SVS Satish, 2025, 13:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

Intellicheck – Plagiarism Detector Using HMM, BERT and Abstract Syntax Tree

SVS Satish, S D Anirudh, P Chandu, K Suhaas Varma, Assistant Professor P.Jyothi
Department of Computer Science and Engineering, Maharaj Vijayaram Gajapathi Raj College of Engineering (Autonomous), Andhra Pradesh, India

Abstract- In the modern era of widespread access to information and digital resources, plagiarism detection has become increasingly essential for preserving academic integrity and protecting intellectual property. This research focuses on the design and development of a dual-functional plagiarism detection system capable of analyzing both text and source code submissions. The text-based detection module employs a Hidden Markov Model (HMM) to evaluate textual similarity and utilizes a BERT model to understand and compare the semantic meaning of the content. The source code module leverages Abstract Syntax Trees (AST) to identify structural similarities in programming code, offering precise detection of plagiarized logic and patterns. The proposed system bridges the gap between surface-level content analysis and deeper contextual and structural understanding, making it a robust and comprehensive tool for a wide array of applications. This paper also addresses implementation challenges, evaluates system performance metrics such as precision and recall, and suggests potential future enhancements, including the incorporation of multilingual support and the expansion to more programming languages. By integrating advanced technologies, this system provides a reliable solution for detecting plagiarism in academic, research, and software development environments, ensuring fairness and originality.

Keywords- Plagiarism Detection, Hidden Markov Model (HMM), BERT Model, paraphrase-MiniLM-L6- v2, Abstract Syntax Tree (AST), Text Similarity, Semantic Analysis, Source Code Analysis, Tree-Edit Distance, Machine Learning, Preprocessing Techniques, Accuracy, Precision, Recall, F1-Score, Text Preprocessing, Code Normalization, Tokenization, Stopword Removal, Stemming, Data Parsing, Structural Similarity, Probabilistic Models, Logical Flow Analysis, Cross-Validation, Evaluation Metrics, User Interface, Plagiarism Report, Academic Integrity, Software Development, Structural Patterns.

I. INTRODUCTION

In today's digital world, the ease of accessing and sharing information has significantly increased instances of plagiarism. From academia to the software industry, plagiarism undermines the value of intellectual property, affecting credibility, fairness, and originality. Text-based plagiarism, which involves the replication of written content, compromises the authenticity of academic work and scientific research. Meanwhile, source code plagiarism is a growing concern in software development, as it undermines the innovative efforts of programmers by copying logic, structure, and design.

The limitations of existing plagiarism detection systems further exacerbate the problem. Current solutions are often narrowly focused, with text-based tools relying on superficial keyword matching, and source code checkers limited to basic pattern recognition. These tools struggle to account for deeper contextual or structural similarities, leading to inaccurate or incomplete results. As a result, there is an urgent need for a more holistic plagiarism detection system that can efficiently and effectively detect similarities across both textual and programming domains.

This paper presents a plagiarism detection system designed to address these challenges. The system offers two options for users: one for detecting plagiarism in text-based content and another for analyzing source code. The text- based plagiarism checker integrates advanced models such as the Hidden Markov Model (HMM) for identifying textual similarity and BERT for analyzing semantic meaning. On the other hand, the source code plagiarism checker utilizes Abstract Syntax Trees (AST) to identify structural and logical similarities in programming code. By combining these methodologies, the proposed system provides a robust and comprehensive solution to tackle modern plagiarism challenges.

1. Existing System

Current plagiarism detection systems have several significant shortcomings. Text- based plagiarism tools primarily rely on basic methods such as keyword matching or string similarity, which can only identify exact matches or superficial patterns. This approach often leads to a high rate of false positives and fails to detect deeper contextual or semantic similarities in the text. Furthermore, these systems are unable to process large datasets effectively or analyze multilingual content, limiting their scope of application.

Similarly, source code plagiarism detection tools are largely limited to simple pattern- matching techniques, which overlook structural and logical similarities. For instance, minor alterations in variable names or function definitions are sufficient to bypass detection in existing systems. Most tools fail to analyze the underlying logic or syntactic structure of programming code, leading to unreliable results. Additionally, there is a lack of integrated platforms that can handle both text and source code plagiarism, requiring users to rely on multiple, fragmented solutions.

