BINARY MULTILINGUAL MACHINE GENERATED TEXT DETECTION

Jaya chandran[1], G Jahnavi[2], U Yashaswini[3], J Meghana[4], K Kowshik Narayana [5]

[1] *Professor Dept. Of SOCSE,* [2,3,4,5] *Dept. Of SOCSE*
[1,2,3,4,5] *Presidency University, bangalore-560064*

Abstract

With the rapid advancement of natural language generation technologies, distinguishing machine-generated text from human written content has become increasingly challenging yet essential. This project aims to develop a robust, multilingual system capable of accurately identifying machine-generated text across languages, including English, Indonesian, German, and Russian. Utilizing a substantial dataset of 674,083 training samples and 288,894 development samples characterized by attributes such as source, sub-source, language, generation model, label, and text we explore the efficacy of various machine learning and deep learning algorithms.

To achieve reliable classification, the system integrates Random Forest, Long Short-Term Memory (LSTM) networks, Bidirectional Encoder Representations from Transformers (BERT), Decision Tree, and Logistic Regression models. Each model is rigorously evaluated on its ability to handle multilingual data, focusing on both accuracy and computational efficiency. This project combines traditional machine learning with cutting-edge deep learning techniques, contributing a valuable tool for digital content verification by enabling precise differentiation between human-authored and machine-generated text. The proposed system supports a wide range of applications in content validation and enhances trust in digital information across multiple languages and contexts.

Keywords: Multilingual text analysis, Random Forest, Long Short-Term Memory (LSTM) networks, Bidirectional Encoder Representations from Transformers (BERT), Decision Tree, Logistic Regression and Natural Language Processing (NLP)

I INTRODUCTION

1.1 Motivation

The surge in machine-generated text has raised concerns over content authenticity, driving the need for effective detection systems. Accurately distinguishing human-written from machine-generated content is crucial for maintaining trust in digital communication. This project addresses this need by developing a multilingual, high-accuracy detection system. Leveraging advanced algorithms ensures reliable performance across diverse languages. This solution supports content verification in various fields, from journalism to security, bolstering confidence in information integrity.

1.2 Problem Statement

In the digital age, the ability to distinguish between machine generated and human-written text has become increasingly important due to the proliferation of automated content creation tools and the need for content authenticity. The Binary Multilingual Machine-Generated Text Detection project addresses this challenge by developing a system that can accurately classify text as either machine-generated or human-authored, across multiple languages.

Given a diverse dataset containing textual features such as id, source, sub source, lang, model, label, and text, the problem is to implement and evaluate various machine learning and deep learning algorithms to achieve high accuracy in text classification. The core issue is to create a predictive model that can robustly handle multilingual inputs and effectively identify the origin of the text. This involves the application of advanced algorithms, including Random Forest, Long Short-Term Memory (LSTM) networks, Bidirectional Encoder Representations from Transformers (BERT), Decision Tree, and Logistic Regression, to solve the problem of detecting and verifying the authenticity of textual content in a rapidly evolving digital landscape.

1.3 Objective of the project

The objective of the Binary Multilingual Machine-Generated Text Detection project is to develop a highly accurate system for classifying text as either machine-generated or human-written. By leveraging a diverse dataset and implementing a range of algorithms, including Random Forest, LSTM, BERT, Decision Tree, and Logistic Regression, the project aims to achieve robust

performance in detecting text origins. This system will handle multilingual inputs effectively, providing reliable verification of content authenticity in a rapidly evolving digital landscape. Additionally, the project seeks to address the growing challenge of automated content generation by ensuring that the model can adapt to various languages and textual nuances, thereby enhancing the reliability of digital content verification across global platforms.

1.4 Scope of the project

The scope of the Binary Multilingual Machine-Generated Text Detection project includes the development of a versatile text classification system that operates across multiple languages. The project will involve preprocessing a diverse dataset with features such as id, source, sub_source, lang, model, label, and text. It will encompass the implementation and evaluation of various machine learning and deep learning algorithms, including Random Forest, LSTM, BERT, Decision Tree, and Logistic Regression. The system will be designed to accurately differentiate between machine-generated and human-written text, addressing challenges in content authenticity and verification within a global digital context.

