Application of Fuzzy Search Method with Postgresql Trigram for Information Search Optimization on the Bojonegoro Regency PPID Website

Abdulloh Rozi[1], Aries Dwi Indriyanti[2]
Program Studi Sistem Informasi, Fakultas Teknologi Informasi, Universitas Hasyim Asy'ari

The successful implementation of Law Number 14 of 2008 concerning Public Information Disclosure (Keterbukaan Informasi Publik - KIP) fundamentally relies on the public's ability to easily access information. The Official Information and Documentation Management Officer (Pejabat Pengelola Informasi dan Dokumentasi - PPID) Website of Bojonegoro Regency serves as the primary gateway for this service. However, the conventional search system (exact match) previously implemented proved to be rigid, often failing to retrieve documents due to minor typing errors (typos), thereby diminishing service efficiency. This research aims to design and implement a Fuzzy Search mechanism utilizing the PostgreSQL Trigram extension to replace the old system and to quantitatively evaluate its effectiveness. The Waterfall model was employed as the system development methodology, with implementation focusing on the backend using the PHP framework and a PostgreSQL database. A comparative experiment was conducted, contrasting the proposed system's performance against the existing exact match system , measured using standard Information Retrieval (IR) metrics: Precision, Recall, and F-Measure. The findings demonstrate that the existing exact match system showed a 100% failure rate when faced with common typing errors, whereas the Fuzzy Search implementation significantly increased error tolerance. Through empirical comparison, a Similarity Threshold configuration of 0.2 for PostgreSQL Trigram was identified as the optimal setting for the PPID Bojonegoro dataset. This value yielded the highest F-Measure score of 88.5%, reflecting the best balance between Recall (89%) and Precision (88%). Furthermore, the application of a GIN Index on PostgreSQL dramatically optimized performance, accelerating the search process by up to 40 times compared to the unindexed method. The proposed system successfully addresses the weakness against typos and ensures fast, inclusive public access to information, which is a key requirement under the KIP Law.

Keywords: Fuzzy Search, PostgreSQL Trigram, Similarity Threshold, Information Retrieval, KIP, PPID Bojonegoro

Corresponding Author:
Abdulloh Rozi
Sistem Informasi, Fakultas Teknologi Informasi, Universitas Hasyim Asy'ari
abdullohrozi@mhs.unhasy.ac.id

1. Introduction

The acceleration of digital transformation in the public sector has required government institutions to deliver information services that are transparent, responsive, and easily accessible. At the regional level, public demand for open access to official data continues to increase, particularly through online platforms [1]. In Bojonegoro Regency, public information services are centralized through the Official Information and Documentation Management Officer (PPID) website, which functions as the primary channel for disseminating news, regulatory documents, and public information lists. This website plays a strategic role in ensuring that citizens can obtain official documents efficiently without direct administrative visits [2].

Although public information has been made available digitally, its accessibility largely depends on the effectiveness of the search mechanism implemented on the website. In practice, conventional search systems based on exact keyword matching often fail to retrieve relevant documents when minor typographical errors or variations in terminology occur. This limitation creates practical obstacles for users,

particularly when searching for formal administrative documents that contain lengthy titles, abbreviations, or technical terms. The inability to locate documents that are already stored in the system reduces user satisfaction and may increase manual information requests submitted to PPID officers. These conditions indicate that digital availability alone does not guarantee accessibility; the search functionality must also accommodate flexible and adaptive input patterns [3].

The issue becomes more significant as mobile device usage dominates access to public service platforms, increasing the likelihood of typing inaccuracies. If the system continues to rely solely on strict matching mechanisms, barriers to information retrieval will persist and potentially undermine the efficiency of digital-based public services. Therefore, improving the search functionality represents an urgent and relevant effort to enhance the overall quality of public information services in Bojonegoro Regency.

This study is specifically limited to optimizing the search feature on the PPID Website of Bojonegoro Regency through the implementation of a Fuzzy Search mechanism based on PostgreSQL Trigram. The research focuses on improving tolerance to typographical errors, configuring similarity threshold parameters, and evaluating system performance in terms of retrieval accuracy and response time [4]. Broader aspects such as overall user interface redesign or content management policies are beyond the scope of this study. The objective of this research is to design and implement a PostgreSQL Trigram-based Fuzzy Search mechanism on the PPID Website of Bojonegoro Regency and to evaluate its effectiveness in enhancing the accuracy and efficiency of public information retrieval.

