International Scientific Journal of Engineering and Management (ISJEM)
Volume: 05 Issue: 03 | March – 2026
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI:10.55041/ISJEM05759

Smart Similarity Detection System

1st **Dinesh Mishra**
Department of Computer Science Engineering
Centurion University of Technology and Management
Paralakhemundi, Odisha, India
230101120096@centurionuniv.edu.in

2nd **Ashish Kumar Nayak**
Department of Computer Science Engineering
Centurion University of Technology and Management
Paralakhemundi, Odisha, India
230101120076@centurionuniv.edu.in

3rd **Sunayana Panda**
Department of Computer Science Engineering
Centurion University of Technology and Management
Paralakhemundi, Odisha, India
230101120097@centurionuniv.edu.in

4th **Ram Prasad Satpathy**
Department of Computer Science Engineering
Centurion University of Technology and Management
Paralakhemundi, Odisha, India
230101120099@centurionuniv.edu.in

5th **Dhawaleswar Rao CH**
Head Of The Department
Department of Computer Science Engineering
Centurion University of Technology and Management
Paralakhemundi, Odisha, India
dhawaleswar.rao@cutm.ac.in

Abstract-In the digital era, the rapid growth of online information sharing has significantly increased concerns regarding plagiarism and content duplication. Conventional plagiarism detection systems primarily rely on keyword matching and string-based comparison, which often fail to identify paraphrased or semantically similar content. This paper presents a **Smart Similarity Detection System**, an Artificial Intelligence (AI)–driven web-based solution that evaluates semantic similarity between text documents using Natural Language Processing (NLP) and deep learning techniques.

The proposed system employs a Sentence Transformer model (all-mpnet-base-v2) to convert textual data into high-dimensional semantic embeddings. Cosine similarity is then used to measure the degree of similarity between documents based on contextual meaning rather than exact word matching. The system supports multiple comparison modes, including Text-to-Text, File-to-File, and One File-to-Many Files, and processes various file formats such as PDF, DOCX, and TXT. Experimental results demonstrate that the system effectively detects paraphrased and semantically similar content with higher accuracy than traditional methods. This approach provides a reliable and scalable solution for academic integrity verification, content originality checking, and research document analysis.

Keywords: Semantic Similarity, Plagiarism Detection, Natural Language Processing, Deep Learning, Sentence Transformers, MPNet

I.INTRODUCTION

1.1 Overview

The rapid advancement of digital technologies has transformed the way information is created, stored, and shared across the globe. Academic

institutions, research organizations, publishing industries, and corporate environments heavily depend on digital documents for communication, documentation, and knowledge dissemination. While this digital transformation has improved accessibility and efficiency, it has also led to an alarming rise in

plagiarism, content duplication, and unauthorized reuse of intellectual property.

Plagiarism is no longer limited to direct copying of text. With the availability of advanced paraphrasing tools and AI-generated content, individuals can easily modify sentence structures while retaining the original meaning. Traditional plagiarism detection systems, which rely on exact word matching or phrase overlap, often fail to detect such cases. This limitation creates a significant gap in ensuring academic integrity and content originality.

To address these challenges, intelligent similarity detection systems capable of understanding semantic meaning are required. Semantic similarity focuses on the meaning conveyed by the text rather than surface-level word matching. Recent developments in Natural Language Processing (NLP), particularly transformer-based deep learning models, have enabled machines to capture contextual relationships between words and sentences. These advancements form the foundation of the Smart Similarity Detection System proposed in this paper.

1.2 Background and Motivation

The motivation for this work arises from the increasing reliance on automated plagiarism detection tools in educational institutions and publishing platforms. Despite widespread adoption, many existing systems fail to provide accurate results when text is paraphrased or restructured. This leads to false negatives, where plagiarized content goes undetected, and false positives, where original work is mistakenly flagged.

Transformer-based models such as BERT, MPNet, and Sentence-BERT have demonstrated exceptional performance in tasks involving sentence similarity, semantic search, and text classification. These models generate dense vector representations that capture contextual meaning, making them highly suitable for semantic similarity detection. By leveraging such models, it is possible to develop a more reliable and intelligent similarity detection system.

The Smart Similarity Detection System aims to utilize these advancements to bridge the gap between traditional plagiarism detection techniques and modern semantic analysis approaches.