These limitations underscore the need for a more advanced and holistic approach to plagiarism detection. A system capable of addressing both textual and structural similarities while reducing false positives and improving accuracy can significantly enhance plagiarism detection efforts across various domains.

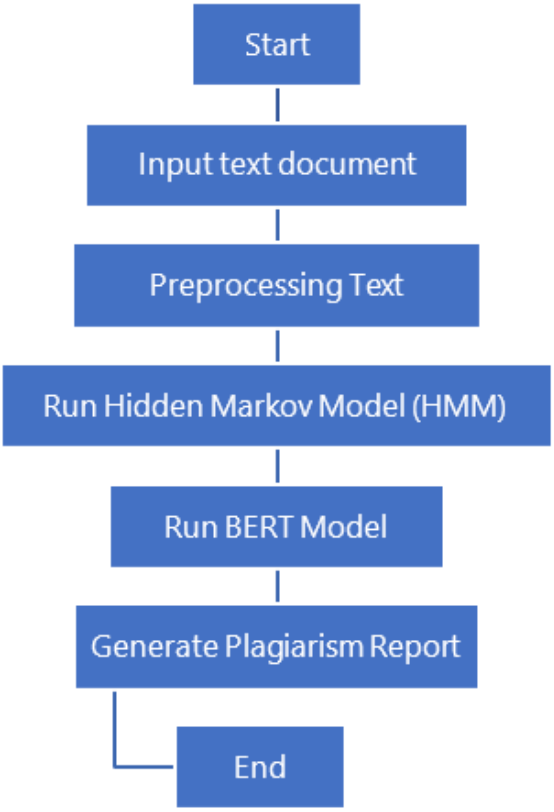
II. PROPOSED SYSTEM

To overcome the deficiencies of existing systems, this paper proposes a unified plagiarism detection system designed to handle both text-based and source code submissions. The system is equipped with advanced technologies to deliver accurate, efficient, and comprehensive plagiarism detection.

1. Text-Based Plagiarism Checker

Hidden Markov Model (HMM): This model evaluates textual similarity by analyzing the sequence of words and their probabilistic relationships. HMM ensures that patterns and repetitions within the text are effectively identified.

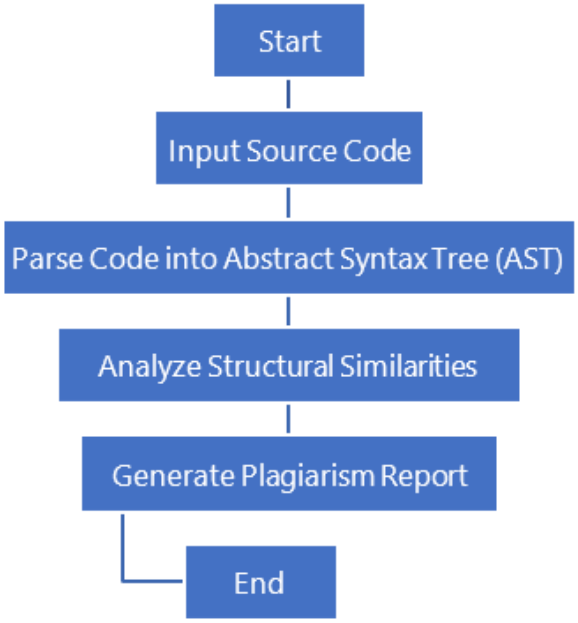
BERT Model: BERT performs semantic analysis, allowing the system to comprehend and compare the deeper meaning of textual content. This ensures that context and intent are accounted for, reducing false positives and improving accuracy.



2. Source Code Plagiarism Checker

Abstract Syntax Trees (AST): AST parses the source code into a tree structure, enabling the system to analyze the syntactic and structural components of the code. This method ensures the detection of copied logic and patterns, even when superficial changes such as variable renaming or code formatting have been made.