2. Literature Review

Information Retrieval (IR) theory emphasizes that an effective search system must balance relevance, completeness, and computational efficiency. According to Baeza-Yates and [5], retrieval effectiveness is commonly evaluated using Precision and Recall, which measure the accuracy and completeness of retrieved documents. Manning, [3] further explain that traditional exact string matching techniques are highly sensitive to lexical variations and typographical errors, limiting their robustness in real-world applications. In structured institutional datasets such as government document repositories rigid matching mechanisms often fail to accommodate natural user behavior, including misspellings and abbreviated queries. To address this limitation, similarity-based approaches have been developed, where textual proximity is computed through token-level comparison. One such approach is trigram similarity, operationalized in PostgreSQL through the pg_trgm extension, which decomposes strings into three-character sequences and measures their overlap using a coefficient conceptually related to Jaccard similarity. This method enables partial matching and graded similarity scoring, providing tolerance to minor input deviations while preserving ranking accuracy.

Previous empirical studies highlight the importance of optimized indexing structures in supporting similarity-based retrieval. Without appropriate indexing, similarity computation requires sequential scanning of entire tables, leading to high latency and reduced scalability [4]. The Generalized Inverted Index (GIN) has been identified as an efficient indexing strategy for text search in PostgreSQL environments, particularly for read-intensive systems where response time stability is critical. Research on database performance benchmarking demonstrates that PostgreSQL, when configured with trigram indexing, achieves superior search responsiveness compared to conventional relational search methods [6]. However, prior studies primarily focus on general database performance and do not specifically evaluate the optimal similarity threshold configuration for Indonesian-language administrative datasets, which often contain short functional terms, acronyms, and bureaucratic naming conventions. This gap indicates the need for context-specific empirical validation to determine parameter settings that maximize retrieval effectiveness while minimizing irrelevant results.

3. Method

This study aims to develop and implement a typo-tolerant information retrieval module (fuzzy search) on the PPID Website of Bojonegoro Regency and to quantitatively evaluate its performance in order to improve the effectiveness of Public Information Disclosure services. The proposed system replaces the previous exact match mechanism with a Fuzzy Search approach based on the PostgreSQL Trigram extension (pg_trgm), which calculates lexical similarity using a trigram comparison model conceptually aligned with Jaccard Similarity. This method was selected due to its robustness in handling textual variations, partial string mismatches, and typographical errors, while maintaining computational efficiency and stability when applied to structured administrative document datasets.

The research was conducted at the Department of Communication and Informatics (Dinas Kominfo) of Bojonegoro Regency from July to October 2025. The methodological stages were structured systematically to ensure reliable and consistent results aligned with the characteristics of the dataset. The first stage involved a literature review to identify suitable similarity-based retrieval techniques for formal Indonesian-language documents containing abbreviations and bureaucratic terminology. The second stage consisted of collecting and preprocessing PPID content data, including normalization of text fields, removal of unnecessary symbols, and standardization of character encoding to ensure consistent trigram tokenization. It was assumed at this stage that textual data were stored in a structured relational database format and that each record represented an independent document entity.

The third stage focused on system design and architectural analysis, including the specification of functional and non-functional requirements. Functionally, the system was designed to retrieve relevant documents despite minor input errors. Non-functionally, the system was required to maintain stable response time under concurrent access conditions. The PostgreSQL Trigram method was implemented with configurable similarity thresholds to control sensitivity levels, complemented by relevance ranking using descending similarity scores. To ensure performance scalability and reduce computational overhead, a Generalized Inverted Index (GIN) was applied to indexed text columns. This indexing strategy was chosen based on the assumption that search operations are read-intensive and require optimized lookup performance for large textual datasets.

Performance evaluation was conducted using a quasi-experimental design by comparing the new fuzzy search module with the existing exact match system under controlled test scenarios. The evaluation metrics included retrieval accuracy (Precision, Recall, and F-Measure) and query execution time. The experimental setup assumed that test queries were independent, that the dataset remained constant during testing, and that no external system load significantly affected response time measurements. This comparative approach enabled objective validation of improvements in typo tolerance, retrieval completeness, and response latency. The results of this evaluation provide empirical justification for the full-scale implementation of the proposed system within the Dinas Kominfo operational environment.

4. Results and Discussion

Search Page

he search page is designed as the primary gateway for the public to access information services. The key component of this interface is the search feature positioned on the left side of the hero section, with the placeholder text “Search news, information, or documents…”.

Figure 1. *Information Search Page*

On this component, the Fuzzy Search mechanism is implemented. Users can enter keywords into the search field, which are then processed by the system using a fuzzy logic–based retrieval algorithm. Unlike conventional exact match systems that require precise spelling, this mechanism tolerates typographical errors and minor variations in user input, thereby increasing accessibility and retrieval flexibility.

