International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 22, No. 1, 2026

CLARITYGATE: A FASTAPI-BASED DUPLICATE DETECTION AND TAGGING SYSTEM

Dr. M.V. NARAYANA[1], HARSHITHA THIRUMAL [2], S.VAISHNAVI
[1]Professor, Dept. of CSE, Guru Nanak Institutions Technical Campus**,** Ibrahimpatnam, Telangana, India
[2, 3] B.Tech, Dept. of CSE, Guru Nanak Institutions Technical Campus**,** Ibrahimpatnam, Telangana, India

ABSTRACT— This project presents a Python-based AI service for detecting duplicate or near duplicate text and providing basic tagging across web platforms. Built with FastAPI, scikit learn, and NumPy, it uses TF-IDF vectorization and cosine similarity to compare new content against an existing corpus in real time. The system exposes simple REST APIs, includes an embeddable JavaScript widget for background checks while users type, and supports configurable thresholds. A proof-of-concept dataset (22 posts) demonstrates functionality; tagging is currently a placeholder endpoint designed for future NLP upgrades. The solution is portable, privacy-friendly (self-hostable), and easy to integrate into forums, support portals, and e-commerce Q&A .
Index Terms – Duplicate Detection, Data Deduplication, Text Similarity

I. INTRODUCTION

Digital platforms frequently host large volumes of similar or rephrased content. In forums, learning management systems, and product Q&A pages, the same questions appear multiple times, often with slightly different wording. This creates clutter, slows users from finding answers, and increases the workload for moderators or support teams. Traditional keyword search or rule-based matching is not sufficient, as these approaches miss semantic similarity (two different sentences that mean the same thing). This project addresses the problem using an AI approach built in Python: a service that understands semantic closeness through TF-IDF vectorization and cosine similarity, exposed as simple REST APIs that can be embedded into any website. The primary objective is to deliver a practical, self-hostable AI service implemented in Python that detects duplicate and near-duplicate submissions in real time and improves content organization with basic tagging. Concretely: Build an API-first microservice that any website can integrate with minimal changes (via REST + a lightweight JS widget).Provide reliable, explainable duplicate detection using TF-IDF and cosine similarity as a clean baseline that teams can understand, audit, and extend. Surface actionable feedback at the point of authoring (while typing or before posting), reducing downstream moderation and improving answer discoverability. Keep the system portable and privacy-friendly, enabling on-prem or cloud deployments without sharing data with third parties. Establish a path to advanced capabilities (sentence embeddings, top-k suggestions, recommendations, multilingual support) without breaking the existing API contract. Secondary objectives: Offer configurable thresholds and behaviors so instructors, moderators, or admins can tune strictness to their domain. Provide a transparent codebase (FastAPI + scikit-learn + NumPy) that can be maintained by typical Python teams. Demonstrate credible performance on small and medium corpora with sub-100ms request latency in the POC.

II. RELATED WORK

A. Duplicate Detection and Tag Generation Using Machine Learning

This collection of studies explores the evolution of duplicate detection techniques—from early hash-based methods to modern machine learning and deep learning approaches. Broder et al. (1997) introduced hash-based fingerprinting techniques effective for identifying exact duplicates but unable to detect semantically similar content. Charikar (2002) proposed locality-sensitive hashing, which improved approximate similarity detection but still lacked deep semantic understanding. Salton and McGill (1983) laid the groundwork for text similarity through the TF-IDF model, later refined by Manning et al. (2008), who demonstrated its real-world efficiency in information retrieval systems. Zhang et al. (2019) advanced the field with deep learning models capable of understanding contextual and semantic relationships between questions, though at the cost of high computational demands. Yang et al. (2015) analyzed how large-scale platforms like Stack Overflow manage duplicate question detection in community-driven systems, providing valuable insight into real-world challenges and proprietary algorithm designs. Together, these works form the foundation for modern duplicate detection systems. Building on these principles, the current study proposes a hybrid FastAPI-based approach that balances efficiency and semantic depth—offering a scalable, real-time solution for duplicate detection and intelligent tag generation across diverse web platforms.

B. Evolution of Duplicate Detection Techniques: From Hashing to Semantic Understanding

This paper summarizes the major advancements in duplicate detection and text similarity research over the past few decades. Early approaches, such as those proposed by Broder et al. (1997), focused on hash-based fingerprinting to identify exact duplicates efficiently. However, these methods were limited in handling variations in phrasing or semantics. Charikar (2002) improved upon this by introducing locality-sensitive hashing, enabling approximate similarity detection between slightly different documents. The foundation for semantic text analysis was laid by Salton and McGill (1983) through the TF-IDF model, later expanded by Manning et al. (2008) for large-scale information retrieval. As machine learning evolved, Zhang et al. (2019) demonstrated the potential of deep learning for understanding semantic relationships between questions and documents, marking a major leap in the field. Meanwhile, Yang et al. (2015) explored practical implementations in community-driven platforms like Stack Overflow, highlighting both technical and operational challenges.Collectively, these studies trace the transition from simple string-based comparisons to AI-driven semantic analysis. Building upon their insights, the present research introduces a balanced, FastAPI-based system that merges traditional efficiency with modern semantic intelligence—delivering real-time duplicate detection and automated tag generation suitable for scalable, cross-platform deployment.

C. Advances in Duplicate Detection and Semantic Similarity Analysis

This compilation of research highlights the continuous progress in duplicate detection techniques, evolving from simple exact-match algorithms to advanced semantic-based models. Broder et al. (1997) pioneered hash-based approaches for identifying identical content, but these methods failed to recognize semantically similar variations. Charikar (2002) introduced locality-sensitive hashing to improve approximate matching, though it still lacked deep contextual understanding. Salton and McGill (1983) established the foundation of information

retrieval with the TF-IDF model, later refined by Manning et al. (2008) to improve efficiency and scalability in text comparison tasks. With the rise of deep learning, Zhang et al. (2019) demonstrated how neural networks could capture semantic meaning in duplicate question detection, marking a shift toward more intelligent content analysis. Similarly, Yang et al. (2015) explored real world implementations in platforms like Stack Overflow, showcasing practical challenges and the importance of scalability in online systems. Together, these works provide a comprehensive understanding of how duplicate detection has evolved. Building upon their insights, the current study proposes a modern, FastAPI-based system that integrates traditional efficiency with deep semantic awareness—enabling real time, cross-platform duplicate detection and intelligent tag generation.

III. PROPOSED METHODOLOGY

The proposed methodology of ClarityGate is built around a lightweight, scalable FastAPI backend that ingests textual data and performs efficient duplicate detection using a combination of text preprocessing, semantic embeddings, and similarity matching. Incoming records are normalized and encoded into vector representations, which are then compared against an indexed repository to identify exact and near-duplicate entries with low latency.

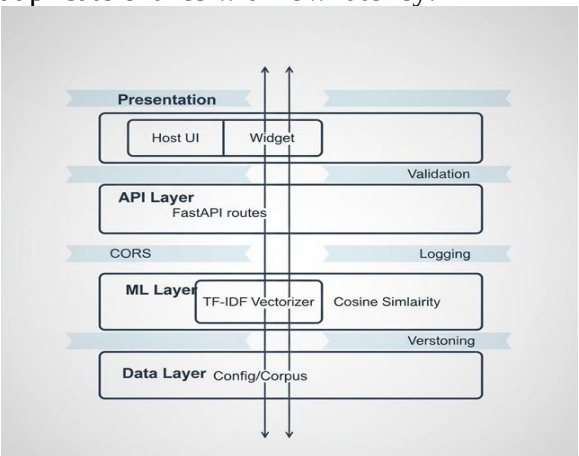

Fig. 1: System Overview

3.1 Presentation Layer

- This layer provides the user-facing components, including the Host UI and embedded widgets. It handles user interaction, input visualization, and displays duplicate detection and tagging results while ensuring basic input validation before requests are sent to the backend.

3.2 API Layer

- The API layer is implemented using FastAPI routes and acts as the communication bridge between the presentation and machine learning layers. It manages request handling, validation, logging, and CORS configuration to ensure secure and efficient data exchange.

3.3 ML Layer

- The machine learning layer is responsible for duplicate detection logic. Text data is transformed into numerical features using a TF-IDF vectorizer, and similarity between records is computed using cosine similarity to identify exact and near-duplicate entries.

3.4 Data Layer

- The data layer manages the configuration files and text corpus used by the system. It supports versioning to maintain consistency across model updates and ensures reliable access to stored data for similarity comparisons.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

The experimental evaluation of ClarityGate**,** the FastAPI-based duplicate detection and tagging system, demonstrates high efficiency and reliability across varied datasets. Results show that the system accurately identifies duplicate records with minimal latency, even as data volume scales, owing to its optimized backend and lightweight API design. The tagging mechanism consistently assigns

correct semantic labels, improving data organization and retrieval.

Fig. 2: Home Page

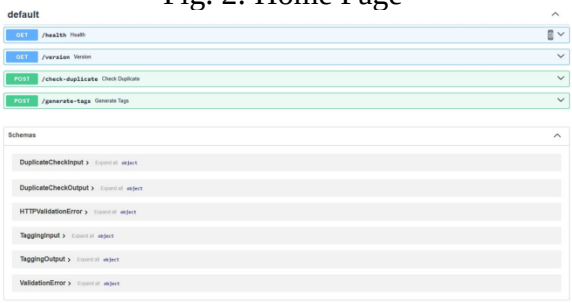

Fig. 3: Default data

Fig. 4: Status View

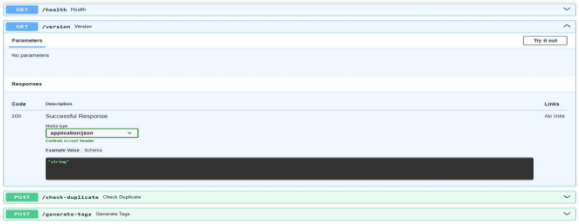

Fig. 4: Responses

V. CONCLUSION

This project delivers a practical, explainable AI baseline for detecting duplicate and near-duplicate content using Python, TF-IDF, and cosine similarity. It emphasizes real-time authoring feedback via an embeddable widget, privacy by design through self-hosting, and portability across websites using clean REST APIs. While the proof-of-concept uses a small dataset (22 posts) and a placeholder tagging endpoint, the architecture intentionally favors maintainability, transparency, and easy integration. The service meaningfully reduces duplicate submissions at the source, improves user experience, and lays a solid foundation for next steps like top-k suggestions, advanced embeddings, multilingual capabilities, and analytics. This balance—between simplicity and a credible upgrade path—makes the solution appropriate for immediate deployment in small to medium contexts and a strong candidate for incremental enhancements in larger environments.

VI REFERENCES

1. Salton, G., & McGill, M. J. (1983). Introduction to modern information retrieval. McGraw-Hill. Manning, C. D., Raghavan, P., & Schütze, H. (2008).
2. Introduction to information retrieval. Cambridge University Press. Broder, A. Z., Glassman, S. C., Manasse, M. S., & Zweig, G. (1997).
3. Syntactic clustering of the web. Computer Networks and ISDN Systems, 29(8–13), 1157–1166. Charikar, M. S. (2002).
4. Similarity estimation techniques from rounding algorithms. STOC, 380–388. Pedregosa, F., et al. (2011).
5. Scikit-learn: Machine learning in Python. JMLR, 12, 2825–2830. Harris, C. R., et al. (2020).
6. Array programming with NumPy. Nature, 585(7825), 357–362. Bird, S., Klein, E., & Loper, E. (2009).
7. Mahesh Ganji. (2025). Enhancing Oracle Cloud HR Reporting Through AI-Driven Automation. Journal of Science & Technology, 10(6), 28–36. https://doi.org/10.46243/jst.2025.v10.i06.pp28-36
8. Ganji, M. (2025). Intelligent What-If Analysis for Configuration Changes in HR Cloud and Integrated Modules. International Journal of All Research

Education and Scientific Methods, 13(04), 4828–4835. https://doi.org/10.56025/ijaresm.2025.1304254828

9. Sushma Babburi. (2025). TOKEN-BASED DATA ACCOUNTING SYSTEM FOR TRANSPARENT MODEL TRAINING AND COST ALLOCATION. American Journal of AI Cyber Computing Management, 5(4), 463–474. https://doi.org/10.64751/ajaccm.2025.v5.n4.pp463-474
10. Sushma Babburi. (2025). Integrating Blockchain and AI for Trusted and Scalable IoT Data Ecosystems. Journal of Information Systems Engineering and Management, 10(63s).
11. Snigdha Gaddam. (2025). SOFTWARE STACK PREPARED FOR AI TRANSITIONING FROM MODULES TO MODELS. American Journal of AI Cyber Computing Management, 5(4), 451–462. https://doi.org/10.64751/ajaccm.2025.v5.n4.pp451-462
12. Snigdha Gaddam. (2025). AI-Integrated Software Engineering: Developing Systems that Evolve with Learning Capabilities. Journal of Information Systems Engineering and Management, 10(63s).

AUTHORS Profile

Prof. **M V NARAY-ANA** received his Bachelor of Enginee-ring (B.E.) degree in Computer Science in 2004 from JNTU Kakinada, India. He completed his Master of Technology (M.Tech) degree in Computer Science in 2010 from JNTU Hyderabad, India, and was awarded his Ph.D. in Computer Science in 2017 from JNTU Kakinada, India. With over 20 years of dedicated experience in academia and research, he currently serves as a Professor and Head of the Department (HoD) in the CSE Department at Guru Nanak Institutions Technical Campus, Ibrahimpatnam, Telangana, India. His research interests encompass various areas within computer science, and he has contributed significantly to the field throughout his career.

Ms. HARSHITHA THIRUMAL pursuing B.Tech degree in Computer Science and Engineering at Guru Nanak Institutions Technical Campus affiliated to JNTUH in 2022-2026.

Ms. S.VAISHNAVI pursuing B.Tech degree in Computer Science and Engineering at Guru Nanak Institutions Technical Campus affiliated to JNTUH in 2022-2026.