subsequently maps the extracted synopsis text to a curated set of subject keywords for cataloging [1]. That system currently uses a Levenshtein-based similarity ratio (via the FuzzyWuzzy library) for this step, accepting a candidate keyword as a match when its similarity score meets or exceeds a threshold $\tau = 80$ on a 0–100 scale. The present study isolates this keyword-classification sub-task and evaluates it independently of the full OCR pipeline, allowing controlled comparison of candidate algorithms under systematically varied noise conditions. To keep this comparison tractable and reproducible, each test query in the noise-injection protocol (Section 2.3) is a synthetically corrupted version of a single ground-truth keyword itself, used as a controlled proxy for the corrupted keyword-length terms an OCR pipeline would extract from a synopsis. This isolates the string-matching step from the upstream problems of keyword extraction and multi-keyword assignment from free-form synopsis text, and the results reported here should accordingly be read as a comparison of matching algorithms under keyword-length character noise, not as an end-to-end evaluation of synopsis-to-keyword classification; validating the algorithm ranking on real OCR-extracted synopsis fragments is identified as future work in Section 4.

Figure 2 situates the keyword-classification module studied in this paper within the broader four-layer system architecture, highlighting its position in the Processing & Validation Layer.

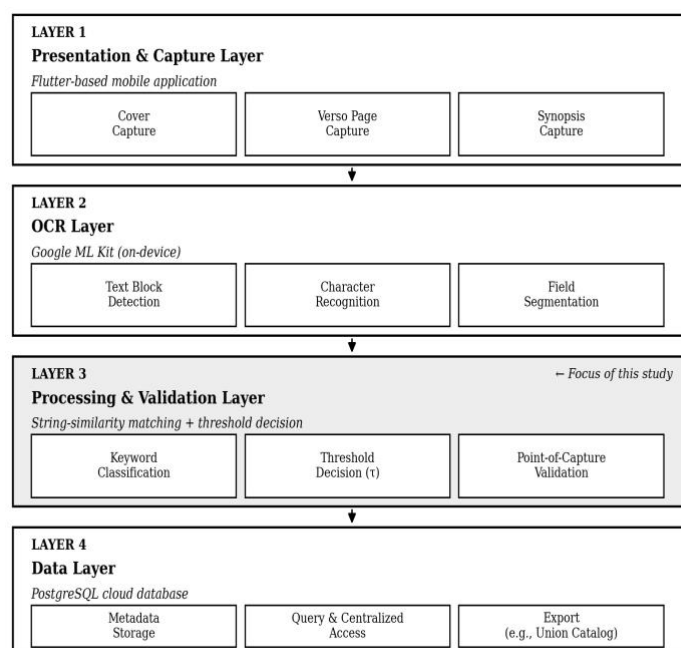


Fig. 2. Four-Layer Architecture of the Mobile OCR Library Metadata System

2.2. Candidate String-Similarity Algorithms

Three algorithms, representative of distinct families of string-similarity methods, were evaluated.

Levenshtein-based similarity ratio converts the Levenshtein edit distance - the minimum number of single-character insertions, deletions, and substitutions required to transform one string into another [21] - into a normalized similarity score:

$$S_{lev}(a, b) = \left(1 - \frac{lev(a, b)}{\max(m, n)}\right) \times 100 \quad (1)$$

Where $lev(a, b)$ is the Levenshtein edit distance and m, n are the lengths of strings a and b , respectively.

Jaro similarity measures agreement based on matching characters within a bounded window and the number of transpositions among them [22]:

$$S_{jaro} = \frac{1}{3} \left(\frac{m}{|a|} + \frac{m}{|b|} + \frac{m-t}{m} \right) \quad (2)$$

Where m is the number of matching characters between a and b (characters agreeing within a window of $\left\lfloor \frac{\max(|a|, |b|)}{2} \right\rfloor - 1$ positions, t is the number of transpositions (matching characters in different order, divided by two), and $|a|, |b|$ are the string lengths.

Jaro-Winkler similarity extends the Jaro score with a prefix-weighting bonus that increases similarity for strings sharing a common initial substring [13], [15], [22]:

$$S_{jw} = S_{jaro} + l \cdot p \cdot (1 - S_{jaro}) \quad (3)$$

where ℓ is the length of the common prefix (up to a maximum of four characters) and p is a scaling constant (standard value $p = 0.1$), giving strings with matching prefixes a similarity boost - a property plausibly advantageous for short subject-keyword matching.