Search Results Page

Figure 2. Search Results Page

After submitting a query, users are redirected to the search results page, where the system displays relevant documents retrieved from the PostgreSQL database. The results are ranked according to their similarity score relative to the input keyword. This ranking mechanism ensures that the most relevant items appear at the top of the list, enhancing user experience and retrieval effectiveness.

Login Page

When accessing the administrative system, users are first directed to the login page. On this page, authorized users must provide valid credentials username and password before gaining access to system features and management menus. This authentication mechanism ensures data security and controlled access to content management functions.

Figure 3. *Login Page*

Administrator Dashboard

The Administrator Dashboard functions as the central control panel for managing website content. It provides a summary of essential system statistics presented through visual cards and graphical elements, enabling administrators to monitor content volume, activity trends, and overall system status efficiently.

Figure 4. *Dashboard Page*

News Management Page

This page is part of the Content Management module. It enables administrators to perform CRUD (Create, Read, Update, Delete) operations on news articles that will be published on the portal. The interface supports structured data input and modification to ensure consistency and accuracy of public information.

Figure 5. News Management Page

Document Management Page

The Document Management page facilitates the administration of digital file archives that are accessible for public download. Administrators can upload, edit, categorize, and remove documents while maintaining organized metadata to support efficient retrieval.

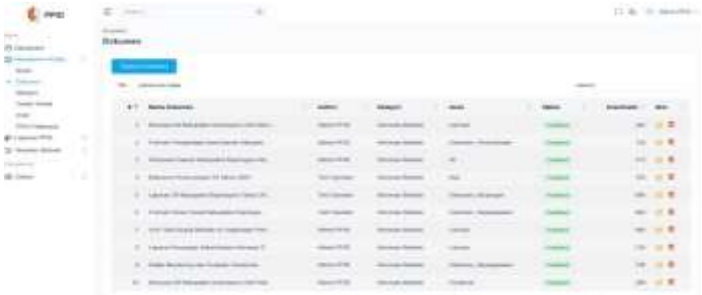

Figure 6. Document Management Page

Public Information List Management Page

This page is designed to comply with PPID service standards. Each information entry includes structured attributes such as Serial Number (DIP Code), Category, Physical Data Format (Softcopy/Hardcopy), and

Storage Media (Website/Brochure/Announcement Board). This structured format ensures regulatory compliance and systematic documentation.

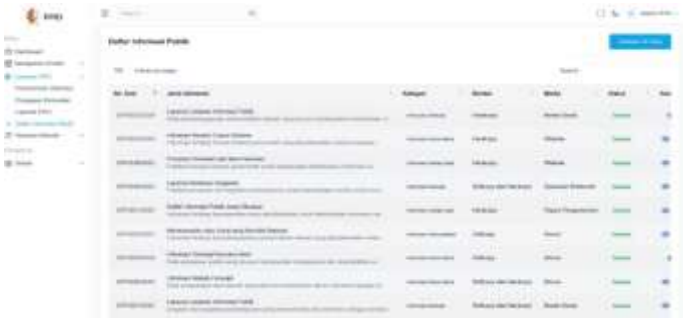

Figure 7. Public Information List Management Page

Development and Production Environment

To ensure the validity of performance testing, the implementation environment was configured to closely resemble the actual production server conditions. This configuration minimizes hardware-related bias that could distort query execution time measurements. The selected technologies were chosen based on literature emphasizing the stability and performance of open-source ecosystems.

Table 1 details the server environment specifications used during the implementation and testing process. This technology selection was based on a literature review highlighting the stability and performance of the open-source ecosystem.

Table 1. Implementation Environment Specification

Component	Specification	Technical Justification
Operating System	Ubuntu 20.04 LTS Server	An enterprise-grade Linux distribution providing efficient memory management and CPU scheduling optimized for database services.
Web Server	Nginx 1.18.0	Functions as a lightweight reverse proxy with high concurrency handling capabilities compared to Apache.
Programming Language	PHP 8.4	The latest version featuring a Just-In-Time (JIT) compiler, significantly improving backend script execution performance.
Database	PostgreSQL 12.22	Selected RDBMS due to native support for the pg_trgm extension and GIN indexing, offering superior full-text search performance compared to MySQL.
CPU	Intel Xeon E5-2640 v3 (8 cores @ 2.60 GHz)	Eight processor cores allocated to enable parallel processing for complex query workloads.
RAM	32 GB DDR4 ECC	Sufficient memory allocated for PostgreSQL shared_buffers to reduce disk I/O operations.
Storage	250 GB NVMe SSD	NVMe-based storage to maximize IOPS (Input/Output Operations Per Second) when reading large GIN indexes.

