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 11, November 2026

AI-Based Intelligent Data Deduplication for Cloud Storage

Thanish L.S[1], Deepak B.T[2], Dr.Durgadevi.P[3]

Department Of Computer Science and Engineering, Faculty Of Engineering And Technology,

SRM Institute Of Science And Technology, Vadapalni ,Chennai,Tamilnadu,India[1,2,3]

Abstract: Cloud storage systems manage amounts of data generated by users, applications and organizations. A significant amount of this data may consist of files or repeated versions of same content, leading to wasted storage space, increased workloads and higher costs for cloud storage.
Traditional deduplication methods typically rely on file names, metadata or cryptographic hashes such as SHA-256.
While these approaches can identify duplicates, they often fail to capture similar files that contain varying content.This project proposes an AI-Based Intelligent Data Deduplication system for cloud storage, which leverages both hash-based and intelligent similarity analysis techniques.
When a file is uploaded, the system generates a fingerprint using cryptography.Additionally, it applies algorithms to identify potentially redundant files.An AI-based decision layer classifies files as unique, exactly duplicated or highly similar.Users can interact with the system through a web interface to upload files, view their storage usage and identify duplicate data.The proposed solution aims to reduce storage utilization and improve storage efficiency while providing users with insights into cloud usage. A working prototype demonstrates that a combination of deduplication methods and sophisticated data analysis techniques are effective for cloud storage management.

Keywords: Cloud Storage, Data Deduplication, Artificial Intelligence Machine Learning, File Similarity, SHA-256 Storage Optimization, Cloud Computing.

1. INTRODUCTION

Cloud storage is increasingly used to store documents, images, videos, and other digital files. As the amount of stored data grows, duplicate files can take up extra storage space and add to storage costs.

By reducing the amount of repeated data, deduplication helps to minimize these costs. Traditional deduplication techniques often use cryptographic hash functions to identify identical files, but even slightly changed files will have different hashes.

For example, two reports with similar content may have different hash values if a few sentences were edited, and thus hash-based methods are not sufficient for detecting files with similar content.

This project proposes a hybrid system for deduplication, which combines the capabilities of exact duplicate detection with those of AI-based similarity analysis. The system is capable of identifying identical files and flagging potentially similar documents for closer inspection.

1.1 Objectives

The objectives of this project are to:
Identify exact duplicate files using SHA-256 hashing. Detect similar text documents using similarity analysis. Minimize unnecessary storage consumption.
Provide a web interface for file management. Display storage usage and estimated savings.
Measure accuracy of duplicate detection and processing time.

2. PROPOSED SYSTEM

The proposed system includes a web interface, a file-processing module, hash-based detection module, AI-based similarity module, and decision engine. When a user uploads a file, the system extracts its metadata and calculates its hash value. The hash is then compared to existing hashes to identify exact duplicates. If no hash match is found, similarity analysis can be used to identify related documents.

 This work is licensed under a Creative Commons Attribution 6.0 International License

Based on these results, the system categorizes the uploaded file as unique, an exact duplicate, or potentially similar to an existing file.

2.1 System Workflow

User Upload → File Processing → SHA-256 Hash Generation → Exact Duplicate Check → Similarity Analysis → Decision Engine → Storage or Reference Creation

2.2 File Classification

The system uses three classifications:
Unique File: No exact duplicate or sufficiently similar document is identified. Exact Duplicate: The file has the same cryptographic hash as an existing file. The system can reuse the stored content and create a reference, subject to access permissions.
Similar File: The file has different content but shows a high degree of similarity to an existing document. The system flags it for review rather than automatically deleting or replacing it.

3. SYSTEM ARCHITECTURE

The architecture is divided into five layers.

3.1 User Interface Layer

The web interface allows users to:
Upload and search for files.

View stored files and their details. Check duplicate and similar files. Monitor storage usage.
Delete files and manage their versions. View estimated storage savings.

3.2 File Processing Layer

When a file is uploaded, the system extracts the information required for comparison. The extracted information includes:
File name and type. File size.
Upload timestamp. Cryptographic hash.
Extracted text or other supported content features.

This information is passed to the duplicate detection and similarity analysis modules.

3.3 Hash-Based Deduplication

The hash-based module uses SHA-256 to generate a fingerprint for each file. The process is as follows:
Receive the uploaded file. Calculate its SHA-256 hash.
Compare the hash with existing database records.

If a match exists, identify the file as an exact duplicate. Reuse the existing stored content when appropriate.
If no match exists, store the new file and its metadata.

3.4 AI-Based Similarity Detection

Hashing cannot identify files that have been modified slightly. To address this limitation, the proposed system uses text similarity analysis.

For text documents, the system extracts the text and converts it into a numerical representation. A suitable similarity model can then compare the representations and calculate a similarity score.

For example:

Document A: Cloud computing provides storage. Document B: Cloud platforms provide data storage.

Although the wording differs, the documents express similar ideas. A semantic similarity model may identify this relationship even though the documents have different hashes.

The similarity score ranges from 0 to 1, where a higher value indicates greater similarity. The following thresholds can be

used for initial testing:
Similarity score Classification 0.00–0.49 Low similarity
0.50–0.79 Moderately similar
0.80–1.00 Highly similar

These are proposed thresholds for experimentation, not validated performance results. They must be adjusted based on testing with a representative dataset.

3.5 Deduplication Decision Engine

The decision engine combines the results of hash comparison and similarity analysis. Case 1: Exact Duplicate
If the uploaded file matches an existing hash, the system reuses the stored content and creates a reference where appropriate.

Case 2: Unique File

If no exact match or significant similarity is detected, the system stores the file as a new item. Case 3: Similar File
If the hash does not match but the similarity score is high, the system marks the file for review. The user can compare the documents and decide whether to retain both versions.