TF-IDF weighting represents each string as a vector of character n -gram weights, where each weight reflects the term's frequency within the string relative to its rarity across the keyword database:

$$w_{i,d} = tf_{i,d} \times \log \left(\frac{N}{df_i} \right) \quad (4)$$

where $tf(i, d)$ is the frequency of n -gram i in string d , N is the number of keywords in the database, and $df(i)$ is the number of keywords containing n -gram i . Character n -grams (bigrams and trigrams) rather than word-level tokens were used, since subject keywords are short (typically one to three words) and word-level TF-IDF would be uninformative at this length.

Cosine similarity then measures the angular similarity between the query vector and each candidate keyword vector:

$$S_{cos}(\vec{q}, \vec{k}) = \frac{\vec{q} \cdot \vec{k}}{\|\vec{q}\| \|\vec{k}\|} \quad (5)$$

where q and k are the TF-IDF vectors of the query and candidate keyword, respectively. All three algorithms output a similarity score on a comparable 0–100 scale and use the same decision threshold $\tau = 80$ for match acceptance, consistent with the threshold used in the source system [1].

Algorithm 1. Threshold-Based Keyword Classification

Require: query string t , keyword database $K = \{k_1, \dots, k_N\}$, similarity function $S(\cdot, \cdot)$, threshold τ

Ensure: matched keyword $\hat{k}$ or reject

```

....1: best_score  $\leftarrow -\infty$ ; best_keyword  $\leftarrow$  null
....2: for each  $k_i \in K$  do
....3:   score  $\leftarrow S(t, k_i)$ 
....4:   if score > best_score then
....5:     best_score  $\leftarrow$  score; best_keyword  $\leftarrow k_i$ 
....6:   end if
....7: end for
```

```

....8: if best_score  $\geq \tau$  then
....9:   return best_keyword
....10: else
....11:   return reject
....12: end if

```

2.3 Keyword Database and Noise-Injection Protocol

A curated database of 20 library subject keywords, representative of common non-fiction subject classifications, was used as the fixed classification target for all three algorithms (Table 1).

Tab. 1. Library Subject Keyword Database (N = 20)

No.	Subject Keyword	No.	Subject Keyword
1	Islamic Studies	11	Education
2	Computer Science	12	Philosophy
3	Mathematics	13	Sociology
4	History	14	Medicine
5	Economics	15	Engineering
6	Psychology	16	Fine Arts
7	Literature	17	Political Science
8	Physics	18	Religion
9	Biology	19	Chemistry
10	Law	20	Geography

To simulate OCR-style character corruption, each test query was generated by selecting a ground-truth keyword and applying random character-level edit operations (substitution, deletion, or insertion) at a controlled rate: 0% (clean), 12% of characters (light noise), and 28% of characters (heavy noise). These three levels mirror the graded degradation observed in real OCR pipelines, from well-lit, high-quality captures through to glare-affected or low-end-camera captures [1].

Clean (0%)	Light Noise (12%)	Heavy Noise (28%)
"Philosophy"	"Philosaphy"	"Philsphqy"

Fig. 3. Example of Character-Level Noise Injection at Three Corruption Levels

2.4 Evaluation Metrics

Four metrics were recorded for each algorithm at each noise level: (1) top-1 classification accuracy — the proportion of queries for which the highest-scoring keyword above threshold τ matched the ground-truth keyword; (2) average matching time per query, in microseconds; (3) false-positive rate — the proportion of queries accepted (score $\geq \tau$) but matched to an incorrect keyword; and (4) missed-true count — the number of queries where the correct keyword was the top-ranked candidate but fell below the acceptance threshold τ .

2.5 Experimental Setup

For each of the three noise conditions, 60 queries were generated per random seed, across five independent seeds, yielding 900 test queries per algorithm (300 per noise condition). All three algorithms were implemented in Python. Levenshtein-based matching used the FuzzyWuzzy library and Jaro-Winkler similarity used the jellyfish library; both libraries have been independently benchmarked for text-matching accuracy and processing efficiency in prior comparative work [20]. TF-IDF with cosine similarity was implemented using scikit-learn's TfidfVectorizer and cosine_similarity functions with a character n-gram range of (2,3), following standard TF-IDF vectorization practice [6], [7]. All experiments were executed under identical conditions on the same machine to ensure timing comparability. The acceptance threshold $\tau = 80$ was adopted directly from the source mobile OCR system [1] to keep the comparison consistent with deployed practice. Because the three algorithms produce similarity scores with different distributions, a single shared threshold could in principle favor one algorithm over another rather than reflect a genuine difference in match quality; this risk is addressed directly through a dedicated sensitivity analysis across $\tau = 60\text{--}90$ in Section 3.4, which shows that the algorithm ranking is stable across the tested threshold range rather than an artifact of the $\tau = 80$ choice.