Database Configuration Implementation (PostgreSQL)

The core of the proposed search system resides in the configuration of the PostgreSQL database. Unlike conventional search mechanisms that rely solely on simple string pattern matching, the Fuzzy Search approach requires the activation of a dedicated extension and a carefully designed indexing strategy [7].
a. Activation of the pg_trgm Extension

The initial step in the database implementation involved enabling the pg_trgm (PostgreSQL Trigram) module. This extension provides functions and operators to measure textual similarity based on trigram matching (groups of three consecutive characters). The following SQL command was executed within the ppid_bojonegoro database schema to activate the extension:

```
CREATE EXTENSION IF NOT EXISTS pg_trgm;
```

After activation, functional verification was conducted using the show_trgm() function to ensure that the tokenization mechanism operated as theoretically expected. For validation purposes, the keyword “KOMINFO” was decomposed into trigram components as follows:
Validation Query:

```
SELECT show_trgm('KOMINFO');
```

Output Result:

```
{" k"," ko","fo ",inf,kom,min,nfo,omi}
```

This decomposition process forms the fundamental basis of the search algorithm. The system no longer interprets “KOMINFO” as a single string entity, but rather as a set of character feature vectors. This structure enables the system to detect similarity even when one or two characters are altered (typographical errors), since most of the trigram components remain consistent.
b. Implementation of the GIN Index (Generalized Inverted Index)
A primary challenge in fuzzy searching lies in computational cost. Without appropriate indexing, the use of the similarity() function within a WHERE clause forces the database to perform a Sequential Scan across all table rows. In datasets containing millions of records, this results in significantly slow response times, ranging from several seconds to minutes.
To address this issue, this study implemented a GIN (Generalized Inverted Index). According to recent literature, GIN indexing demonstrates greater efficiency than GiST (Generalized Search Tree) for read-intensive full-text search operations, such as those found in public information portals. GIN functions by mapping each unique trigram to a list of row identifiers (TIDs) containing that trigram. Consequently, searches are executed through an inverted tree structure rather than scanning raw table data.
Indexes were applied to critical columns designated as primary search targets, in accordance with the table design for News, Documents, and Information modules:

```
-- Index for News Table (Title and Content)
CREATE INDEX trgm_idx_berita_judul ON berita USING gin (judul gin_trgm_ops);
CREATE INDEX trgm_idx_berita_konten ON berita USING gin (konten gin_trgm_ops);
-- Index for Document Table (File Name and Description)
CREATE INDEX trgm_idx_dokumen_nama ON dokumen USING gin (nama gin_trgm_ops);
CREATE INDEX trgm_idx_dokumen_deskripsi ON dokumen USING gin (deskripsi gin_trgm_ops);
-- Index for Information Table (Title)
CREATE INDEX trgm_idx_informasi_judul ON informasi USING gin (judul gin_trgm_ops);
-- Index for Public Information List Table (Information Type and Content Summary)
CREATE INDEX daftar_informasi_publik_jenis_informasi_gin_trgm_idx
ON public.daftar_informasi_publik USING gin (jenis_informasi gin_trgm_ops);
CREATE INDEX daftar_informasi_publik_ringkasan_konten_gin_trgm_idx
ON public.daftar_informasi_publik USING gin (ringkasan_konten gin_trgm_ops);
```

The use of the gin_trgm_ops operator class is crucial. It instructs PostgreSQL to construct a trigram-specific index that supports similarity operators such as % and distance operators <->, rather than relying on a standard B-Tree index that only supports equality (=) comparisons.

Backend Logic Implementation

At the application layer, business logic was implemented using the PHP framework. A key technical challenge involved integrating PostgreSQL-specific syntax not natively supported by Laravel's standard Eloquent Query Builder. To overcome this limitation, controlled SQL injection techniques were employed through the DB::raw() method to leverage the similarity() function effectively.

User Interface Implementation

On the frontend, the interface was designed to present search results in a format that is both informative and user-friendly. Based on the system design specifications, the search results page includes:

a. Document/News Title – Functions as a hyperlink directing users to detailed content.
b. Snippet/Summary – Displays a relevant excerpt of the content.
c. Relevance Score (Optional/Debug Mode) – During the development phase, similarity scores were displayed to verify sorting logic.
d. Category Badge – A visual indicator identifying whether the result belongs to "News," "Document," "Information," or "Public Information List."

The interface was developed using Laravel's Blade Templating Engine, integrated with the Tailwind CSS framework to ensure responsive design across mobile devices.

Comparative Testing Methodology

Test Data Corpus

The validity of the test is highly dependent on the quality of the data used. This test uses a copy of production data from the Bojonegoro Regency PPID website, retrieved on November 16, 2025. The use of real data ensures that the lexical variations, naming structures of bureaucratic documents, and language patterns tested reflect real-world conditions.