The proposed system offers a seamless and user-friendly experience for detecting plagiarism in both text and source code. By combining these methodologies, the system delivers a reliable and accurate tool that meets the needs of academic institutions, researchers, and software developers. The integration of these techniques bridges the gap between surface-level analysis and deeper contextual and structural understanding, making it a valuable contribution to the field of plagiarism detection.



III. METHODOLOGY

The methodology for the proposed plagiarism detection system focuses on two distinct areas: text-based content analysis and source code analysis. Each module employs advanced

computational techniques to ensure accurate and efficient plagiarism detection.

1. Text-Based Plagiarism Checker

The text-based module integrates two key models to achieve high accuracy in detecting similarities:

Hidden Markov Model (HMM)

HMM is used to evaluate textual similarity by analyzing the sequences of words and their probabilistic relationships. This model effectively identifies repetitive patterns and subtle similarities between documents.

Formula: $P(O|\lambda) = \text{sum}(P(O|Q, \lambda) * P(Q|\lambda))$

Where:

- $P(O|\lambda)$ is the probability of the observed sequence O given the model λ.
- Q is a possible sequence of states.
- λ represents the model parameters, including transition, emission, and initial probabilities.

BERT Model

BERT, specifically the paraphrase-MiniLM-L6-v2 variant, is used to analyze and compare the semantic meaning of text. This model excels at understanding context and intent, making it ideal for detecting paraphrased content.

Formula: $\text{Cosine Similarity} = (A \cdot B) / (||A|| * ||B||)$

Where:

- A and B are the vector embeddings of the two texts.
- (A . B) is the dot product of the vectors.
- ||A|| and ||B|| are the magnitudes of the vectors.

By combining HMM and the BERT paraphrase-MiniLM-L6-v2 model, the text-based checker is capable of identifying both surface-level and contextual similarities, significantly improving accuracy and reducing false positives.

2. Source Code Plagiarism Checker

The source code analysis module employs Abstract Syntax Trees (AST) to detect structural and logical similarities in programming code. AST parses source code into a hierarchical tree structure, where each node represents a syntactic construct.

AST allows the system to analyze the logic and structure of code, detecting plagiarism even when the code is obfuscated (e.g., through variable renaming or formatting changes).

Formula: Tree-Edit Distance = sum(Cost of Inserts, Deletes, and Replacements)

The AST-based analysis ensures that the system identifies plagiarized logic and patterns at a deeper level, making it a robust solution for software developers and educators.

3. Preprocessing Steps

Both modules involve preprocessing stages to standardize input data and enhance performance:

4. Text Preprocessing

- **Tokenization:** Splitting text into words or sentences.
- **Stopword Removal:** Filtering out common words (e.g., "the," "is") that do not add significant meaning.
- **Stemming/Lemmatization:** Reducing words to their base forms (e.g., "running" becomes "run").

5. Code Preprocessing

Normalization: Removing comments, extra whitespaces, and formatting inconsistencies.
Parsing: Converting source code into AST for structural analysis.

IV. IMPLEMENTATION

The implementation of the proposed plagiarism detection system involves a series of steps designed to ensure effective and accurate plagiarism detection for both text-based content and source code. This section outlines the key stages of the implementation process.

1. Training the Models
Text-Based Plagiarism Checker
HMM and BERT Models: The Hidden Markov Model (HMM) and the BERT (paraphrase-MiniLM-L6-v2) model are trained on datasets containing examples of both plagiarized and original text. The training data includes a diverse range of documents to cover various writing styles and contexts. The HMM focuses on identifying textual patterns, while BERT analyzes semantic meaning.

Source Code Plagiarism Checker:
AST Parsing: Source code files are parsed into Abstract Syntax Trees (ASTs). The training dataset consists of multiple programming languages and coding styles. The AST-based model learns to detect structural and logical similarities in code, even when superficial changes (such as variable renaming or formatting) are made.

2. Evaluating Performance
The system's performance is evaluated using several metrics to ensure its accuracy and reliability:
Accuracy: Accuracy measures the proportion of correctly identified plagiarism cases.