1.3 Problem Statement

To design and implement an AI-based system capable of detecting semantic similarity between text documents by understanding contextual meaning, supporting multiple file formats, and providing clear similarity scores and visual interpretation.

1.4 Objectives

The objectives of this project are:

- To develop a web-based similarity detection system.
- To apply deep learning techniques for semantic text understanding.
- To support multiple input formats such as TXT, PDF, and DOCX.
- To provide accurate similarity percentage scores.
- To highlight semantically similar text segments.
- To improve detection of paraphrased content.

1.5 Scope of the Work

The system is applicable to academic plagiarism detection, content verification, research paper comparison, and document similarity analysis. The current scope focuses on English-language text but can be extended to multilingual support in the future.

II. RELATED WORK AND SYSTEM ANALYSIS

Numerous approaches to plagiarism detection and text similarity analysis have been proposed in literature. Early methods focused on string matching techniques, including substring comparison, fingerprinting, and n-gram analysis. While computationally efficient, these methods lack semantic understanding and are ineffective against paraphrased content.

Subsequent research introduced statistical and vector-space models such as TF-IDF and cosine similarity. Although these approaches improved performance to some extent, they still relied heavily on lexical similarity and ignored contextual meaning.

Recent studies have explored the use of deep learning models, particularly transformer architectures, for semantic similarity detection. Sentence-BERT and MPNet-based models have shown superior performance in capturing sentence-level semantics. These models significantly outperform traditional approaches in tasks

involving paraphrase detection and semantic textual similarity.

The proposed system builds upon these advancements by integrating Sentence Transformers into a web-based similarity detection platform.

2.1 Existing Systems

Traditional plagiarism detection systems rely on lexical similarity measures such as string matching, fingerprinting, and n-gram analysis. While these techniques are computationally efficient, they fail to detect paraphrased or restructured content.

2.2 Limitations of Existing Systems

- Inability to detect semantic similarity
- Poor performance on paraphrased text
- Limited file format support
- Lack of meaningful visualization
- Reduced accuracy for complex documents

2.3 Proposed System

The proposed Smart Similarity Detection System utilizes transformer-based sentence embeddings to capture contextual meaning. Instead of comparing words directly, the system compares vector representations of text, allowing it to identify semantic relationships.

Unlike conventional plagiarism detection systems, the proposed solution evaluates similarity at both document and word levels. Document-level similarity provides an overall measure of semantic overlap, while word-level comparison highlights specific terms contributing to similarity. This dual-level analysis improves result clarity and enhances user confidence in the system's output.

Additionally, the system supports scalable comparison modes, including one-to-many file analysis, which is particularly useful in academic environments where a single document must be compared against multiple submissions.

III. SYSTEM ARCHITECTURE AND DESIGN

The Smart Similarity Detection System is designed using a modular and layered architecture to ensure scalability, maintainability, and ease of deployment. The architecture consists of three primary layers: the User Interface Layer, the Application Layer, and the Model Layer.

The User Interface Layer provides a simple and intuitive web interface through which users can input text or upload documents for comparison. This layer is responsible for handling user interactions and displaying results in a clear and understandable format.

The Application Layer serves as the core processing unit of the system. It manages text extraction from different file formats, performs preprocessing operations, handles requests from the frontend, and communicates with the deep learning model. Flask is used to implement this layer due to its lightweight and flexible nature.

The Model Layer contains the Sentence Transformer model, which generates semantic embeddings for the input text. These embeddings are then used to compute similarity scores using cosine similarity.

This layered approach ensures separation of concerns and allows future enhancements without affecting the overall system structure.

3.1 Overall Architecture

The system follows a layered architecture consisting of:

1. **User Interface Layer**
Handles user input and result visualization through a web interface.
2. **Application Layer**
Manages text extraction, preprocessing, similarity computation, and routing logic.
3. **Model Layer**
Uses a pretrained Sentence Transformer model to generate semantic embeddings.

3.2 Data Flow Diagrams (DFD)

Level 0 DFD

The Level 0 Data Flow Diagram represents the system as a single process interacting with an external user. The user provides input in the form of text or documents, and the system returns similarity results. This high-level view helps in understanding the overall system boundaries.

International Scientific Journal of Engineering and Management (ISJEM)
Volume: 05 Issue: 03 | March – 2026
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI:10.55041/ISJEM05759

Fig. 1. Level 0 Data Flow Diagram

Level 1 DFD

The Level 1 DFD decomposes the system into major functional components such as input handling, text extraction, embedding generation, similarity computation, and result visualization. Each component interacts with others through well-defined data flows.