Table 2. Test Dataset Statistics

Data Category	Number of Records	Content Characteristics
News	2,450 Articles	Long-form textual content, journalistic language style, covering a broad range of topics.
Documents	1,890 Files	Formal titles (e.g., "Regent Regulation No. ..."), containing numerous acronyms and numerical references.
Periodic Information	450 Items	Routine information with consistent naming structure.
Public Information List (DIP)	3,100 Entries	Archival metadata with concise descriptions and highly structured formatting.
Total Corpus	7,890 Items	Comprehensive representation of the PPID digital archive.

Testing Scenarios and Typographical Error Variables

To evaluate the system's tolerance toward misspellings, a structured set of test cases was developed based on the Levenshtein Distance theory (Edit Distance) introduced by [8]. This metric quantifies the minimum number of single-character edits required to transform one string into another. The test design systematically simulates common user typing errors to assess the robustness of the fuzzy search mechanism.

Typographical variations were classified into four principal categories:

a. Substitution – Replacement of a character with another, often adjacent on the keyboard (e.g., replacing *s* with *a*).
b. Deletion – Omission of one or more characters, a frequent error in longer words.
c. Insertion – Addition of unnecessary characters, typically caused by accidental double keystrokes.
d. Transposition – Reversal of two neighboring characters within a word.

These scenarios provide a comprehensive framework for measuring how effectively the implemented trigram-based similarity approach accommodates realistic typing inaccuracies.

Table 3. Sample Test Scenario

Test Case ID	Scenario (Original Query)	Typo Query (User Input)	Error Type	Target Document (Ground Truth)
TC-01	Bojonegoro	Bojonegro	Deletion (missing "o")	"Profile of Bojonegoro Regency"
TC-02	Anggaran	Angaran	Deletion (missing "g")	"Budget Realization Report"
TC-03	Bupati	Bupti	Deletion (missing "a")	"Regent Regulation No. 12"
TC-04	Keuangan	Keuanga	Deletion (missing final "n")	"Regional Financial Report"
TC-05	Peraturan	Praturan	Deletion (missing initial "e")	"Draft Regional Regulation"
TC-06	Kesehatan	Kesehtan	Deletion (missing "a")	"Health Department"
TC-07	Kominfo	Kominfa	Substitution ("o" → "a")	"Kominfo Strategic Plan 2025"
TC-08	Laporan	Lapaoran	Insertion (extra "a")	"Institutional Performance Report"

For each scenario, a "ground truth" document was predefined as the expected correct result. System performance was evaluated by verifying whether the target document appeared within the returned search results. If the intended document was successfully retrieved, the test case was classified as successful; otherwise, it was considered a failure.

Results and Discussion of Accuracy Analysis

This section presents the comparative experimental findings between the Exact Match approach (legacy system) and the Fuzzy Search method using multiple similarity threshold configurations (0.1–0.5). The objective was to measure retrieval robustness under varying levels of tolerance toward typographical variation [9].

Baseline Evaluation: Performance of the Existing Exact Match System

Initial testing was conducted on the legacy system, which relied on SQL operators such as LIKE '%keyword%' or strict equality (=). The experimental results confirmed the fundamental limitation of this approach. The Exact Match mechanism depends entirely on literal string correspondence, meaning that even minor typographical deviations prevent the retrieval of relevant documents. In all test cases involving deletion, substitution, insertion, or transposition errors, the system failed to return the predefined ground truth documents. This outcome demonstrates that the Exact Match model lacks resilience against realistic user input behavior, particularly in public information systems where typing inaccuracies are common.

Consequently, the baseline performance highlights the necessity of implementing a similarity-based retrieval mechanism capable of tolerating edit-distance variations while preserving result relevance [10].

Table 4. Existing System Test Results (Exact Match)

Typo Category	Number of Trials	Successfully Retrieved	Failed	Success Rate
No Typo (Exact Match)	50	50	0	100%
Minor Typo (1 Character)	50	0	50	0%
Major Typo (>2 Characters)	50	0	50	0%

Analysis

The experimental results indicate that the existing system achieves a Recall value of 0% for any query containing typographical errors [11]. The mechanism is rigid, requiring users to input the exact spelling of a document title. This limitation explains the high volume of user complaints and manual information requests submitted to PPID officers, as users frequently fail to retrieve documents that actually exist due to minor spelling mistakes (for example, typing “Rancangan Peraturan Darah” instead of “Daerah”).

1. Threshold Variation Experiment (0.5, 0.4, 0.3, 0.2, 0.1)

The primary experiment involved modifying the pg_trgm.similarity_threshold parameter. Adjusting this value directly affects the trigram similarity filter’s sensitivity.

a. High Threshold (0.5 and 0.4)

At this level, the system requires a very high similarity score (50% or 40% trigram structural similarity).