2.6 Statistical Analysis

To assess whether the observed accuracy differences were statistically reliable rather than artifacts of a particular random seed, overall accuracy was recorded separately for each of the five seeds, and paired two-tailed t-tests were computed between algorithm pairs across the five paired seed measurements, with a Bonferroni-corrected significance threshold of $\alpha = 0.025$ applied to account for the two pairwise comparisons reported (Jaro-Winkler vs. Levenshtein; Jaro-Winkler vs. TF-IDF). Per-condition and overall accuracies are reported as mean $\pm$ standard deviation across the five seeds, and paired-sample effect sizes (Cohen's $d_r = t/\sqrt{n}$) are reported alongside each t-test to distinguish non-significance from a genuinely negligible effect. Because only five seeds were used, the design has limited statistical power to detect moderate effect sizes, which is noted as a limitation in Section 4.

3. Results and Discussion

Table 2 summarizes classification accuracy, average matching time, and false-positive rate for all three algorithms across the three noise conditions and overall.

Tab. 2. Classification Accuracy, Matching Time, and False-Positive Rate by Algorithm and Noise Condition (900 queries per algorithm)

Algorithm	Condition	Accuracy (%)	Avg. Time (μ s)	FP Rate (%)
Levenshtein (FuzzyWuzzy)	Clean	100.0	33.90	0.0
	Light Noise	98.3	34.53	0.0
	Heavy Noise	91.0	34.92	0.0
	Overall	96.4	34.45	0.0
Jaro-Winkler	Clean	100.0	22.21	0.0
	Light Noise	98.0	23.04	0.0
	Heavy Noise	95.3	22.79	0.0
	Overall	97.8	22.68	0.0

TF-IDF + Cosine	Clean	100.0	588.05	0.0
	Light Noise	88.7	574.74	0.0
	Heavy Noise	34.0	580.25	0.0
	Overall	74.2	581.01	0.0

Jaro-Winkler similarity achieved the highest overall classification accuracy ($97.8 \pm 0.9\%$, mean $\pm$ SD across the five seeds) and the lowest average matching time ($22.68 \mu\text{s}$ per query, Table 2), compared with the Levenshtein-based approach ($96.4 \pm 1.2\%$ accuracy, $34.45 \mu\text{s}$) currently deployed in the source system [1] and with TF-IDF with cosine similarity ($74.2 \pm 4.0\%$ accuracy, $581.01 \mu\text{s}$). A paired two-tailed t-test across the five seeds confirmed that the accuracy advantage of Jaro-Winkler over TF-IDF was statistically significant ($t = 12.01$, $p < 0.001$, $d_r = 5.37$, a very large effect) and remained significant under the Bonferroni-corrected threshold of $\alpha = 0.025$, whereas the accuracy difference between Jaro-Winkler and Levenshtein did not reach significance at either threshold ($t = 1.86$, $p = 0.136$, $d_r = 0.83$, a large effect size despite non-significance); with only five seeds, the design has limited power to confirm an effect of this size, so this result is better read as inconclusive than as evidence of equivalence. The practical advantage of Jaro-Winkler over Levenshtein therefore rests primarily on its lower and more stable per-query latency rather than on a confirmed accuracy improvement. The accuracy gap against TF-IDF widened sharply under heavy noise, where TF-IDF fell to 34.0% — far below Levenshtein (91.0%) and Jaro-Winkler (95.3%) at the same noise level. All three algorithms exhibited a false-positive rate of 0.0% at the threshold $\tau = 80$ used throughout, indicating that the primary failure mode under noise was rejecting a correct match (missed-true) rather than accepting an incorrect one.

This near-zero false-positive rate is a property of the keyword database used here rather than a general guarantee of the algorithms themselves. Pairwise similarity among the 20 subject keywords showed that even the two most lexically similar entries (“Psychology” and “Philosophy”) scored below the acceptance threshold under Jaro-Winkler (77.7 , clean input), and character-level noise, being random and undirected, was far more likely to lower a query's similarity to its own true keyword than to raise its similarity to a different one. Keyword databases containing more lexically confusable entries (e.g., “Biology” and “Biotechnology”) would likely yield a non-zero false-positive rate, and this should be examined in future work using taxonomies with finer-grained, more closely related subject categories.