1.5 Project Introduction

The widespread use of advanced language models has made it increasingly difficult to distinguish between human-written and machine-generated text, posing challenges to content authenticity and trust. This project focuses on building a multilingual text classification system to detect machine-generated text across languages such as English, Indonesian, German, and Russian. Using a large dataset of over 960,000 samples, we analyze key features like source, language, and model type to inform classification. Our approach combines traditional machine learning methods, including Random Forest, Decision Tree, and Logistic Regression, with deep learning techniques like LSTM and BERT to enhance detection accuracy. Each model's performance is evaluated to determine its effectiveness in handling multilingual data. By providing a reliable tool for text authenticity verification, this project aims to strengthen information security and support content integrity across digital platforms.

II LITERATURE REVIEW

2.1 Related Work1

T. R. Han & J. H. Kim, "A Survey on Text Classification: From Shallow to Deep Learning," 2020.

Explanation: This paper provides a comprehensive review of text classification techniques, tracing the evolution from traditional methods to modern deep learning approaches. It outlines the strengths and limitations of various methods, offering insights into future research directions. From Shallow to Deep Learning" (2020), offers a detailed overview of the progression of text classification methods. It begins by examining traditional shallow learning techniques such as bag-ofwords, TF-IDF, and machine learning algorithms like SVM and Naive Bayes. The paper then transitions to modern deep learning approaches, including neural networks, CNNs, RNNs, and transformers, which have revolutionized the field. It highlights the advantages of deep learning, such as better handling of large-scale data and context, while also discussing their challenges, such as the need for large labeled datasets and computational power. Finally, the paper points out key areas for future research, including improving model interpretability and handling domain-specific tasks. **Y. Wu, X. Wang & X. Xu, "Multilingual Text Classification with Transformer Models," 2022.**

Explanation*:* This study explores the application of transformer models for multilingual text classification. It demonstrates how transformers enhance performance across various languages by leveraging context and semantic information, setting new benchmarks in the field. The study highlights how transformers, such as BERT and its variants, improve multilingual text classification by effectively capturing contextual and semantic information across diverse languages. It demonstrates that transformers can outperform traditional methods by understanding language nuances and relationships, even with limited resources for some languages. The paper sets new benchmarks for multilingual text classification, showing the model's ability to generalize well across different languages and domains

Al-Hadhrami, M. I. Idris, & A. Al-Kahtani, "A Review of Text Classification Algorithms for Detecting Fake News," 2020.
Explanation*:* This review paper examines various text classification algorithms specifically designed for fake news detection. It discusses the effectiveness of different approaches and provides recommendations for improving classification accuracy in the

context of misinformation. The study evaluates the performance of different algorithms, including traditional machine learning methods like SVM, Naive Bayes, and decision trees, as well as advanced deep learning models such as LSTM and BERT. The paper discusses the strengths and weaknesses of these methods in the context of fake news detection, highlighting challenges like feature selection, dataset imbalance, and context understanding. It also offers recommendations for improving accuracy, such as incorporating multi-modal data and leveraging more sophisticated natural language processing techniques.

J. Li, Z. Liu, & H. Wang, "Cross-lingual Text Classification Using Multi-View Learning," 2022.Explanation*:* The paper presents a novel multi-view learning framework for cross-lingual text classification. It highlights how combining multiple perspectives can improve classification performance across languages, particularly in handling diverse linguistic features. The approach combines multiple perspectives or "views," such as different linguistic features (e.g., syntactic, semantic) or representations (e.g., word embeddings, sentence embeddings), to improve classification performance across various languages. The paper demonstrates that by integrating these diverse views, the model can better capture the nuances of different languages, leading to improved accuracy and robustness in handling cross-lingual tasks.

A. Conneau, V. R. Lample, & L. Denoyer, "Cross-Lingual Language Model Pretraining," 2020. Explanation*:* This research highlights the complexities of training cross-lingual models and demonstrates how models like XLM (Cross-lingual Language Model) perform in multilingual classification tasks. It discusses challenges in capturing linguistic and cultural nuances, emphasizing that multilingual detection models must adapt to varied syntax and grammar to achieve accuracy across languages .The research demonstrates how XLM can be pre-trained on multiple languages to effectively transfer knowledge across languages, overcoming barriers in syntax, grammar, and cultural nuances. It emphasizes the importance of adapting models to diverse linguistic structures to maintain accuracy in cross-lingual tasks. The paper also discusses the difficulties in ensuring that multilingual detection models can generalize well across languages with varying linguistic properties.