Formula: Accuracy = (True Positives + True Negatives) / Total Samples

Precision: Precision indicates the proportion of true plagiarism cases out of all detected cases. Formula: Precision = True Positives / (True Positives + False Positives)

Recall: Recall reflects the system's ability to identify all actual plagiarism cases. Formula: Recall = True Positives / (True Positives + False Negatives)

F1-Score combines precision and recall into a single metric. Formula: F1-Score = 2 * (Precision * Recall) / (Precision + Recall)

Cross-validation techniques, such as K-fold cross-validation, are employed to ensure the models' robustness and prevent overfitting. These evaluation methods help in assessing the system's overall performance and identifying areas for improvement.

3. User Interface
The system includes a user-friendly interface that allows users to upload text documents and source code files for plagiarism detection. The interface is designed to be intuitive and accessible to a wide range of users, including academic institutions, researchers, and software developers.

4. Features
- **Text Upload Feature:** Users can upload text documents in various formats (e.g., DOCX, PDF, TXT) for similarity analysis.
- **SOURCE CODE SUBMISSION PLATFORM:** USERS CAN upload source code files in different programming languages for structural analysis.

- **Plagiarism Report:** The system generates a detailed plagiarism report that highlights matched sections and provides an overall
- **Similarity Score.** The report includes insights on both textual and structural similarities, helping users understand the extent of plagiarism.

V. GRAPHICAL USER INTERFACE(GUI) RESULTS

The GUI is developed using the MERN stack with integrated flask framework. The GUI consists of a Home Page, Check Plagiarism page, Login Page, History Page, Feedback Modal and requesting a demo modal

Welcome or Home Page
Figure 1 and 2 shows the welcome page of the Intellicheck

Check Plagiarism Page
Figure 3 and 4 shows the check plagiarism page i.e., the interface where you need to submit the text to check plagiarism

Login Page
Figure 5 and 6 shows the Login and History page

History Page
Figure 7 shows the History Page where the documents are not accessible but sessions are available

Feedback or Contact Modal
Figure 8 shows the contact modal where users can be able to provide feedback about the App
Demo Modal
Figure 9 shows the Demo modal where user can request a demo and will be notified once the demo is scheduled by admins

Result Page
The Result Page displays the plagiarism analysis results after submitting text or source code. Users can upload their content, extract text, and check for plagiarism.

Figure 10 Shows the process of uploading a document and extracting text.

Figure 11 Displays the interface for submitting text and checking plagiarism.

Figure 12 Demonstrates the submission of source code for plagiarism detection.

This page provides a detailed plagiarism report, including semantic similarity, structural similarity, and overall similarity scores through graphical representations such as bar charts and circular progress indicators.

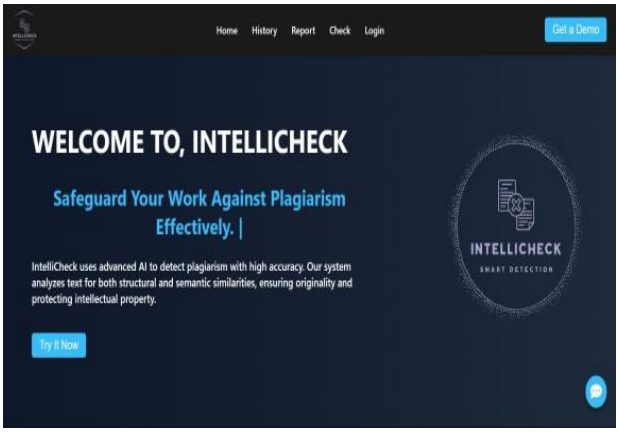


Figure 1. Welcome page – 1

Figure 2. Welcome page – 2

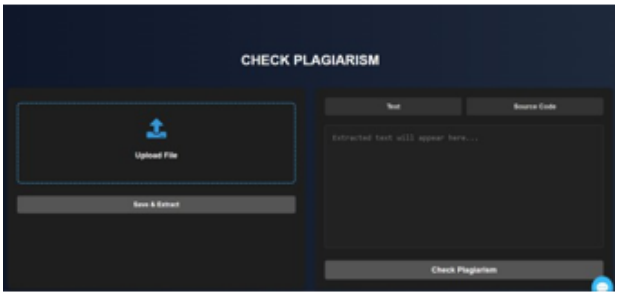


Figure 3. Check Plagiarism page – 1

Figure 4. Check Plagiarism page – 2

Figure 5. Login Page

Figure 6. Register page

Figure 7. History Page

Figure 8. Feedback or Contact Modal

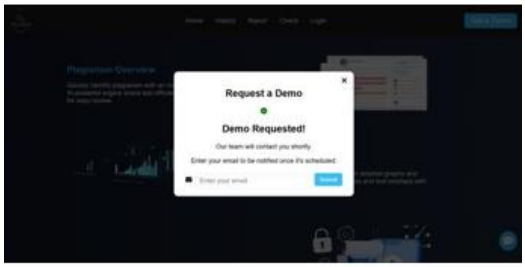


Figure 9. Request a demo Modal

Figure 10. Uploading text/Interface

Figure 11. Plagiarism result

Figure 12. Source Code

VI. CONCLUSIONS

The Intellicheck system uses sophisticated techniques to accurately identify cases of plagiarism. By applying Hidden Markov Models (HMMs) to assess structural similarities and utilizing natural language processing tools, the system compares input text against a predefined database. During the development phase, various HMMs were tested, such as Multinomial HMM for analyzing token distributions and Structural HMM for identifying sequence patterns. However, these approaches did not achieve the desired level of accuracy.

After thorough testing, the team selected the Gaussian HMM as the primary model because of its superior ability to detect complex patterns in text. This model effectively identifies both traditional and AI-generated instances of plagiarism. By focusing on the Gaussian HMM, Intellicheck ensures reliable and precise plagiarism detection, making it an excellent tool for analyzing text and identifying similarities.

REFERENCES

1. Meuschke, N, Gipp, B. (2019). Academic Plagiarism Detection: A Systematic Literature Review. ACM Computing Surveys, 52(6). https://doi.org/10.1145/3345317
2. Dixit, A., Unnathi, P. N., Guttedar, R. J., & More, S. (2024). Advancements in Plagiarism Detection: A Comprehensive Review. Journal of Emerging Technologies https://www.jetir.org/papers/JETIR2404050.pdf
3. Meuschke, N., & Gipp, B. (2013). State- of-the-art in detecting academic plagiarism. International Journal for Educational Integrity, 9(1),50-71. https://doi.org/10.21913/IJEI.v9i1.899
4. Maurer, H. A., Kappe, F., & Zaka, B. (2006). Plagiarism—a survey. Journal of Universal Computer Science, 12(8), 1050-1084. https://doi.org/10.3217/jucs-012-08-1050
5. Potthast, M., Stein, B., & Anderka, M. (2008). A Wikipedia-based multilingual retrieval model

for plagiarism detecti. In European Conference on Information Retrieval https://doi.org/10.1007/978-3-540-78646-7_81

6. Alzahrani, S. M., Salim, N., & Abraham,vA. (2012). Understanding plagiarism linguistic patterns, textual features, and detection methods. IEEE Transactions on Systems, Man, and https://doi.org/10.1109/TSMCC.2011.2134847
7. Mor, B., Garhwal, S., & Kumar, A. (2021). A Systematic Review of Hidden Markov Models and Their Applications. Archives of Computational Methods in Engineering, 28, https://doi.org/10.1007/s11831-020-09422-4
8. Gupta, A., & Dhingra, B. (2012). Stock Market Prediction Using Hidden Markov Models. IEEE Conference Proceedings. https://users.cs.duke.edu/~bdhingra/papers/stock_hmm.pdf
9. Gudala, C., Padi, T. R., Rekha, S., & Dar, Q. F. (2022). Stochastic Modeling for the Analysis and Forecasting of Stock Market Trend using Hidden Markov Model. Asian Journal of Probability and Statistics, 18(1), 43-56.
10. Resmi, N. G., & Soman, K. P. (2014). Abstract Syntax Tree Generation Using Modified Grammar for Source Code Plagiarism Detection. International Journal of Computer Applications in Technology, https://ijcat.org/articles/1-6/Abstract-Syntax- Tree-Generation-using-Modified-Grammar- for-Source-Code-Plagiarism-Detection.html