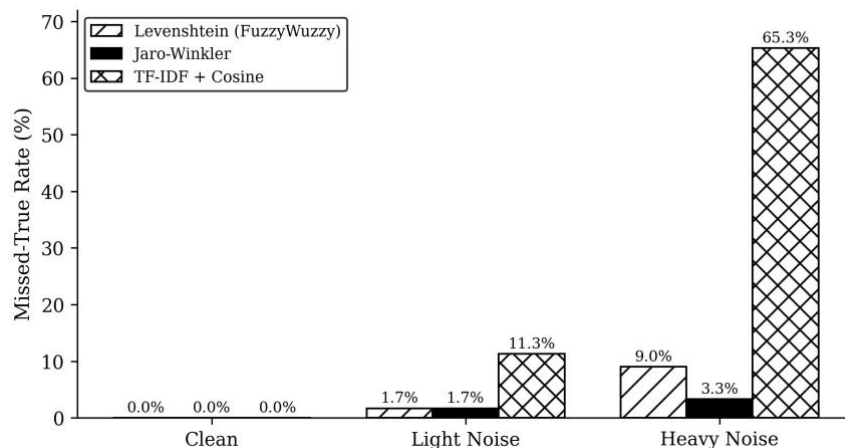
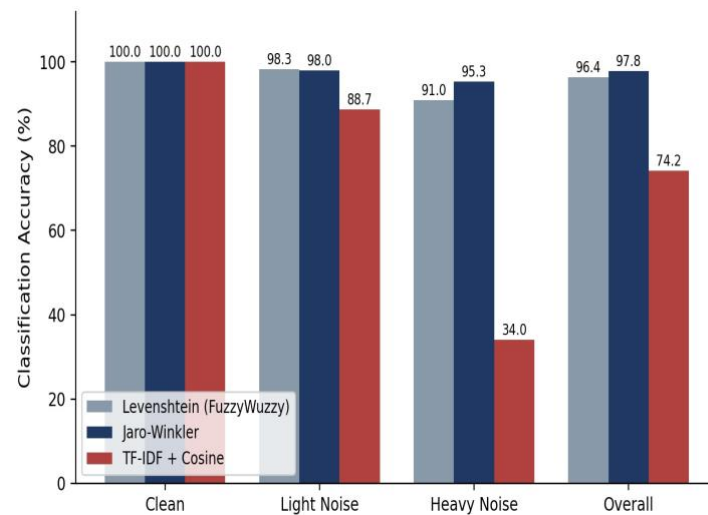
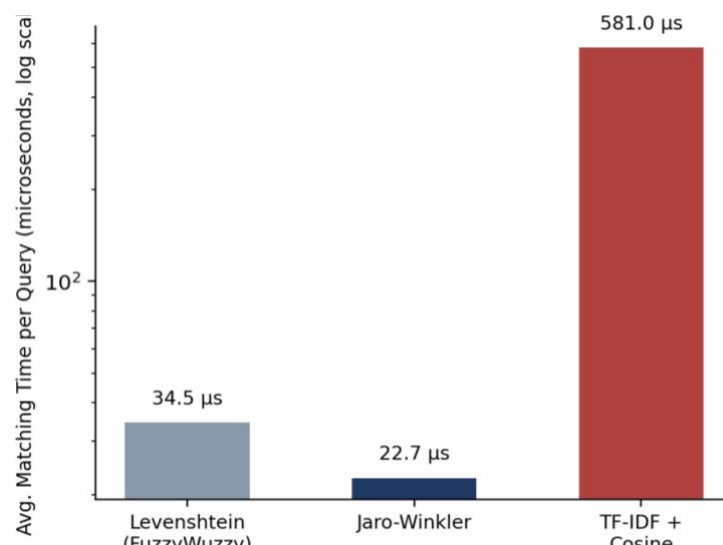


Fig. 4. Missed-True Rate by Algorithm and Noise Condition



(a)



(b)

Fig. 5. (a) Classification accuracy by algorithm and noise condition; **(b)** average matching time per query (log scale, μ s).

3.1 Why Jaro-Winkler Outperforms Levenshtein for Short Keywords

Jaro-Winkler's prefix-weighting bonus rewards strings that share a common initial substring. Because the noise-injection protocol (Section 2.3) distributes character-level edits uniformly at random across each keyword rather than targeting a specific position, the probability that a query's first few characters remain unedited is higher than the probability that the string as a whole remains unedited, and this probability falls only gradually as the corruption rate increases. Jaro-Winkler's prefix bonus can therefore preserve some similarity signal from an intact prefix even when the remainder of the string has been substantially corrupted, whereas Levenshtein-based similarity weights all character positions uniformly and gains no comparable benefit from an intact prefix alone. This may partly explain why Jaro-Winkler's accuracy declines more gradually than Levenshtein's as noise increases (Table 2); however, as reported above, the accuracy difference between the two algorithms was not statistically significant across seeds ($t = 1.86$, $p = 0.136$). This account should therefore be read as a plausible partial explanation

for a latency-favorable rather than an accuracy-favorable result, not as a confirmed causal mechanism — a position-resolved analysis of where noise-induced edits fall relative to each algorithm's errors would be needed to test it directly, and is left to future work.

