IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.155154

A Modular Data Deduplication Framework with Support for Multi-Format Document Analysis

Shreya Modi, Sreenivasa M, Shridevi, Shreya Y.P, Trisha Prameela Y

Department of Computer Science and Engineering, Ballari Institute of Technology & Management, Ballari, India

Abstract: Data deduplication is a critical technique used to reduce repeated storage of documents in digital systems. In many organizational and academic environments, identical files or portions of files are stored multiple times, leading to increased storage consumption and management complexity. This research presents a comprehensive document-based deduplication framework that provides native support for PDF, DOCX, and plain text file formats. The proposed system processes uploaded documents by extracting textual content and systematically dividing it into smaller, manageable chunks. A cryptographic hash-based comparison methodology is employed to determine whether content segments already exist within the storage repository. When duplicate content is identified, the system maintains reference pointers rather than storing redundant copies, thereby achieving significant storage optimization. Experimental evaluation demonstrates that the chunk-level approach successfully identifies partial duplicates that would be missed by traditional file-level comparison methods. The framework is designed for practical deployment in academic institutions and organizational document management systems where efficient handling of duplicate content is essential.

Index Terms: Data deduplication, document analysis, hashbased comparison, chunk-level processing, storage optimization, multi-format support.

I. INTRODUCTION

Digital documents constitute a fundamental component of information management in contemporary academic institutions, corporate offices, and various organizations. The proliferation of digital content has led to an exponential increase in storage requirements, with significant portions attributed to redundant data. When documents are shared, edited by multiple users, and stored across different systems, the same content frequently appears in multiple locations, creating substantial inefficiencies in storage utilization and document management workflows.

The challenge of duplicate content extends beyond simple storage waste. Organizations face difficulties in maintaining data consistency, tracking document versions, and ensuring efficient backup and recovery processes. In academic environments, the management of student submissions, research papers, and administrative documents presents similar challenges where identical or near-identical content appears across numerous files.

A. Research Question

This study addresses the following research question: *How can a modular data deduplication system effectively identify and eliminate duplicate content across multiple document formats using chunk-based hashing techniques while maintaining system simplicity and practical applicability?*

This question encompasses several sub-objectives:

- Developing a unified text extraction mechanism for heterogeneous document formats
- Implementing an efficient chunk-based content segmentation strategy
- Designing a hash-based duplicate detection algorithm with high accuracy
- Creating a reference-based storage system that eliminates redundant data
- Evaluating the framework's effectiveness in real-world document analysis scenarios

B. Novelty and Contribution

The primary contributions of this research are:

1) Chunk-Level Analysis: Unlike traditional file-level deduplication systems, our framework operates at the chunk level, enabling detection of partial duplicates where only portions of documents are repeated.
2) Multi-Format Support: The system provides integrated support for PDF, DOCX, and plain text formats, addressing the heterogeneity of document repositories in practical settings.
3) Modular Architecture: The framework's modular design facilitates easy extension and modification without requiring fundamental changes to the core system.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026
DOI: 10.17148/IJARCCE.2026.155154

4) Practical Applicability: The system is designed for deployment in academic and organizational environments with standard computing resources, without requiring specialized hardware.

C. Paper Organization

The remainder of this paper is organized as follows: Section II presents a comprehensive literature review of existing deduplication techniques. Section III describes the proposed system architecture. Section IV details the methodology employed. Section V presents the algorithms used. Section VI discusses experimental results. Section VII explores practical applications. Section VIII analyzes advantages and limitations. Section IX concludes the paper and outlines future research directions.

II. LITERATURE REVIEW

Data deduplication has been an active area of research, with numerous approaches proposed to improve storage efficiency and reduce redundancy in digital systems.

A. Block-Level Deduplication Systems

Zhu et al. [1] introduced a data-domain deduplication file system designed to eliminate disk bottlenecks by removing redundant blocks during storage operations. Their approach demonstrated significant improvements in storage efficiency by identifying and eliminating duplicate data blocks before they are written to disk. However, the computational overhead associated with processing large datasets presented scalability challenges, particularly in environments with high data ingestion rates.

Lillibridge et al. [2] proposed sparse indexing as a solution for large-scale inline deduplication. This methodology improved system scalability by reducing memory requirements through selective indexing strategies. The approach proved effective for structured data but showed limitations when processing unstructured document content where traditional block boundaries may not align with semantic content boundaries.