This approach prevents similar but meaningfully different documents from being removed automatically.

4. DATABASE DESIGN

The database maintains file metadata, hash values, storage locations, and similarity results.
Field Description

file_id Unique file identifier user_id Owner of the file
file_name Name of the uploaded file file_type File extension or type file_size Size of the file
file_hash SHA-256 hash value storage_path Location of the stored content
similarity_score Calculated similarity score, when available duplicate_status Unique, duplicate, or similar
created_at Upload timestamp

The database acts as an index for locating existing files and recording the results of duplicate detection.

5. IMPLEMENTATION TECHNOLOGIES

The proposed prototype can be developed using the following technologies.

5.1 Frontend

HTML and CSS JavaScript React.js Tailwind CSS

5.2 Backend

Python FastAPI

5.3 Database

PostgreSQL or Supabase

5.4 File Storage

Local file storage can be used during prototype development. Cloud object storage can be integrated for deployment.

5.5 AI Similarity Module

A Python-based text similarity model or embedding model can be used to compare document content.

5.6 Security

The implementation should include user authentication, access control, file validation, and secure handling of file metadata. Duplicate detection must not expose one user's private files to another user

6. EXPERIMENTAL METHODOLOGY

The system should be tested using a dataset containing different categories of files: Exact copies of the same file.

Renamed copies of existing files.

Documents with minor text modifications.

Documents with similar meanings but different wording. Completely unrelated documents.
Multiple versions of the same document.

The files should be uploaded through the application, and the system's decisions should be compared with the known classifications in the test dataset.

6.1 Evaluation Metrics

The following metrics can be used to evaluate the system:

Total files: Number of files submitted for testing.

Exact duplicates detected: Number of identical files correctly identified. Similar files detected: Number of relevant similar documents identified. Detection accuracy: Proportion of correct classifications.
Processing time: Time required to process and classify a file.

Storage saved: Difference between the original storage requirement and the actual storage used.

6.2 Storage Saving Calculation

Storage savings can be calculated using:

Storage Saved = Original Storage − Actual Storage Used The percentage reduction is:
[
\text{Storage Reduction (%)} =
\frac{\text{Original Storage}-\text{Actual Storage Used}}
{\text{Original Storage}}\times100
]

Actual storage savings and detection accuracy must be calculated from the files used in the experiments. No experimental results are claimed in this paper until the prototype has been implemented and tested.

7. DISCUSSION

The proposed hybrid approach addresses two different problems: identifying identical files and finding documents with similar content.

Hash-based detection can identify exact duplicates efficiently. The similarity module provides additional information about documents that have been edited or rewritten. Using hash comparison before similarity analysis can also reduce unnecessary computational work.

However, semantic similarity does not prove that two documents are interchangeable. A small change in a legal document, academic report, or technical specification may be important. Therefore, similar files should be reviewed before any decision is made to remove or replace them.

The actual effectiveness of the system will depend on the selected similarity model, dataset, file types, and processing resources.

8. ADVANTAGES

The proposed system offers the following potential advantages:
Reduces storage used by exact duplicate files. Identifies potentially similar documents.
Provides a simple interface for file management. Displays storage usage and estimated savings.
Supports file-version comparison and review. Can be extended to cloud storage platforms.

9. LIMITATIONS

9.1 Processing Overhead

Similarity analysis requires additional computation compared with simple hash matching.

9.2 File-Type Dependency

Text similarity models work best with supported text documents. Images, videos, and other binary files require different analysis techniques.

9.3 False Similarity

Documents with similar wording may contain different information. Incorrect classification can occur if similarity thresholds are not properly selected.

9.4 Data Security

The system must protect file contents and metadata. Deduplication across users must be designed carefully to prevent unauthorized access or information leakage.

9.5 Evaluation Requirements

The system's storage savings, detection accuracy, and performance cannot be established without implementing the prototype and conducting experiments.

10. FUTURE SCOPE

Future development can include:

Integration with Amazon S3, Google Cloud Storage, or Azure Blob Storage. Improved semantic similarity models.
Image and video similarity detection. File-version tracking and recovery.
Distributed deduplication across storage servers. Privacy-preserving deduplication techniques.
Storage usage prediction.
Automated recommendations for reviewing redundant files.

These extensions could improve the system's usefulness for larger cloud storage environments.

11. CONCLUSION

This paper presents a proposed intelligent data deduplication system for cloud storage using SHA-256 hashing and AI-based similarity analysis. Hash comparison identifies exact duplicate files, while similarity analysis highlights documents with related content. A web interface provides file management and storage monitoring features.

The system is designed to reduce redundant storage while preserving distinct file versions. Its effectiveness must be established through prototype implementation and experimental evaluation. With appropriate security controls and reliable similarity analysis, the approach could provide a practical method for improving cloud storage management.

REFERENCES

1. J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer, “Reclaiming Space from Duplicate Files in a Serverless Distributed File System,” Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS), 2002.

2. NIST, Secure Hash Standard (SHS), Federal Information Processing Standards Publication 180-4, 2015.

3. A. Muthitacharoen, B. Chen, and D. Mazieres, “A Low-Bandwidth Network File System,” Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP), 2001.

4. M. O. Rabin, Fingerprinting by Random Polynomials, Center for Research in Computing Technology, Harvard University, Technical Report TR-15-81, 1981.

5. L. Breiman, Random forests, Machine Learning, vol. 45, no. 1, pp. 5-32, 2001

6. T. Chen and C. Guestrin, XGBoost: A scalable tree boosting system, Proc. ACM SIGKDD, pp. 785-794, 2016.