3.2 Why TF-IDF with Cosine Similarity Is Poorly Suited to This Task

TF-IDF with character n-gram weighting is designed for longer documents, where sufficient n-gram overlap exists between related texts even after moderate corruption. For short keywords of only one to three words, however, a single character edit can eliminate two or three overlapping bigrams or trigrams entirely, causing the query vector to diverge sharply from its true-match vector in cosine space. This explains the steep accuracy decline observed under heavy noise (34.0%), where the majority of queries had their correct keyword's true-match score fall below the acceptance threshold (missed-true, Table 2). TF-IDF with cosine similarity also incurred substantially higher computational cost — approximately 17 times slower than Jaro-Winkler on average — due to the overhead of vectorization and vector-space cosine computation for each query, compared to the direct character-comparison operations used by Levenshtein and Jaro-Winkler.

3.3 Implications for Mobile OCR Deployment

For mobile OCR-based library metadata systems operating under on-device compute constraints, these results indicate that Jaro-Winkler similarity is a reasonable drop-in alternative to the Levenshtein-based approach currently used in the source system [1]: it matches Levenshtein's accuracy (the difference was not statistically significant) while running with lower and more stable per-query latency. All timings reported here were measured in a Python implementation on desktop/server hardware (Section 2.5), not on an Android device; while the relative ordering of the three algorithms — Jaro-Winkler fastest, Levenshtein close behind, TF-IDF roughly 17× slower — is unlikely to reverse on mobile hardware given the algorithms' comparable asymptotic complexity, the absolute microsecond figures should not be read as predictions of on-device latency. Confirming the practical benefit for real-time, point-of-capture keyword classification will require benchmarking on representative Android hardware using the actual FuzzyWuzzy/jellyfish/scikit-learn (or equivalent native) implementations deployed in the pipeline; we identify this as necessary follow-up work rather than a claim made by the present study. TF-IDF with cosine similarity, despite its widespread use in document-level text classification, is not recommended for this short-keyword classification task given its substantially lower accuracy under noise and its higher computational cost at every noise level tested.

3.4 Sensitivity to the Acceptance Threshold

Because the acceptance threshold τ governs the trade-off between accepting correct matches and rejecting spurious ones, its influence on overall accuracy was examined across the range $\tau = 60$ to 90 (Figure 6). Jaro-Winkler retained the highest accuracy across the entire range and degraded most gracefully as τ increased, remaining above 89% even at the strict setting $\tau = 90$, where Levenshtein dropped to 69.0% and TF-IDF to 48.1%. At the deployed setting $\tau = 80$, Jaro-Winkler (97.8%) already led Levenshtein (96.4%) and TF-IDF (74.2%). This monotonic ordering — Jaro-Winkler $\geq$

Levenshtein $\gg$ TF-IDF at every threshold tested — indicates that the observed ranking is not an artifact of the specific threshold choice, strengthening the algorithm-selection guidance offered here.