B. Primary Data Deduplication

El-Shimi et al. [3] conducted an extensive study on primary data deduplication systems, demonstrating significant storage reduction capabilities in enterprise environments. Their research provided valuable insights into the practical implementation of deduplication at scale. However, the focus on blocklevel redundancy did not address the specific requirements of document-level similarity analysis, where semantic content relationships play a crucial role.

C. Chunking Strategies

Meyer and Bolosky [4] performed a comprehensive analysis of practical deduplication techniques, examining the tradeoffs between chunk size, processing performance, and storage efficiency. Their work highlighted the importance of selecting appropriate chunking strategies based on data characteristics. The study revealed that fixed-size chunking, while computationally efficient, may miss duplicate detection opportunities when content shifts occur within files.

Eshghi and Tang [5] developed a framework for analyzing and improving content-based chunking algorithms. Their research demonstrated that content-defined chunking could improve duplicate detection rates by adapting chunk boundaries to content characteristics. This approach provides better alignment with document structure but introduces additional computational complexity.

D. Text Mining and Document Similarity

Recent research has explored text mining and document similarity detection using natural language processing techniques [10]. These approaches can detect semantic similarity between documents with different wording, addressing a limitation of hash-based methods. However, the computational complexity associated with semantic analysis makes these techniques less suitable for large-scale storage optimization applications where processing speed is critical.

Manning et al. [8] provided foundational work on information retrieval techniques applicable to document analysis. Their contributions to text processing and similarity measurement inform modern approaches to document deduplication, though the focus on search and retrieval differs from storage optimization objectives.

E. Research Gap

The existing literature reveals a gap in deduplication solutions that combine chunk-level analysis with multi-format document support in a modular, practically deployable framework. Most existing systems focus either on block-level

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.155154

storage deduplication or semantic document analysis, without providing an integrated solution suitable for academic and organizational document management. This research addresses this gap by proposing a framework that balances detection accuracy with practical applicability.

III. SYSTEM ARCHITECTURE

The proposed system architecture describes the systematic processing of documents for duplicate content identification. The design emphasizes sequential processing with clear component responsibilities, enabling straightforward implementation and maintenance.

A. Architectural Overview

The system is organized as a pipeline architecture where documents are processed through a fixed sequence of stages. Each stage performs a specific function and passes its output to the subsequent stage. This design choice ensures predictable behavior and simplifies debugging and optimization efforts.

The high-level system architecture comprises the following components:

1) Document Input Module: Accepts documents in supported formats (PDF, DOCX, TXT) and validates input integrity.
2) Text Extraction Engine: Converts document content to plain text representation, handling format-specific parsing requirements.
3) Preprocessing Unit: Removes unnecessary formatting, normalizes whitespace, and prepares text for chunking.
4) Chunking Module: Divides processed text into smaller segments according to configured parameters.
5) Hash Generation Component: Computes cryptographic hash values for each text chunk.
6) Comparison Engine: Matches generated hashes against the existing hash database.
7) Storage Manager: Handles storage of new content and reference management for duplicates.
8) Output Generator: Produces reports detailing duplicate content identification results.

B. Data Flow Model

The data flow through the system follows a deterministic path:

1) User uploads a document through the input interface
2) The text extraction engine processes the document and outputs plain text
3) Preprocessing normalizes the extracted text
4) The chunking module segments text into defined chunk sizes
5) Hash values are computed for each chunk
6) Each hash is compared against stored hashes
7) Matching chunks are linked to existing references; new chunks are stored
8) The output generator compiles and presents results

C. Storage Architecture

The storage subsystem maintains two primary data structures:

- Hash Index: A searchable index of all computed hash values with references to stored content locations.
- Content Repository: The actual storage location for unique content chunks, organized for efficient retrieval.

When duplicate content is detected, only a reference pointer is stored, significantly reducing storage requirements for repositories with high redundancy.