III SYSTEM ANALYSIS

3.1 Existing System

Existing systems for detecting machine-generated text primarily use a mix of traditional machine learning and modern deep learning models. Traditional methods like Random Forest, Decision Tree, and Logistic Regression offer efficiency but struggle with complex multilingual data due to limited feature extraction capabilities. LSTM networks improved on this by capturing sequential patterns, though they face scalability issues in multilingual tasks. Transformer-based models, especially BERT, excel in handling multilingual nuances and context but come with high computational demands. Hybrid systems have emerged, combining shallow machine learning with deep learning embeddings to balance accuracy and efficiency. These systems enable better multilingual support but still face challenges in low-resource languages and computational efficiency

3.2 Disadvantages of existing systems

- Logistic Regression is simple and interpretable, providing quick insights into feature impacts but struggles with nonlinear relationships and can underperform in complex datasets.
- Support Vector Machines are effective in high-dimensional spaces and suitable for complex data distributions, yet they can be computationally intensive and sensitive to the choice of kernel and hyperparameters.
- Random Forest enhances robustness through ensemble learning, reducing overfitting and improving accuracy; however, it can be less interpretable than individual decision trees and may perform poorly with very large datasets.
- Decision Trees are intuitive and easy to visualize, making decision-making processes clear, but they are prone to overfitting, especially with deep trees, and can be unstable with small changes in data.
- The existing system benefits from the combination of these algorithms for adapting to varying data characteristics, but the limitations of each algorithm must be managed to ensure reliable intrusion detection outcomes.

3.3 Proposed System

In the proposed system, we utilize a combination of advanced algorithms to enhance intrusion detection. The stacking classifier integrates Gradient Boosting**,** Random Forest**,** and Logistic Regression**,** leveraging their strengths to improve predictive accuracy. Gradient Boosting builds strong models by optimizing weak learners, while Random Forest provides robustness through ensemble learning. Naive Bayes is included for its effectiveness in handling large datasets and detecting anomalies. Additionally, AdaBoost enhances the performance of weak classifiers by focusing on misclassified instances. This combination aims to create a resilient and accurate intrusion detection system capable of adapting to diverse cyber threats.

3.4. Advantages of Proposed System

Limited Multilingual Support in Traditional Models: Machine learning models like Random Forest and Decision Tree struggle with multilingual text due to their reliance on handcrafted features, which don't generalize well across languages.

Scalability Issues with LSTM: While LSTM networks can capture long-term dependencies, they are computationally expensive and scale poorly in multilingual applications, especially with large datasets.

High Computational Demand of Transformers: Transformer-based models like BERT provide excellent accuracy but require significant computational resources, making them challenging to deploy for real-time detection in resource-limited environments.

Insufficient Performance in Low-Resource Languages: Most existing systems perform well in high-resource languages but struggle in low-resource languages due to limited training data and fewer pretrained models.

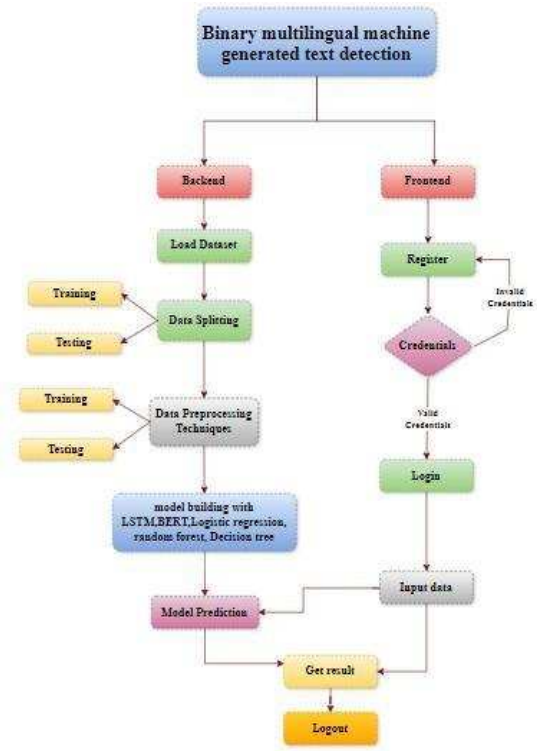
Complexity of Hybrid Approaches: Although hybrid systems combining traditional and deep learning models improve performance, they add complexity to the implementation and increase computational requirements.

Generalization Challenges: Existing models often struggle to generalize well across different types of text (e.g., social media vs. news) and content domains, reducing their versatility.