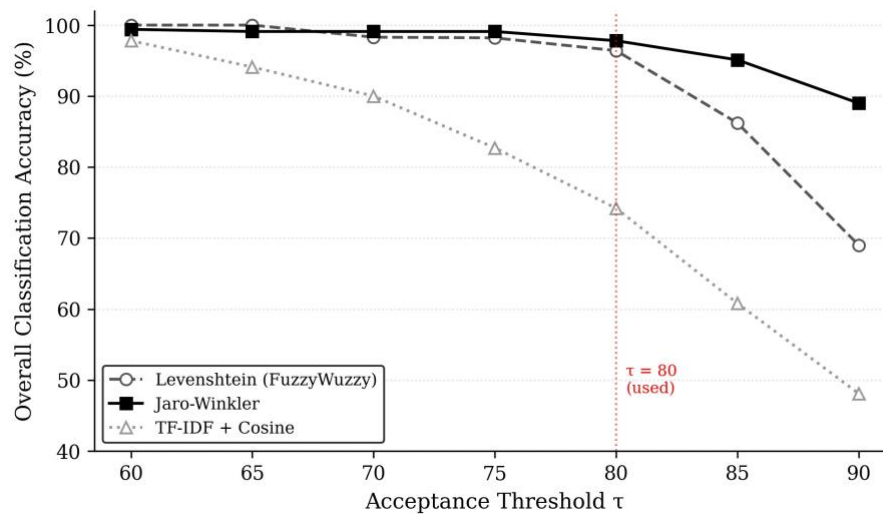


Fig. 6. Overall Classification Accuracy versus Acceptance Threshold τ

4. Conclusion

This study compared three string-similarity algorithms — Levenshtein-based similarity ratio, Jaro-Winkler similarity, and TF-IDF with cosine similarity — for keyword classification in OCR-based library metadata systems, under a controlled, noise-stratified evaluation protocol simulating three levels of OCR-style character corruption on keyword-length queries. Jaro-Winkler similarity achieved the highest overall classification accuracy (97.8%) and the fastest average matching time (22.68 μ s per query); its accuracy advantage over the Levenshtein-based approach commonly deployed in existing mobile OCR library systems (96.4%, 34.45 μ s) was not statistically significant, but its latency advantage was consistent and substantial, while TF-IDF with cosine similarity performed substantially worse on both accuracy (74.2%) and speed (581.01 μ s), particularly under heavy noise. These findings provide evidence-based guidance recommending Jaro-Winkler similarity as a low-risk, latency-favorable alternative to Levenshtein-based matching, and recommending against TF-IDF with cosine similarity, for short-keyword classification tasks in mobile, compute-constrained OCR pipelines.

The limitations of this study should be acknowledged. The noise-injection protocol simulates OCR-style character corruption through random edit operations, which approximates but does not fully replicate the structured error patterns of real OCR engines (such as systematic character confusions, e.g., “rn” misread as “m”). The keyword database was limited to 20 English-language subject categories, and results may differ for larger, multilingual, or domain-specific keyword taxonomies. Because the threshold-based classifier evaluated here (Algorithm 1) performs a linear scan over the keyword database, per-query cost scales as $O(N)$ in the number of candidate keywords N for all three algorithms, with the per-comparison cost differing by algorithm: Levenshtein edit-distance computation is $O(m \cdot n)$ in the two string lengths (Eq. 1), Jaro-Winkler comparison is $O(m+n)$ given its bounded-window character matching (Eqs. 2–3), and TF-IDF cosine similarity is $O(v)$ in the size of the query’s n -gram vector plus per-query vectorization cost (Eqs. 4–5). For the short keywords used here ($m, n \leq \sim 20$ characters), these asymptotic differences are small in absolute terms, so the roughly 17 $\times$ latency gap observed for TF-IDF (Table 2) is attributable primarily to constant-factor overhead — vector construction and floating-point dot products — rather than to a worse asymptotic complexity. At larger

N, however, all three costs still grow linearly in N, and the reported 22–581 μ s per-query latencies were measured at N = 20 keywords only; these figures should not be extrapolated to larger keyword taxonomies without further testing, and scalability to production-sized subject vocabularies (hundreds to thousands of keywords) remains untested. The study is further limited by its use of only five random seeds, which — as noted in Section 3 — gives limited statistical power to confirm or rule out moderate effect sizes such as the Jaro-Winkler–Levenshtein accuracy difference ($d_r = 0.83$, $p = 0.136$); a larger number of seeds would be needed to determine whether this difference reflects a real but underpowered effect or genuine equivalence. Similarly, the near-zero false-positive rate observed here is a property of the 20-keyword database’s low lexical density (Section 3): a larger or more lexically dense taxonomy — for example, one containing near-duplicate or hierarchically related subject terms such as “Biology” and “Biotechnology” — would likely produce a materially higher false-positive rate, and this trade-off between taxonomy granularity and false-positive risk should be characterized directly in future work rather than assumed favorable from the present results. Future work should validate these findings against real OCR-extracted noisy text logs from operational library systems, extend the comparison to additional algorithms such as Damerau-Levenshtein distance [17] and indexing structures such as BK-trees for large-scale keyword databases, and evaluate multilingual keyword classification performance.

For vocational ICT education and practice, these results translate into a concrete, low-cost design recommendation: institutions developing or maintaining mobile OCR-based library metadata systems — a common applied ICT project in polytechnic and vocational software engineering curricula — can adopt Jaro-Winkler similarity as a drop-in replacement for Levenshtein-based matching to obtain both higher classification accuracy and lower on-device latency, without additional training data, model development, or computational infrastructure. This makes the recommendation directly actionable for resource-constrained vocational institutions and small-library deployments, where development time and computing budgets are limited and access to large-scale machine-learning infrastructure is often impractical.

5. Acknowledgment

The authors would like to thank Politeknik Elektronika Negeri Surabaya (PENS) for the support and facilities provided during this research.

6. Author’s Note

The authors hereby declare that there is no conflict of interest related to the publication of this article. Furthermore, the authors confirm that the manuscript is original and free from any form of plagiarism.

7. References

- [1] A. Basofi, S. Sukaridhoto, F. F. F. H. and M. R. Mufid, D. G. Aji Pradana, M. Noer Dafiq, and D. M. Irhamni, “A Mobile-Based Approach to Library Book Cataloguing: Leveraging OCR, AI, and Integrated RPA,” in *3rd Software & Technologies, Visual Informatics & Applications 2025 (SOTVIA 25)*, Nanjing, China, Sep. 2025, p. 10.
- [2] S. Kayalvizhi, N. Akash Silas, R. K. Tarunaa, and S. Pothirajan, “OCR-Based Ingredient Recognition for Consumer Well-Being,” in *Lecture Notes in Networks and Systems*, 2024. doi: [10.1007/978-981-99-6906-7_41](https://doi.org/10.1007/978-981-99-6906-7_41).

- [3] Q. D. Nguyen, N. M. Phan, P. Kromer, and D. A. Le, "An Efficient Unsupervised Approach for OCR Error Correction of Vietnamese OCR Text," *IEEE Access*, vol. 11, 2023, doi: [10.1109/ACCESS.2023.3283340](https://doi.org/10.1109/ACCESS.2023.3283340).
- [4] J. Xiang, O. Salem, and A. Mehaoua, "Keyword-Driven Email Classification: Leveraging Machine Learning Techniques," in *Lecture Notes in Computer Science*, 2026. doi: [10.1007/978-3-032-00552-6_8](https://doi.org/10.1007/978-3-032-00552-6_8).
- [5] J. Waterstraat and L. Walter, "Designing tailored patent search approaches – A case study on nursing care technology," *World Patent Information*, vol. 84, 2026, doi: [10.1016/j.wpi.2025.102420](https://doi.org/10.1016/j.wpi.2025.102420).
- [6] V. B. Lestari and C. A. Hutagalung, "Evaluation of TF-IDF Extraction Techniques in Sentiment Analysis of Indonesian-Language Marketplaces Using SVM, Logistic Regression, and Naive Bayes," *Journal of Computer Science and Applications*, vol. 8, no. 1, 2025.
- [7] P. Sheridan, Z. Ahmed, and A. A. Farooque, "A Fisher's Exact Test Justification of the TF-IDF Term-Weighting Scheme," *American Statistician*, 2025, doi: [10.1080/00031305.2025.2539241](https://doi.org/10.1080/00031305.2025.2539241).
- [8] Y. Onodera and H. Shinnou, "Exploring Data Augmentation for Keyword-Based Document Classification Using Large Language Models," in *Communications in Computer and Information Science*, 2026. doi: [10.1007/978-981-95-4088-4_19](https://doi.org/10.1007/978-981-95-4088-4_19).
- [9] A. Rahmatullah and Q. Annisa, "Application Of TF-IDF And Word2vec For Feature Extraction In Sentiment Analysis Of Free Nutritious Food Policies," *Journal of Computer Electronic and Telecommunication*, vol. 6, no. 2, 2026, doi: [10.52435/complete.v6i2.741](https://doi.org/10.52435/complete.v6i2.741).
- [10] R. Oucheikh, T. Pettersson, and T. Löfström, "Product verification using OCR classification and Mondrian conformal prediction," *Expert Syst. Appl.*, vol. 188, 2022, doi: [10.1016/j.eswa.2021.115942](https://doi.org/10.1016/j.eswa.2021.115942).
- [11] B. Shannaq, "Advancing User Authentication in Cloud Computing: Automated Smart Application as a Multi-Factor Approach Beyond Extended-Levenshtein Distance," *TEM Journal*, vol. 14, no. 2, 2025, doi: [10.18421/TEM142-15](https://doi.org/10.18421/TEM142-15).
- [12] J. Pardede, A. Yudistira, and A. Yudistira, "Comparison of Cosine Similarity, Rabin-Karp, and Levenshtein Distance Algorithms for Plagiarism Detection in Document," *Ultimatics : Jurnal Teknik Informatika*, vol. 17, no. 1, 2025, doi: [10.31937/ti.v17i1.3867](https://doi.org/10.31937/ti.v17i1.3867).
- [13] K. Xia, "Design and application of efficient English learning system based on Jaro-Winkler," *Journal of Computational Methods in Sciences and Engineering*, 2025, doi: [10.1177/14727978251371179](https://doi.org/10.1177/14727978251371179).
- [14] D. A. Anggoro and I. Nurfadilah, "Active Verb Spell Checking Mem- + P in Indonesian Language Using the Jaro-Winkler Distance Algorithm," *Iraqi Journal of Science*, vol. 63, no. 4, 2022, doi: [10.24996/ijs.2022.63.4.38](https://doi.org/10.24996/ijs.2022.63.4.38).
- [15] Rasna, N. W. E. R. Dewi, A. U. Bani, S. Lamsir, and F. Nugroho, "Application Dictionary of Scientific Plants and Animals Android-Based Algorithm Using Jaro Winkler Distance," *International Journal of Engineering, Science and Information Technology*, vol. 5, no. 2, 2025, doi: [10.52088/ijesty.v5i2.839](https://doi.org/10.52088/ijesty.v5i2.839).
- [16] W. Suwarningsih and Nuryani, "Generate fuzzy string-matching to build self attention on Indonesian medical-chatbot," *International Journal of Electrical and Computer Engineering*, vol. 14, no. 1, 2024, doi: [10.11591/ijece.v14i1.pp819-829](https://doi.org/10.11591/ijece.v14i1.pp819-829).