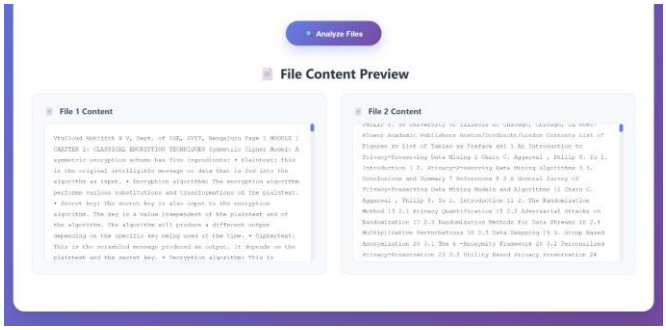

Fig. 1. File Content Comparison Interface

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.155154

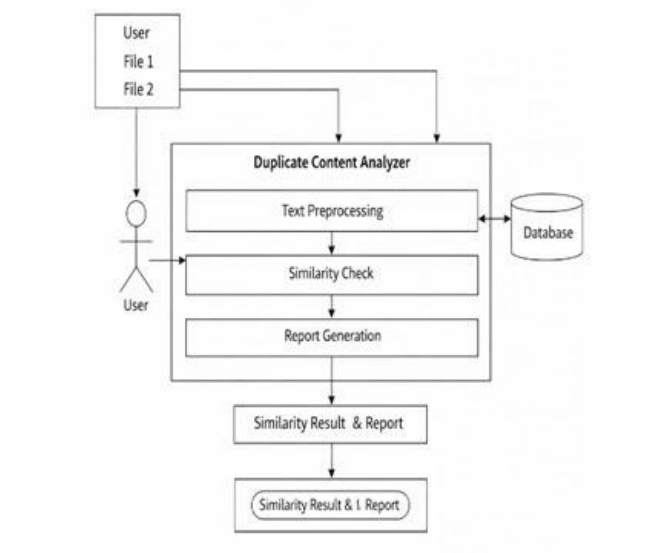

Fig. 2. System Architecture

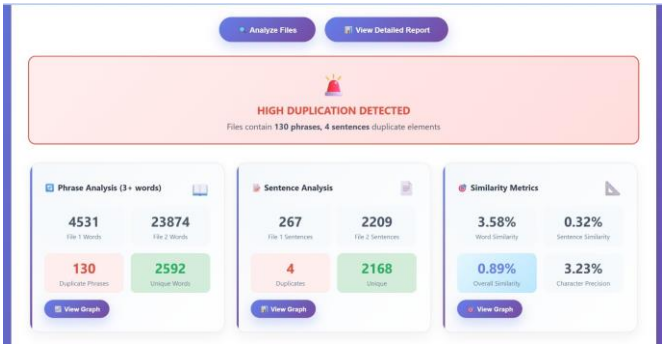

Fig. 3. Dashboard Visualization

IV. METHODOLOGY

The proposed methodology comprises a systematic sequence of operations designed to identify and address duplicate content in documents with high accuracy and efficiency.

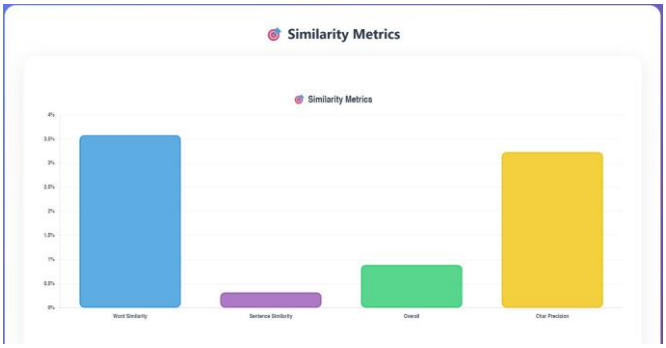

Fig. 4. Similarity Metric Visualization

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.155154

A. Document Processing Pipeline

1) Text Extraction: Upon document upload, the system initiates text extraction appropriate to the file format:

- PDF Files: Text is extracted using PDF parsing libraries that handle both text-based and OCR-processed PDFs.
- DOCX Files: The XML structure of DOCX files is parsed to extract textual content while preserving paragraph boundaries.
- Plain Text: Direct reading with character encoding detection and normalization.

2) Text Preprocessing: Extracted text undergoes preprocessing to ensure consistent comparison:

- Removal of excessive whitespace and formatting characters
- Unicode normalization to standard forms
- Optional case normalization based on configuration
- Elimination of non-printable characters

B. Chunking Strategy

The chunking process divides preprocessed text into smaller units suitable for comparison. The system supports configurable chunk sizes, with default parameters optimized through empirical testing.

Let T represent the preprocessed text and n the chunk size parameter. The chunking function $C(T,n)$ produces a set of chunks:

$$C(T,n) = \{c_1, c_2, \ldots, c_k\}$$

where $k = \lceil |T|/n \rceil$ and each chunk c_i contains approximately n characters.