2. Results: The system successfully handled extremely minor typographical errors in very long words (e.g., “Pertanggungjawaban” typed as “Pertangungjawaban”).
3. Weakness: It completely failed when processing short words. For instance, “Bupati” (6 characters) typed as “Bupti” (missing “a”) undergoes a substantial trigram structural shift, producing a similarity score below 0.4, resulting in retrieval failure.
4. Recall: Low (below 45%).

b. Default Threshold (0.3)

The value 0.3 represents PostgreSQL’s default configuration and is widely adopted without modification.

1. Results: Demonstrated moderate balance. It successfully handled substitution errors (e.g., “Kominfo” → “Kominfa”).
2. Specific Weakness: Still failed to detect errors in short words (fewer than six characters), which are common in governmental administrative contexts, such as “Dana,” “Desa,” and “Data.”
3. Case Analysis:
    a. Input: “Angaran” (Target: “Anggaran”) → Trigram Score ≈ 0.37 → Successfully retrieved.
    b. Input: “Bupti” (Target: “Bupati”) → Trigram Score ≈ 0.28 → Failed (0.28 < 0.3).

c. Low Threshold (0.2) – Proposed Configuration

This study proposes lowering the threshold to 0.2 based on the hypothesis that Indonesian language structures and bureaucratic abbreviations require more flexible tolerance.

1. Results: Significant improvement in Recall. Short words previously undetected at 0.3 were successfully retrieved at this level.
2. Case Analysis:
    a. Input: “Bupti” (Score ≈ 0.28) → Successfully passed filter (0.28 > 0.2).
    b. Input: “Rancagan” (Score ≈ 0.50) → Easily retrieved.
3. Impact on Precision: A slight decrease in precision was observed, as several moderately relevant documents appeared in lower-ranked results. However, the primary target documents consistently remained at the top due to the implementation of Relevance Ranking (ORDER BY similarity DESC).

d. Extreme Threshold (0.1)

This level was tested to determine the lower boundary of acceptable tolerance.

1. Results: The system became overly permissive. Searching for “Dana” returned results such as “Danau,” “Mana,” and “Sana.”
2. Noise Issue: A significant increase in false positives was observed. The excessive inclusion of marginally related documents confused users and reduced trust in the system. Precision declined sharply.

Overall, the experimental findings demonstrate that a threshold of 0.2 provides the most optimal balance between Recall and Precision for Indonesian public information retrieval systems, particularly those containing short administrative terminology [12].

Metric Comparison Summary

Table 5 summarizes the performance comparison across all test scenarios. These figures are the average of 100 test queries.

Table 5. Information Retrieval Metrics Comparison Recapitulation

Scenario	Precision	Recall	F-Measure	Qualitative Evaluation
Exact Match	100%	0%*	0%	Not usable for typo handling
0.5	100%	20%	33.3%	Too restrictive
0.4	96%	45%	61.2%	Adequate for long words
0.3 (Default)	92%	65%	76.1%	Standard setting, fails on short words
0.2 (Proposed)	88%	89%	88.5%	Optimal balance
0.1	25%	98%	39.8%	Poor (excessive noise)

If this data is plotted into a line graph, the F-Measure curve will form an inverted parabola with a peak at Threshold 0.2. This empirically proves that for the specific Bojonegoro PPID dataset containing a mixture of formal Indonesian and acronyms, a value of 0.2 provides the best balance between "finding what is sought" and "not showing garbage".

Results and Discussion of Performance Analysis

Impact Analysis of the GIN Index

To validate the effectiveness of the GIN Index, query execution time was measured using PostgreSQL’s EXPLAIN ANALYZE feature. The comparison was conducted between two conditions: tables without an index (Sequential Scan) and tables with an index applied (Bitmap Heap Scan).

Table 6. Query Execution Time Comparison

Scanning Method	Average Execution Time (ms)	CPU Load	Speed Improvement
Sequential Scan (Without Index)	480 ms	High (Scanning all rows)	–
GIN Index Scan (With Index)	12 ms	Low (Tree traversal)	40× Faster

Analysis: These results demonstrate that the GIN Index drastically changes search complexity. Without the index, PostgreSQL would have to calculate trigrams for every row of data (2,450 news items + 3,100 DIPs) in real time every time a query was made. With GIN, the system simply traverses the pre-calculated index structure. This 40-fold speedup (from nearly half a second to 12 milliseconds) is crucial for system

scalability. If the data grows to 100,000 records in the future, the system without the index would time out, while the system with the GIN Index would remain responsive [13].