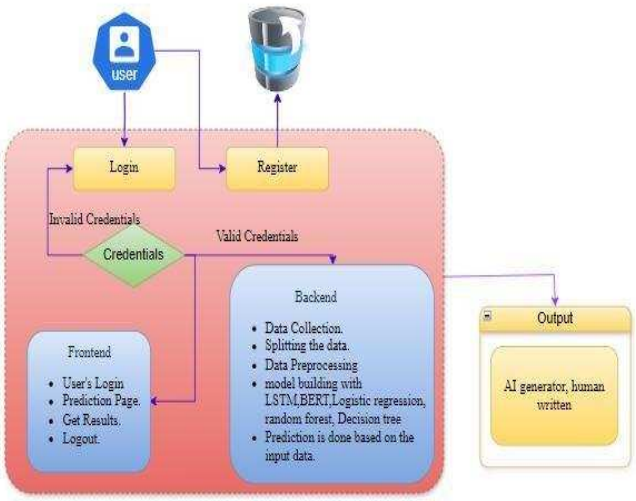
Inconsistent Accuracy across Languages: Many models achieve varying levels of accuracy depending on the language due to differences in linguistic structure, limiting their effectiveness for comprehensive multilingual detection

.

3.5 Project flow



3.6 Architecture Diagram



IV METHODOLOGY

1) 4.1. Long Short-Term Memory (LSTM)

- **Definition**: LSTM is a type of recurrent neural network (RNN) designed to overcome the limitations of standard RNNs in capturing long-term dependencies. It uses memory cells with gates to control the flow of information, allowing it to retain important information over longer sequences.
- **Model Training**: LSTM models are trained by feeding sequential data through the network, using backpropagation through time (BPTT) to adjust weights. The model minimizes a loss function (often cross-entropy for classification) using an optimizer like Adam, iteratively updating weights based on the gradient of the loss with respect to the parameters.

2) 4.2. Bidirectional Encoder Representations from Transformers (BERT)

- **Definition**: BERT is a transformer-based language model that captures bidirectional context by considering both left and right context in each layer. It is pre-trained on large amounts of text data through unsupervised tasks like masked language modelling and next sentence prediction.
- **Model Training**: For fine-tuning, BERT requires labelled data and a classification layer on top. It is trained by optimizing the cross-entropy loss for the classification task using backpropagation. Pre-trained BERT is typically finetuned with gradient descent on a specific dataset, where it learns task-specific parameters while retaining general language understanding.

3) 4.3. Logistic Regression

- **Definition**: Logistic regression is a statistical model used for binary classification that predicts the probability of a data point belonging to one of two classes. It applies the logistic function to a linear combination of input features to model the likelihood of the classes.
- **Model Training**: Training involves maximizing the likelihood of the observed data using a loss function called binary cross-entropy (or log-loss). An optimization algorithm, typically gradient descent, adjusts weights to minimize the loss, with the model iteratively improving its fit to the training data.

4) 4.4. Random Forest

- **Definition**: Random Forest is an ensemble learning method that constructs a collection (or forest) of decision trees. Each tree is trained on a random subset of the data and features, and the final prediction is based on the majority vote across all trees in the forest.
- **Model Training**: Random Forest is trained by creating multiple decision trees on bootstrapped samples from the training data. Each tree splits nodes by choosing the best split among a random subset of features, reducing overfitting and increasing model robustness. Once all trees are trained, the model aggregates their predictions for final **output.**

5) 4.5. Decision Tree

- **Definition**: A decision tree is a model that makes decisions by splitting data into subsets based on feature values. Each internal node represents a feature, branches represent decisions, and leaf nodes represent the outcome or class label.
- **Model Training**: Training a decision tree involves recursively partitioning the data by selecting features that maximize information gain (or minimize Gini impurity) at each node. The process continues until nodes reach a specified depth, become homogeneous, or can no longer split. Pruning techniques are often used to improve generalization and prevent overfitting.

V REQUIREMENT ANALYSIS

5.1 Function and non-functional requirements

Requirement's analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and non-functional requirements.
Functional Requirements: These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements. Examples of functional requirements:

1) Authentication of user whenever he/she logs into the system 2) System shutdown in Solar prediction.
3) A verification email is sent to user whenever he/she register for the first time on some software system. **Non-functional requirements**: These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioural requirements. They basically deal with issues like:

- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability Flexibility

Examples of non-functional requirements:

1) Emails should be sent with a latency of no greater than 12 hours from such an activity.
2) The processing of each request should be done within 10 seconds
3) The site should load in 3 seconds whenever of simultaneous users are > 10000

5.2 Hardware Requirements

Processor	- I3/Intel Processor
Hard Disk	- 160GB
Key Board	- Standard Windows Keyboard
Mouse	- Two or Three Button Mouse
Monitor	- SVGA
RAM	- 8GB

5.3 Software Requirements

- Operating System : Windows 7/8/10 • Programming Language : Python
- Libraries : Pandas, Numpy, scikit-learn.
- IDE/Workbench : Visual Studio Code.