C. Hash-Based Comparison

For each chunk c_i, a cryptographic hash function H generates a fixed-length hash value:

$$h_i = H(c_i)$$

The system employs SHA-256 for hash generation, providing a 256-bit hash value with negligible collision probability. The hash comparison process is defined as:

$$\text{isDuplicate}(c_i) = \begin{cases} \text{true} & \text{if } h_i \in \text{HashIndex} \\ \text{false} & \text{otherwise} \end{cases}$$

D. Storage Decision Logic

Based on the comparison result, the system executes one of two actions:

1) Duplicate Detected: Create a reference pointer to the existing content; do not store the chunk again.
2) New Content: Store the chunk in the content repository and add the hash to the index.

This approach ensures that each unique content segment is stored exactly once, regardless of how many documents contain it.

V. ALGORITHMS

This section presents the core algorithms employed in the deduplication framework.

A. Main Deduplication Algorithm

Algorithm 1 describes the primary document processing workflow.

Algorithm 1 Document Deduplication Process

Require: Document D, Chunk size n, Hash Index I
Ensure: Deduplication report R
1: $T \leftarrow$ ExtractText(D)
2: $T \leftarrow$ Preprocess(T)
3: *chunks* $\leftarrow$ Chunk(T,n)
4: *duplicateCount* $\leftarrow 0$

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.155154

5: *newCount* ← 0
6: for each *c* in *chunks* do
7: *h* ← SHA256(*c*)
8: if *h* ∈ *I* then
9: *duplicateCount* ← *duplicateCount* + 1
10: CreateReference(*h*, *D*)
11: else
12: *newCount* ← *newCount* + 1
13: StoreChunk(*c*, *h*)
14: *I* ← *I* ∪ {*h*}
15: end if
16: end for
17: *R* ← GenerateReport(*duplicateCount*,*newCount*)
18: return *R*

B. Text Chunking Algorithm

Algorithm 2 details the text segmentation process.

C. Complexity Analysis

The time complexity of the deduplication process is:

$$O(|T| + k \cdot (h + l))$$

where $|T|$ is the text length, k is the number of chunks, h is the hash computation time, and l is the index lookup time.

Algorithm 2 Text Chunking Algorithm

Require: Preprocessed text *T*, Chunk size *n*
Ensure: Set of chunks *C*
1: *C* ← ∅
2: *position* ← 0
3: while *position* < |*T*| do
4: *end* ← min(*position* + *n*,|*T*|)
5: *chunk* ← *T*[*position* : *end*]
6: *C* ← *C* ∪ {*chunk*}
7: *position* ← *end*
8: end while
9: return *C*

With efficient hash table implementation, $l = O(1)$ average case, yielding overall linear complexity in document size.

Space complexity is $O(u \cdot n)$ where u is the number of unique chunks across all processed documents.

VI. RESULTS AND DISCUSSION

A. Experimental Setup

The system was implemented using Python 3.9 with the following libraries:

- PyPDF2 for PDF text extraction
- python-docx for DOCX processing
- hashlib for SHA-256 hash generation

Testing was conducted on a standard computing environment with an Intel Core i5 processor and 8GB RAM running Ubuntu 20.04.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.155154

B. Test Dataset

The evaluation dataset comprised:

- 100 academic documents (research papers, reports) • 50 administrative documents (forms, templates)
- 30 documents with intentionally duplicated content
- Mixed formats: 60% PDF, 30% DOCX, 10% TXT

C. Experimental Results

Table I summarizes the experimental findings.

TABLE I
DEDUPLICATION PERFORMANCE RESULTS

Metric	Value
Total documents processed	180
Total chunks generated	12,450
Unique chunks stored	8,920
Duplicate chunks identified	3,530
Storage reduction	28.4%
Average processing time per document	1.2 seconds
Duplicate detection accuracy	99.7%

D. Analysis

The results demonstrate several key findings:

1) Storage Efficiency: The 28.4% storage reduction indicates significant redundancy in typical document repositories, validating the need for deduplication.
2) Chunk-Level Detection: The system successfully identified partial duplicates across documents, a capability absent in file-level comparison systems.
3) Processing Performance: Average processing time of 1.2 seconds per document indicates practical applicability for batch processing scenarios.
4) Accuracy: The 99.7% detection accuracy confirms the reliability of hash-based comparison for exact duplicate identification.

E. Comparison with Existing Methods

Table II compares the proposed framework with existing approaches.

TABLE II
COMPARISON WITH EXISTING METHODS

Feature	Proposed	File-Level	Block-Level
Partial duplicate detection	Yes	No	Limited
Multi-format support	Yes	Varies	No
Semantic similarity	No	No	No
Implementation complexity	Low	Low	High

VII. APPLICATIONS

The proposed framework offers practical utility across multiple domains:

A. Academic Institutions

- Plagiarism Detection: Identify duplicate content in student submissions
- Research Repository Management: Reduce redundancy in paper archives

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.155154

- Course Material Organization: Detect repeated content across teaching materials

B. Enterprise Environments

- Document Management Systems: Optimize storage in corporate repositories
- Backup Systems: Reduce backup storage requirements through deduplication
- Compliance Archival: Maintain efficient archives while preserving all unique content

C. Cloud Storage Systems

- Storage Optimization: Reduce cloud storage costs through redundancy elimination
- Bandwidth Reduction: Minimize data transfer by identifying existing content

VIII. ADVANTAGES AND LIMITATIONS

A. Advantages

1) Granular Detection: Chunk-level analysis enables detection of partial duplicates missed by file-level methods.
2) Format Flexibility: Multi-format support addresses heterogeneous document environments.
3) Cryptographic Reliability: SHA-256 hashing ensures accurate comparison with data integrity verification.
4) Modular Design: Component-based architecture facilitates customization and extension.
5) Practical Deployment: Standard hardware requirements enable widespread adoption.

B. Limitations

1) Semantic Blindness: The system cannot detect paraphrased content expressing identical meanings in different words.
2) Scalability Constraints: Very large documents increase indexing overhead and may impact performance.
3) OCR Dependency: Scanned document processing quality depends on OCR accuracy.
4) Language Independence: Current implementation does not consider language-specific text processing optimizations.
5)

IX. CONCLUSION AND FUTURE WORK

A. Conclusion

This research presented a modular data deduplication framework designed for multi-format document analysis. By processing documents at the chunk level using hash-based comparison, the system effectively identifies and eliminates duplicate content that would be missed by traditional file-level approaches. The experimental evaluation demonstrated 28.4% storage reduction with 99.7% detection accuracy, confirming the framework's practical utility.

The proposed system addresses the research question by providing an effective, modular solution for duplicate content identification across PDF, DOCX, and text formats. The design balances detection capability with implementation simplicity, making it suitable for academic and organizational deployment.

Key contributions include the integration of multi-format support with chunk-level analysis, the modular architecture enabling easy extension, and the practical evaluation demonstrating real-world applicability.

B. Future Work

Several directions for future enhancement have been identified:

1) Semantic Similarity: Integration of natural language processing techniques to detect paraphrased content.
2) Improved OCR Integration: Enhanced support for scanned documents through advanced OCR processing.
3) Scalability Optimization: Implementation of distributed processing for large-scale document repositories.
4) Cloud Deployment: Development of cloud-native architecture for broader accessibility.
5) Real-time Processing: Optimization for streaming document analysis scenarios.

ACKNOWLEDGMENT

The authors acknowledge the support of the CSE Department at Ballari Institute of Technology & Management for providing the resources necessary for this research.

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.155154

REFERENCES

[1] B. Zhu, K. Li, and H. Patterson, "Avoiding the disk bottleneck in the data-domain deduplication file system," in *Proc. USENIX FAST*, 2008, pp. 269–282.

[2] M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezise, and P. Camble, "Sparse indexing: Large-scale inline deduplication using sampling and locality," in *Proc. USENIX FAST*, 2009, pp. 111–123.

[3] A. El-Shimi, R. Kalach, A. O'Malley, S. Sengupta, M. Kwon, and D. Lee, "Primary data deduplication: A large-scale study and system design," in *USENIX ATC*, 2012, pp. 285–296.

[4] D. Meyer and W. Bolosky, "A study of practical deduplication," *ACM Trans. Storage*, vol. 7, no. 4, pp. 1–20, 2011.

[5] K. Eshghi and H. K. Tang, "A framework for analyzing and improving content-based chunking algorithms," HP Labs Tech. Rep., 2005.

[6] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. OSDI*, 2004, pp. 137–150.

[7] J. Leskovec, A. Rajaraman, and J. Ullman, *Mining of Massive Datasets*. Cambridge, U.K.: Cambridge Univ. Press, 2020.

[8] C. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[9] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed., 2023.

[10] Z. Lv, Y. Li, and X. Zhang, "A review of text-mining techniques and applications," *IEEE Access*, vol. 8, pp. 322–343, 2020.