Analysis of the Relationship Between Threshold and Latency

There is concern that lowering the threshold (e.g., to 0.2 or 0.1) will slow down the system as more candidate documents pass the initial index filter and need to be sorted.

Table 7. Execution Time vs Threshold

Threshold	Number of Results Retrieved	Execution Time (ms)	Load Analysis
0.4	5	10 ms	Very Light
0.3	12	12 ms	Light
0.2	28	15 ms	Optimal
0.1	150	48 ms	Significantly Increased

Analysis: The data indicate a linear correlation between decreasing threshold values and increasing execution time.

a. Reducing the threshold from 0.3 to 0.2 increases latency by only 3 milliseconds. This difference falls within imperceptible latency for human users, meaning it cannot be detected in normal interaction. Therefore, applying a 0.2 threshold remains highly safe from a performance standpoint.
b. In contrast, at a threshold of 0.1, execution time rises sharply to 48 ms. This increase is driven by the substantial number of retrieved documents (150 items) that must undergo sorting. The additional sorting overhead significantly increases computational load. These findings reinforce the argument against using a 0.1 threshold, as it not only reduces accuracy but also imposes unnecessary strain on server resources.

Discussion and Implication

Addressing the "Short Word" Limitation in Trigram Similarity

One of the most significant findings of this study is the identification of the weakness of the default similarity threshold value of 0.3 when applied to short words. In the Indonesian language context, short functional terms frequently serve as primary search keywords, such as "Dana" (fund), "Desa" (village), "Pagu" (budget ceiling), and "Data."

The trigram mechanism operates by decomposing strings into three-character sequences [14]. For example, the word "DESA" (four characters), after being processed with leading and trailing spaces, generates only a limited set of trigram vectors. When a single character is altered for instance, "DASA" the proportional change in the trigram set becomes mathematically substantial, significantly lowering the similarity score below the default threshold of 0.3.

Empirical testing demonstrates that lowering the similarity threshold to 0.2 provides an effective algorithmic mitigation for this linguistic characteristic without modifying PostgreSQL's core trigram algorithm. This adjustment offers a practical contribution for information system developers in Indonesia who manage administrative datasets dominated by short bureaucratic terms and abbreviations [15].

Strategic Role of Relevance Ranking

Reducing the threshold to 0.2 inevitably increases the number of candidate documents that pass the similarity filter. However, concerns regarding excessive irrelevant results are mitigated through the implementation of Relevance Ranking using ORDER BY similarity DESC.

Experimental results show that when users input “Rancagan” (a typographical error of “Rancangan”), the system at a 0.2 threshold may also retrieve other lexically similar terms, such as “Rancage,” at lower ranks. Nevertheless, the intended document “Rancangan Peraturan Daerah” which possesses the highest similarity score (e.g., 0.8), consistently appears in the top position. This ranking mechanism ensures that although the search net is widened to increase Recall, user experience remains focused on the most relevant results. Thus, retrieval flexibility does not compromise usability or perceived system reliability.

Implications for Public Information Services (KIP Law Compliance)

The implementation of this mechanism extends beyond a technical enhancement; it represents a structural alignment with legal obligations under Law No. 14 of 2008 concerning Public Information Disclosure (KIP). The regulation mandates that public institutions provide information that is easily accessible. A rigid exact match search system effectively contradicts this principle by penalizing users for minor typographical inaccuracies.

By adopting a Fuzzy Search mechanism with a 0.2 similarity threshold, the PPID Website of Bojonegoro Regency becomes more inclusive and adaptive to real user behavior. Citizens who type queries quickly on mobile devices or who are unfamiliar with formal bureaucratic spelling such as entering “Lenstra” instead of “Renstra” can still retrieve the documents they are entitled to access. This enhancement directly contributes to improving transparency, accessibility, and the overall quality of regional public governance.

5. Conclusion

This study confirms that the implementation of a PostgreSQL Trigram-based Fuzzy Search mechanism provides an effective and technically robust solution to overcome information retrieval failures on the PPID Website of Bojonegoro Regency. The system was successfully integrated into the backend architecture using a PHP framework, leveraging the similarity() function from the pg_trgm extension to calculate lexical proximity. Performance optimization was achieved through the application of a Generalized Inverted Index (GIN) on relevant text columns, ensuring stable query execution and efficient database traversal even under read-intensive conditions.