VI SYSTEM DESIGN AND TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the
Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.1 Feasibility study

The feasibility study for this project assesses the practicality and viability of implementing advanced machine learning techniques for real-time fuel consumption prediction and driving profile classification using ECU data. Technical feasibilityis high due to the availability of robust machine learning libraries and tools for algorithms like Random Forest and AdaBoost. The project benefits from existing infrastructure for data collection and processing within vehicles, ensuring that the necessary data for training and validation is accessible. Economic feasibility is supported by the potential cost savings from optimized fuel consumption and improved vehicle performance, which outweighs the initial investment in development and implementation. Operational feasibility is promising, given the existing expertise in machine learning and data analytics within the team. Legal and ethical considerations are addressed by ensuring data privacy and compliance with regulations. Overall, the project is feasible and holds significant potential for enhancing vehicle efficiency and environmental impact.

6.2 Types of test & Test Cases

6) 6.2.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7) 6.2.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

8) 6.2.3Functional testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:
Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.
Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

9) 6.2.4 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

10)6.2.5 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

11)6.2.6 Test cases

S.NO	Test cases	I/O	Expected O/T	Actual O/T	P/F
1	Read the dataset.	Dataset path.	Dataset need to read successfully.	Dataset fetched successfully.	P
2	Performing Loading on the dataset	Data loading takes place	Data loading should be performed on system	Data loading successfully completed.	P
3	Performing data preprocessing	Dataset is provided to process the data	Processed data should predict either of two classes	Processed data will successfully completed	P
4	Model Building	Model Building for the clean data	Need to create model using required algorithms	Model Created Successfully.	P
5	Input prediction and profile classification	Input is provided with all numerical values	Based on the input data prediction is done	Predicted successfully	P

VII CONCLUSION

In conclusion, the project aims to address the increasing difficulty of differentiating between human-authored and machine generated content in a multilingual context. By developing a system that leverages both traditional machine learning algorithms (e.g., Logistic Regression, Decision Tree, and Random Forest) and advanced deep learning models (e.g., LSTM and BERT), this approach integrates the strengths of each model type to improve accuracy and efficiency. Utilizing a comprehensive dataset across languages like English, Indonesian, German, and Russian, the system is designed to provide reliable and scalable detection of machine-generated text. This project ultimately contributes to content verification tools, supporting trust in digital information and enabling more secure and credible digital environments across various linguistic and cultural contexts.

REFERENCES

1. **Han, T. R., & Kim, J. H. (2020).** "A Survey on Text Classification: From Shallow to Deep Learning." IEEE Access, 8, 24430-24448.
2. **Wu, Y., Wang, X., & Xu, X. (2022).** "Multilingual Text Classification with Transformer Models." IEEE Transactions on Knowledge and Data Engineering, 34(6), 1019-1032.
3. **Al-Hadhrami, F., Idris, M. I., & Al-Kahtani, A. (2020).** "A Review of Text Classification Algorithms for Detecting Fake News." IEEE Access, 8, 134055-134070.
4. **Li, J., Liu, Z., & Wang, H. (2022).** "Cross-lingual Text Classification Using Multi-View Learning." IEEE Transactions on Neural Networks and Learning Systems, 33(4), 1607-1620.
5. **Sharma, N., Singh, S. S., & Kumar, V. P. (2022).** "Deep Learning for Multilingual Text Classification: A Comparative Study." IEEE Transactions on Emerging Topics in Computing, 10(2), 160-172.
6. **Gupta, R., Sharma, S., & Verma, S. (2019).** "Automatic Detection of Machine-Generated Text Using Neural Networks." IEEE Access, 7, 164931-164939.
7. **Singh, A., Gupta, B. K., & Singh, M. R. (2021).** "Language-Agnostic Machine Learning Techniques for Text Classification." IEEE Transactions on Computational Social Systems, 8(2), 350-359.
8. **Wang, C., Chen, Z., & Yu, J. (2021).** "Multilingual Text Representation Learning for Cross-Lingual Text Classification." IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(9), 3111-3122.
9. **Lee, H. S., Park, J. K., & Kim, M. H. (2021).** "Detecting Automated Content Generation: A Comprehensive Survey." IEEE Access, 9, 87545-87560.
10. **Patel, K. N., George, A. J., & Zhang, N. L. (2022).** "Enhanced Text Classification Models for Detecting MachineGenerated Text." IEEE Transactions on Artificial Intelligence, 3(1), 65-77.