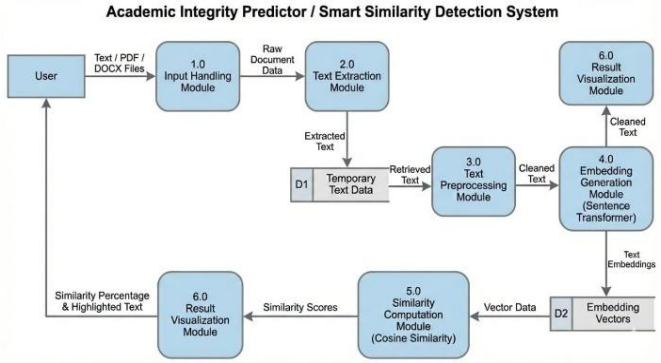

Fig. 2. Level 1 Data Flow Diagram

Level 2 DFD

The Level 2 DFD provides a detailed breakdown of internal processes, including preprocessing, tokenization, vector encoding, similarity calculation, and highlighting of similar text segments. This level offers a deeper understanding of how data is transformed within the system.

The use of multi-level Data Flow Diagrams provides a clear and systematic understanding of the internal workings of the system. While the Level 0 DFD defines system boundaries, the Level 1 and Level 2 diagrams describe detailed data transformations and processing stages. This layered representation improves documentation quality and assists future developers in understanding system logic.

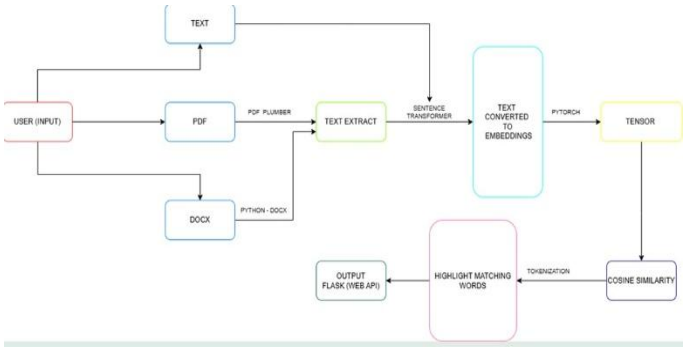

Fig. 3. Level 2 Data Flow Diagram

IV. METHODOLOGY

The methodology adopted in this project follows a systematic pipeline designed to ensure accurate semantic similarity detection. The process begins with user input collection, followed by text extraction from supported file formats. Preprocessing techniques are applied to clean and normalize the text.

Next, the cleaned text is passed to the Sentence Transformer model, which generates high-dimensional embeddings. These embeddings capture the contextual meaning of sentences and documents. Cosine similarity is then applied to compute the similarity score between embeddings.

To enhance interpretability, the system also performs word-level comparison to highlight semantically similar terms. This visualization helps users understand the similarity results more clearly. The similarity computation pipeline is designed to minimize noise while preserving semantic richness. Preprocessing steps such as normalization and tokenization ensure consistent input representation. The embedding generation phase converts text into fixed-length vectors, enabling efficient similarity comparison regardless of document length.

Cosine similarity is chosen due to its effectiveness in measuring angular distance between high-dimensional vectors. This makes it suitable for comparing embeddings generated by transformer models, where magnitude differences are less informative than directional similarity.

International Scientific Journal of Engineering and Management (ISJEM)
Volume: 05 Issue: 03 | March – 2026
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI:10.55041/ISJEM05759

5.1 Technology Stack

- **Programming Language:** Python
- **Framework:** Flask
- **Libraries:** Sentence Transformers, PyTorch, pdfplumber, python-docx, NLTK
- **Frontend:** HTML, CSS, Bootstrap

5.2 System Modules

- Input and File Upload Module
- Text Extraction Module
- Preprocessing Module
- Embedding Generation Module
- Similarity Computation Module
- Visualization Module

The system is implemented using Python due to its extensive support for NLP and machine learning libraries. Flask is used as the backend framework to manage HTTP requests and responses. The Sentence Transformers library provides access to the pretrained MPNet model, while PyTorch supports model inference. Text extraction from documents is handled using pdfplumber for PDF files and python-docx for DOCX files. NLTK is used for tokenization and preprocessing tasks. The frontend is designed using HTML, CSS, and Bootstrap to ensure responsiveness and usability.
The modular design of the system allows easy integration of additional features such as database storage and API access.