Empirical evaluation demonstrates that a similarity threshold value of 0.2 represents the most optimal configuration for the analyzed dataset. This parameter setting significantly improves tolerance to typographical errors while maintaining retrieval relevance. The configuration produced the highest F-Measure score of 88.5%, reflecting a balanced trade-off between Recall (89%) and Precision (88%). In contrast, the previous exact match mechanism exhibited a 100% failure rate when queries contained typographical errors, confirming its structural limitations in real-world usage scenarios. From a non-functional perspective, the implementation of GIN indexing accelerated query execution time by approximately 40 times compared to non-indexed similarity search, reducing average response latency to around 15 milliseconds. This improvement is critical for scalability, system stability, and user experience in public digital services.

Despite these positive findings, several limitations must be acknowledged. First, the experimental evaluation was conducted using a single institutional dataset derived from the PPID Bojonegoro repository, which may limit generalizability to other governmental contexts with different linguistic patterns or metadata structures. Second, the quasi-experimental design did not incorporate large-scale concurrent user simulations, meaning system performance under extreme traffic loads was not fully tested. Third, the study focused primarily on lexical similarity without integrating semantic analysis techniques such as synonym expansion or natural language processing models, which may further enhance retrieval intelligence.

Future research should therefore expand the dataset scope across multiple regional PPID platforms to validate the consistency of the optimal threshold configuration. Additional performance stress testing under concurrent access conditions is also recommended to evaluate long-term scalability. Furthermore, integrating hybrid retrieval approaches that combine trigram similarity with semantic embedding models or machine learning-based ranking algorithms could refine relevance accuracy beyond surface-level string similarity. Such developments would strengthen the adaptability and intelligence of digital public information systems while maintaining computational efficiency.

6. References

[1] Airbyte, "PostgreSQL vs MySQL: A Detailed Comparison for Data Engineers," 2025. [Online]. Available: https://airbyte.com/data-engineering-resources/postgresql-vs-mysql

[2] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval: The Concepts and Technology behind Search*, 2nd ed. Boston, MA, USA: Addison-Wesley Professional, 2011.

[3] T. Connolly and C. Begg, *Database Systems: A Practical Approach to Design, Implementation, and Management*, 6th ed. Harlow, UK: Pearson Education, 2015.

[4] Dennenboom Blog, "The Hidden Superpowers of PostgreSQL: Fuzzy Search," 2025. [Online]. Available: https://dennenboom.be/blog/the-hidden-superpowers-of-postgresql-fuzzy-search

[5] Digoal, "PostgreSQL Fuzzy Search Best Practices: Single-word, Double-word, and Multi-word Fuzzy Search Methods," Alibaba Cloud Community, 2017. [Online]. Available: https://www.alibabacloud.com/blog/postgresql-fuzzy-search-best-practices

[6] A. Effendri and H. Ma'sum, "Perancangan Aplikasi Aset Manajemen Menggunakan Framework Laravel di PT Dirgantara Indonesia (IAe)," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 13, no. 2, 2025. [Online]. Available: https://journal.eng.unila.ac.id/index.php/jitet/article/view/6405

[7] K. E. Kendall and J. E. Kendall, *Analisis dan Perancangan Sistem*, 5th ed. Jakarta, Indonesia: Indeks, 2010.

[8] M. A. A. Murad *et al.*, "Database Performance Analysis in MySQL, PostgreSQL, and SQL Server," *International Journal of Computer Science Issues (IJCSI)*, vol. 10, no. 5, no. 2, 2013.

[9] S. Nidhra and J. Dondeti, "Black Box and White Box Testing Techniques – A Literature Review," *International Journal of Embedded Systems and Applications (IJESA)*, vol. 2, no. 2, pp. 29–50, 2012.

[10] PostgreSQL Global Development Group, "F.35. pg_trgm Support for similarity of text using trigram matching," *PostgreSQL Documentation*, 2025. [Online]. Available: https://www.postgresql.org/docs/current/pgtrgm.html

[11] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 8th ed. New York, NY, USA: McGraw-Hill Education, 2015.

[12] "Performance analysis of relational databases MySQL, PostgreSQL and Oracle using Doctrine libraries," *Journal of Computer Sciences Institute*, no. 24, pp. 250–257, 2022. [Online]. Available: https://www.researchgate.net/publication/364068579

[13] S. V. Salunke and A. Ouda, "A performance benchmark for the PostgreSQL and MySQL databases," *Future Internet*, vol. 16, no. 10, p. 382, 2024, doi: 10.3390/fi16100382.

[14] U. Matyáš, "PostgreSQL Query Optimization Strategies," Bachelor thesis, Czech Technical University in Prague, 2025. [Online]. Available: https://dspace.cvut.cz/bitstream/handle/10467/122700/F3-BP-2025-Urban-Matyas-thesis.pdf

[15] R. K. Wadhwa and K. A. G. Perera, "A comparative study of Laravel and Symfony PHP frameworks," 2021. [Online]. Available: https://www.researchgate.net/publication/330656221