- [17] Y. Chaabi and F. Ataa Allah, "Amazigh spell checker using Damerau-Levenshtein algorithm and N-gram," *Journal of King Saud University – Computer and Information Sciences*, vol. 34, no. 8, pp. 6116–6124, 2022, doi: [10.1016/j.jksuci.2021.07.015](https://doi.org/10.1016/j.jksuci.2021.07.015).
- [18] M. S. M. Rudwan and J. V. Fonou-Dombeu, "Hybridizing Fuzzy String Matching and Machine Learning for Improved Ontology Alignment," *Future Internet*, vol. 15, no. 7, 2023, doi: [10.3390/fi15070229](https://doi.org/10.3390/fi15070229).
- [19] I. Nica, C. Delcea, and N. Chiriță, "Mathematical Patterns in Fuzzy Logic and Artificial Intelligence for Financial Analysis: A Bibliometric Study," *Mathematics*, vol. 12, no. 5, 2024, doi: [10.3390/math12050782](https://doi.org/10.3390/math12050782).
- [20] N. Elmobark, "A Comparative Analysis of Python Text Matching Libraries: A Multilingual Evaluation of Capabilities, Performance and Resource Utilization," *International Journal of Environment, Engineering and Education*, vol. 7, no. 1, pp. 48–60, 2025, doi: [10.55151/ijeedu.v7i1.188](https://doi.org/10.55151/ijeedu.v7i1.188).
- [21] B. Berger, M. S. Waterman, and Y. W. Yu, "Levenshtein Distance, Sequence Comparison and Biological Database Search," *IEEE Transactions on Information Theory*, vol. 67, no. 6, pp. 3287–3294, 2021, doi: [10.1109/TIT.2020.2996543](https://doi.org/10.1109/TIT.2020.2996543).
- [22] I. P. A. E. D. Udayana, I. G. T. A. Putra, I. P. S. Udyana, I. G. S. C. Nugraha, N. P. Widantari, and B. K. Wijaya, "Optimizing Latin to Balinese Script Transliteration: Hybrid Jaro Winkler and Damerau Levenshtein Methods," in *2024 IEEE International Symposium on Consumer Technology (ISCT)*, 2024, pp. 12–18, doi: [10.1109/ISCT62336.2024.10791083](https://doi.org/10.1109/ISCT62336.2024.10791083).

Authors Biographies



Damar Galih Aji received his applied bachelor's degree (D4) in Informatics Engineering from Politeknik Elektronika Negeri Surabaya (PENS), Indonesia. He is currently a Research Assistant at the Human-Centric Multimedia (HCM) Laboratory. His research interests include software engineering, mobile application development, software architecture, artificial intelligence, optical character recognition (OCR), and cross-platform systems. He has been involved in several research and development projects related to intelligent multimedia and software systems.

Email: damargalih@student.pens.ac.id

ORCID: [0009-0004-6316-7091](https://orcid.org/0009-0004-6316-7091)



Sritrusta Sukaridhoto received his B.E. degree in Electrical Engineering, Computer Science Program, from Institut Teknologi Sepuluh Nopember (ITS), Indonesia, in 2002, and his Ph.D. degree in Communication Networks Engineering from Okayama University, Japan, in 2013. He joined Politeknik Elektronika Negeri Surabaya (PENS), Indonesia, as a lecturer in 2002 and became an Assistant Professor in 2011. He was a visiting researcher at Tohoku University, Japan, in 2004. Since 2017, he has served as the Head of the Human-Centric Multimedia Laboratory. His research interests include computer networks, human-centric systems, immersive multimedia, and the Industrial Internet of

Things. He has received several academic awards, best paper awards, and the IEEE Young Researcher Award in 2009. He is a member of IEEE.

Email: dhoto@pens.ac.id

ORCID: [0000-0002-8157-5863](https://orcid.org/0000-0002-8157-5863)



Nirwana Haidar Hari is a lecturer in the Department of Informatics and Computer Engineering at Politeknik Elektronika Negeri Surabaya (PENS), Indonesia. He holds academic degrees in computer engineering and informatics and has extensive experience in teaching, research, and student supervision. His research interests include IoT systems, edge computing, intelligent control systems, data-driven applications, and applied artificial intelligence. He has published articles in national and international journals and actively supervises undergraduate research related to smart systems and digital technologies.

Email: nhh@pens.ac.id

ORCID: [0009-0009-3113-6916](https://orcid.org/0009-0009-3113-6916)