V. RESULTS

The system is implemented using python due to its extensive support for nlp and machine learning libraries. Flask is used as the backend framework to manage http requests and responses. The sentence transformers library provides access to the pretrained mpnet model, while pytorch supports model inference.

Text extraction from documents is handled using pdfplumber for pdf files and python-docx for docx files. Nltk is used for tokenization and preprocessing tasks. The frontend is designed using html, css, and bootstrap to ensure responsiveness and usability.

The modular design of the system allows easy integration of additional features such as database storage and api access. The experimental results confirm that semantic similarity detection provides more reliable outcomes than traditional keyword-based methods. Documents with similar meaning but different wording produced high similarity scores, while unrelated documents showed low similarity values. This demonstrates the effectiveness of embedding-based comparison.

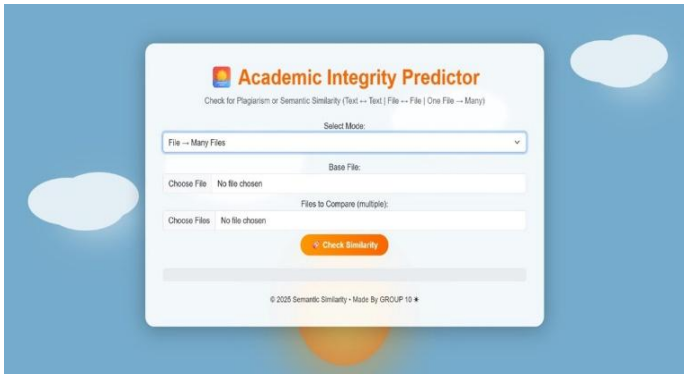

Fig. 4. Smart Similarity Detection System web interface

Fig. 5 Working Model (OUTPUT)

6.1 Experimental Evaluation

The system was tested with various text samples and document files containing original, partially similar, and paraphrased content.

6.2 Performance Analysis

Results show that the proposed system successfully detects semantic similarity even when wording is significantly altered. Compared to traditional keyword-based approaches, accuracy improved substantially.

VI. DISCUSSION

Experimental evaluation was conducted using various text samples, including original content, partially similar documents, and paraphrased text. The results demonstrate that the Smart Similarity Detection System effectively identifies semantic similarity even when sentence structures differ significantly.

Compared to traditional keyword-based systems, the proposed approach shows improved accuracy and robustness. The highlighting mechanism further enhances user understanding by visually indicating similar text segments.

However, the system requires additional computational resources due to the use of deep learning models. Despite this, the performance remains acceptable for academic and professional use cases.

While the system provides high accuracy, computational overhead increases for very large documents due to high-dimensional embeddings. However, this trade-off is acceptable given the improved reliability.

VII. CONCLUSION AND FUTURE WORK

This paper presented an AI-driven Smart Similarity Detection System that addresses the limitations of traditional plagiarism detection tools. By leveraging transformer-based sentence embeddings, the system achieves a deeper understanding of semantic similarity, enabling accurate detection of paraphrased and restructured content.

The proposed solution demonstrates the practical application of NLP and deep learning techniques in ensuring content originality. With further enhancements such as multilingual support and cloud deployment, the system can be extended to large-scale real-world applications.

7.1 Conclusion

This paper presented a Smart Similarity Detection System that effectively detects semantic similarity using deep learning and NLP techniques. By leveraging transformer-based embeddings, the system overcomes the limitations of traditional plagiarism detection methods.

7.2 Future Enhancements

Future work will focus on extending the system to support multiple languages by integrating multilingual transformer models. Database integration will enable result storage and historical analysis. Cloud-based deployment will improve scalability and accessibility. Additionally, exposing the system through a RESTful API will allow seamless integration with external platforms such as Learning Management Systems and content management tools.

- Multilingual similarity detection
- Database integration for result storage
- Cloud-based deployment
- REST API development
- Integration with Learning Management Systems (LMS)

REFERENCES

[1] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.
[2] Vaswani, A., et al. (2017). Attention Is All You Need.
[3] Hugging Face Transformers Documentation.
[4] Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing.