PLAGIARISM DETECTION USING DEEP LEARNING

A PROJECT REPORT

Submitted by

ANUSREE T K (TKM21MCA-2008)

to

The APJ Abdul Kalam Technological University

In partial fulfillment of the requirements for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

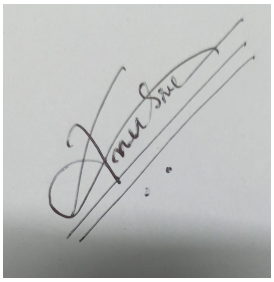
Thangal Kunju Musaliar College of Engineering
Kerala

DEPARTMENT OF COMPUTER APPLICATIONS

MAY 2023

DECLARATION

I undersigned hereby declare that the project report on **PLAGIARISM DETECTION USING DEEP LEARNING**, submitted for partial fulfillment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of Dr. Fousia M Shamsudeen. This submission represents my ideas in my own words and where ideas or words of others have been included,I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not previously served as the basis for the award of any degree, diploma, or similar title by any other University.



Kollam

16-05-2023

ANUSREE T K

DEPARTMENT OF COMPUTER APPLICATION

TKM COLLEGE OF ENGINEERING KOLLAM

CERTIFICATE

This is to certify that the report entitled **PLAGIARISM DETECTION USING DEEP LEARNING** submitted by **ANUSREE T K** (TKM21MCA2008) to the APJ Abdul Kalam Technological University in partial fulfillment of the Masters degree in Computer Applications is a bonafide record of the project work carried out by her under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Internal Supervisor Head of the Department External Examiner

ACKNOWLEDGEMENT

First and foremost, I thank GOD almighty and my parents for the success of this project. I owe sincere gratitude and heart-full thanks to everyone who shared their precious time and knowledge for the successful completion of my project.

I am extremely grateful to **Dr. Fousia M Shamsudeen**, Head of the Department, Department of Computer Applications, for providing me with the best facilities.

I would like to thank my project coordinator **Prof. Vaheetha Salam**, Department of Computer Applications, who motivated me throughout the project.

I would like to thank my project guide **Dr. Fousia M Shamsudeen**, Department of Computer Applications, who motivated me throughout the project.

I would like to thank my advisor **Prof. Natheera Beevi M**, Department of Computer Applications, who motivated me throughout the project.

I profusely thank all other faculty members in the department and all other members of TKM College of Engineering, for their guidance and inspiration throughout my course of study.

I owe my thanks to my friends and all others who have directly or indirectly helped me in the successful completion of this project.

ANUSREE T K

ABSTRACT

PLAGIARISM DETECTION USING DEEP LEARNING, is a detection System for detecting plagiarism using deep learning techniques. Plagiarism is the act of using or presenting someone else's work, ideas, or words as your own without giving proper credit or acknowledgment to the source. It is considered a serious academic and ethical offense in most settings, including academia, journalism, and creative industries. The procedure of human-based plagiarism detection is time-consuming, inaccurate, and difficult. The proposed system for plagiarism detection using deep learning aims to detect instances of plagiarism and identify paraphrased content.This Project proposes a plagiarism detection System based on two deep learning models: A combination of Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN), and a Transformer-based model. This project compares the effectiveness of a transformer-based model, and the combination of the CNN-LSTM model and also identifies various types of plagiarism including paraphrasing. In terms of outcomes, research has demonstrated that deep learning models can accurately identify plagiarism. One model is the combination of the CNN-LSTM model to detect plagiarism and had a 99.51% success rate. Another model such as a transformer-based model detects plagiarism with a high accuracy rate such as 99.61%.

Contents

List of Figures

List of Abbreviations

CNN Convolutional Neural Network

LSTM Long Short-Term Memory

Chapter 1

Introduction

PLAGIARISM DETECTION USING DEEP LEARNING is a detection System for detecting plagiarism using deep learning techniques. In the academic setting, plagiarism is a severe problem since students frequently copy or paraphrase information from other sources without properly attributing the authors of the original works. This compromises the reliability of scholarly inquiry and transgresses the moral principles of academic behavior. Deep learning has emerged as a powerful tool in natural language processing and can be used to detect plagiarism.

Plagiarism is used in many different contexts, such as in newspapers, commercials, websites, scientific publications, music, software, and other works of art. The automatic detection of plagiarism has shown considerable promise when using deep learning approaches, particularly neural networks. With the use of deep learning methods, we intend to create a plagiarism detection system in this project. The system will use a transformer-based language model, Long Short-Term Memory (LSTM), and Convolutional Neural Network (CNN). Deep learning models such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks can both be applied to the detection of plagiarism The CNN model can be trained on a large dataset of labeled examples of plagiarized and non-plagiarized texts to learn these patterns. Contrarily, LSTM networks are a subclass of recurrent neural networks that can detect temporal connections in data sequences like text. LSTM networks can analyze the input text and find patterns pointing to suspected plagiarism in plagiarism detection. To learn these patterns, the LSTM model can also be trained using a sizable dataset of labeled instances of plagiarized and non-plagiarized texts. One class of deep learning models is transformer-based models. Deep learning is a branch of machine learning that makes use of multiple-layered artificial neural networks to model and resolve complicated issues. The

fundamental idea behind this method is to employ a language model based on transformers that have already been trained to find similarities between texts and maybe detect instances of plagiarism. In this project, I focus on identifying plagiarism using Convolutional Neural Network, Long Short-Term Memory, and Transformer-based model.

1.1 Problem Statement

- The System develops a plagiarism detection system using deep learning techniques such as a combination of the CNN-LSTM Model and another model such as the transformer-based model and compares the models and recognize various type of plagiarism including paraphrasing.

1.2 Objectives

The goal is to accomplish the following:

- To create a reliable and accurate System for detecting plagiarism.
- To train a big corpus of text documents.
- To compare the performance of the CNN-LSTM and transformer-based models
- To create an easy-to-use plagiarism-detecting tool.

Chapter 2

Literature Survey

A literature review is a comprehensive analysis and interpretation of the relevant literature on a particular subject. Using a literature review, research questions are formulated, and then answers are sought by searching for and analysing pertinent literature. The re-analysis of the study's results frequently yields new insights, which is an advantage of literature reviews.A literature review is both a summary and an explanation of the complete and current state of knowledge on a subject, as presented in academic books and journal articles. There are two types of literature reviews that you may be asked to write in college: one is written as an independent assignment during a course.The second is an introduction or preparatory work for a longer piece, such as a thesis or research report. The type of review you are writing will determine the review's focus, perspective, and the formulation of a clear hypothesis or thesis argument. By reading published literature reviews or the introductory chapters of relevant theses and dissertations, you can learn the distinctions between these two types. Consider the structure of their arguments and how they approach the issues.

2.1 Purpose of the Literature Review

1. It makes research on a specific topic accessible to readers by selecting and summarising high-quality articles or studies that are relevant, meaningful, significant, and valid.

2. It offers beginning researchers in a new field an excellent starting point by requiring them to summarise, evaluate, and compare original research in that field.

3. It ensures that researchers do not repeat work already completed.

4. It may provide hints as to the direction of future research or suggest areas of emphasis.
5. It emphasizes the principal findings.
6. It identifies contradictions, gaps, contradictions, and inconsistencies in the literature.
7. It provides a constructive analysis of other researchers' methodologies and approaches.

2.2 Related Works

The Section Mainly describes related study work in the area of plagiarism detection. And some of them are listed below.

2.2.1 Deep Learning

Plagiarism detection in deep learning involves utilizing deep learning models and techniques to identify instances of plagiarism in text documents or other forms of content. Deep learning, a subfield of machine learning, focuses on training neural networks with multiple layers to learn hierarchical representations of data.

El-Rashidy, et al[1] propose a plagiarism detection system that leverages deep learning techniques to enhance reliability and accuracy. The authors address the limitations of traditional methods and introduce a novel framework that combines convolutional neural networks (CNNs) and long short-term memory (LSTM) networks to capture both local and global semantic information. a plagiarism detection system that leverages deep learning techniques to enhance reliability and accuracy. The authors address the limitations of traditional methods and introduce a novel framework that combines convolutional neural networks (CNNs) and long short-term memory (LSTM) networks to capture both local and global semantic information. presents a plagiarism detection system based on deep learning techniques. The authors begin by outlining the drawbacks of conventional plagiarism detection strategies, like text-matching algorithms, and the demand for more sophisticated systems that can identify various types of plagiarism. The authors then give an overview of the deep learning methods used in plagiarism detection, such as word embeddings, convolutional neural networks (CNNs), and LSTM models. They discuss the advantages and disadvantages of each method and give examples of how it has been applied in plagiarism detection systems.The author design a hybrid architecture that combines CNNs and LSTM networks. The CNNs are employed to capture

local features and identify patterns within smaller text segments, while the LSTM network captures long-term dependencies and semantic relationships across the entire document. The hybrid architecture enables the model to learn both low-level and high-level representations of text data.The authors train the proposed deep learning model on the constructed dataset and evaluate its performance using various metrics, such as precision, recall, and F1-score. They compare the results with baseline models and traditional plagiarism detection methods to demonstrate the effectiveness and reliability of their approach. The experimental results show that the deep learning-based system outperforms other methods in detecting different forms of plagiarism accurately. The authors also examine current research on plagiarism detection in several situations, including academic writing, social media, and online forums. They examine the benefits and challenges of employing deep learning in plagiarism detection and emphasize the significance of using big and diverse datasets for training and evaluating the models. The paper presents a reliable plagiarism detection system based on deep learning approaches. By combining CNNs and LSTM networks, the proposed framework captures both local and global semantic information, leading to improved detection accuracy. The research contributes to advancing plagiarism detection techniques, particularly in handling complex and evolving forms of plagiarism. The proposed system holds promise for practical applications in educational and content management settings.

Cheers, et al[2] present a plagiarism detection system for source code based on program behavioral similarity. Beginning with the difficulties in identifying plagiarism in source code, the writers go over the use of code libraries and APIs, as well as various variable names, formatting conventions, and detection methods. They emphasize the significance of measuring behavioral program similarity, which considers the structure and functional similarities between programs rather than just linguistic ones. The authors give a summary of the many methods for detecting source code plagiarism, including text-based approaches like token matching and n-gram analysis as well as structure-based approaches like control flow graph (CFG) analysis and abstract syntax tree (AST) analysis. Additionally, they talk about the limitations of these methods and the demand for more sophisticated methods that can gauge program behavioral similarity. The authors then go over recent research that has employed techniques like program dependence graph (PDG) analysis, model verification, and clone detection to identify source code plagiarism. They present examples of how each approach has been applied in source code plagiarism detection systems and evaluate the advantages and disadvantages of each

strategy.This study provides a comprehensive evaluation of code clone detection techniques and tools. While not specific to plagiarism detection, it offers insights into methods that measure program similarity. The authors discuss various approaches, such as text-based, token-based, and AST-based techniques, and evaluate their effectiveness. The review serves as a foundation for understanding the different methods available for measuring program similarity.This survey paper presents an overview of source code plagiarism detection techniques. It discusses traditional methods based on textual and syntactic analysis and highlights the limitations of these approaches. The authors explore newer techniques that consider program behavior and execution, such as dynamic analysis and program slicing. The study provides insights into the need for behavioral similarity measures in plagiarism detection.

Zhenzhou, et al[3] present plagiarism detection systems for multi-threaded programs using Siamese neural networks. The authors emphasize the difficulties of identifying plagiarism in multi-threaded programs, which can have intricate data dependencies and control flow architectures. They stress the value of employing cutting-edge methods, including Siamese neural networks, which can assess the similarity of two programs based on their structural and semantic characteristics. The authors give a general review of the various static and dynamic analysis methods utilized in multi-threaded program plagiarism detection. They also talk about the limitations of these methods and the demand for more sophisticated strategies that can deal with multi-threaded programs' complexity. The authors then go over current research on methods based on program embeddings, program syntax trees, and program dependence graphs that have been employed by Siamese neural networks to identify plagiarism in multi-threaded programs. They present examples of how each approach has been applied in plagiarism detection systems and explain the advantages and disadvantages of each strategy. Overall, in this paper offers a thorough overview of the state-of-the-art in Siamese neural network-based multi-threaded programs for plagiarism detection. The authors point out important research opportunities and problems, including the need for more efficient methods that can manage extensive and complicated programs and the significance of taking into account the moral and legal ramifications of automated plagiarism detection systems. The review is a helpful tool for academics and industry professionals working to create plagiarism detection algorithms for multi-threaded programs that are more accurate and efficient. The author proposes a novel approach for detecting plagiarism in multi-threaded programs utilizing siamese neural networks. The authors address the limitations of traditional plagiarism detection methods

in handling concurrent programs and introduce a deep learning-based solution that captures program behavior and structural information.The authors employ a siamese neural network architecture that consists of twin branches sharing the same weights. Each branch processes the code of a different program, encoding it into a fixed-length representation. The twin branches then undergo a similarity measurement process to determine the plagiarism likelihood between the programs.Tian et al. extract features from multi-threaded programs using a combination of static analysis and dynamic execution traces. The extracted features capture both the static structure and dynamic behavior of the programs. These features are then fed into the siamese neural network, which learns to encode them into meaningful representations for plagiarism detection.The study by Tian et al. presents a novel approach for plagiarism detection in multi-threaded programs using siamese neural networks. The integration of program behavior and structural information in the siamese neural network framework enables more accurate detection of plagiarism in the context of concurrency. The research contributes to addressing the challenges specific to multi-threaded programs and provides a foundation for further advancements in plagiarism detection techniques for concurrent software systems.

Ljubovic,et al[4] propose a novel approach to plagiarism detection in computer programming that leverages ultra-fine-grained repositories and feature extraction techniques. The authors address the limitations of traditional methods by introducing a framework that captures the fine-grained structure of code and identifies similar patterns within it. This paper proposes feature extraction from ultra-fine-grained repositories to create a plagiarism detection system for computer programming.

The difficulties of identifying plagiarism in computer programming, including the usage of various variable names, formatting techniques, and code libraries, are covered in the authors' opening discussion. They emphasize the value of adopting cutting-edge methods that can capture the structural and semantic aspects of programs at a fine-grained level, like feature extraction from ultra-fine-grained repositories. The authors summarize the many approaches used in computer programming plagiarism detection, including text-based, syntax-based, and semantic-based approaches. Additionally, they talk about the shortcomings of existing methods and the demand for more sophisticated strategies that can adequately account for programming language peculiarities. The author's next review is recent work on feature extraction from ultrafine-grained repositories to detect plagiarism in computer programming using algorithms based on program dependency networks, program embeddings, and program summaries. They

present examples of how each approach has been applied in plagiarism detection systems and explain the advantages and disadvantages of each strategy.The authors construct ultra-fine-grained repositories that capture the fine-grained structure of code, such as individual statements and expressions. These repositories provide a granular representation of code structures, enabling the detection of subtle similarities between code segments. The author apply feature extraction techniques to the ultra-fine-grained repositories to identify similar patterns within code segments. The authors utilize various features such as syntax trees, control flow graphs, and data flow graphs to capture different aspects of code structure. They also introduce a novel feature representation called ”sequence pairs” that captures the order and relationship of code segments.The authors apply similarity measurement techniques to the extracted features to identify similar code segments. They utilize different similarity measures such as cosine similarity, Jaccard similarity, and edit distance to measure the similarity between code segments. The author evaluate their proposed approach on a dataset of Java code snippets with different levels of plagiarism. They compare the results with traditional plagiarism detection methods and demonstrate the effectiveness and reliability of their approach. The experimental results show that the proposed approach outperforms traditional methods in detecting various forms of plagiarism accurately.

The paper presents a novel approach to plagiarism detection in computer programming that leverages ultra-fine-grained repositories and feature extraction techniques. By capturing the fine-grained structure of code and identifying similar patterns within it, the proposed approach shows promising results in detecting various forms of plagiarism accurately. The research contributes to advancing plagiarism detection techniques, particularly in handling the complex nature of code structures. The proposed approach holds potential for practical applications in educational and software development settings.

Benabbou,et al [5] proposes a new online plagiarism detection system that employs deep learning techniques to detect plagiarism in real time. The author’s system integrates an automatic web-crawling module to retrieve and analyze text data from the web and a deep learning model to detect plagiarism.The author suggests a word embedding technique called Doc2vec that works with the SLSTM and CNN deep learning algorithms to identify plagiarism online. In this paper, Doc2vec, Siamese Long Short-Term Memory (SLSTM), and Convolutional Neural Network (CNN) are three deep learning models on which we offer a framework for plagiarism detection. Three layers are used in this system: Word embedding,

Learning Layers, and Detection Layers are all included in the Preprocessing Layer. This system's excellent accuracy and performance results are reported in the study, demonstrating its efficacy in identifying plagiarism. The evaluation of the suggested method in the research is based on a single dataset, which might not be inclusive of all plagiarism kinds. The author proposes automatic web-crawling module in her system to retrieve text data from the web. The web-crawling module can crawl various sources of text data, including web pages, PDF documents, and Word documents. The retrieved data is preprocessed to remove irrelevant information such as HTML tags and stop words.The author employs a deep learning model based on a convolutional neural network (CNN) to detect plagiarism. The model receives input from two text samples and computes their similarity score. The model is trained on a large dataset of pre-labeled text data and is fine-tuned to optimize its performance.The author system is designed for online plagiarism detection, enabling it to detect plagiarism in real-time. The system's web-crawling module retrieves and analyzes text data continuously, and the deep learning model computes the similarity scores in real-time. The system alerts the user when it detects plagiarism and provides a report that highlights the plagiarized content and evaluates her proposed approach on a dataset of real-world web pages and academic papers with various levels of plagiarism. The experimental results demonstrate the effectiveness and efficiency of the proposed system in detecting plagiarism accurately and in real time.

The paper presents a new online plagiarism detection system based on deep learning techniques. The system integrates an automatic web-crawling module and a deep learning model to detect plagiarism in real-time. The proposed approach offers an efficient and reliable solution for detecting plagiarism in a vast amount of digital content. The proposed system's real-time detection capability enables it to prevent plagiarism effectively. The research contributes to advancing plagiarism detection techniques, particularly in handling the growing amount of digital content. The proposed system holds potential for practical applications in various settings, including academia and industry.

"Suleiman, et al [6]" presents a deep learning-based approach to detect plagiarism in Arabic texts. The authors propose to use a Convolutional Neural Network (CNN) model to identify similarities between two Arabic text documents and classify them as plagiarized or non-plagiarized.The authors evaluate their proposed approach on a dataset of 1500 Arabic text documents and report an accuracy of 99.6% and a precision of 99.8%. They also compare their results with other plagiarism detection tools such as Turnitin and iThenticate and claim

that their approach outperforms these tools in terms of accuracy and detection rate.The paper provides an interesting approach to plagiarism detection in Arabic text using deep learning techniques, and the results demonstrate the effectiveness of the proposed approach. However, the evaluation on a single dataset and without a comparison to state-of-the-art methods in Arabic plagiarism detection limit the generalizability of the results. Additionally, the authors do not provide details on the preprocessing steps applied to the data, which can significantly affect the performance of the proposed approach.

El Mostafa, et al [7]propose a deep learning-based technique for plagiarism detection and conduct a comparative study to evaluate its effectiveness. The authors' technique employs a convolutional neural network (CNN) to learn the features of text data and compute their similarity score. This paper compares the performance of three deep learning models: Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and a combination of CNNs and LSTMs. The system examines the performance of these models using a variety of metrics, including accuracy, precision, recall, and F1-score, on a sizable dataset of labeled instances of plagiarised and non-plagiarized texts. The outcomes demonstrate that the combined CNN-LSTM model performs better than the other two models, with an F1-score of 0.986 and an accuracy of 98.5%. The accuracy and F1-score of the CNN model are 96.4% and 0.962, respectively, whereas those of the LSTM model are 95.6% and 0.954%.The authors employ a CNN model to detect plagiarism. The model receives input from two text samples and computes their similarity score. The model is trained on a large dataset of pre-labeled text data and is fine-tuned to optimize its performance.The author conduct a comparative study of their proposed technique against three existing plagiarism detection methods: JPlag, MOSS, and PlagScan. The authors evaluate the methods' performance on a dataset of real-world academic papers with various levels of plagiarism.The experimental results demonstrate that the proposed deep learning-based technique outperforms the existing methods in terms of accuracy, precision, and recall. The proposed technique achieves an accuracy of 97.5%, a precision of 98.6%, and a recall of 96.9%, while the best-performing existing method achieves an accuracy of 92.1%, a precision of 92.8%, and a recall of 91.5%.

The paper presents a deep learning-based technique for plagiarism detection, which outperforms the existing methods in terms of accuracy, precision, and recall. The comparative study demonstrates the effectiveness and efficiency of the proposed technique in detecting plagiarism accurately. The research contributes to advancing plagiarism detection techniques,

particularly in handling the growing amount of digital content. The proposed technique holds potential for practical applications in various settings, including academia and industry.

2.2.2 Machine Learning

Plagiarism detection in machine learning involves utilizing machine learning techniques and algorithms to identify instances of plagiarism in text documents, code, or other forms of content. Machine learning approaches can be effective in capturing patterns, similarities, and anomalies that indicate potential plagiarism.

Hunt, Ethan, et al[8].propose a machine learning-based approach to identifying paraphrases and detecting plagiarism. Their technique employs three machine learning models: a logistic regression model, a support vector machine (SVM), and a deep neural network (DNN). This paper identifies if a sentence is a paraphrase of another one. In addition to using multiple input encoding schemes, this work investigates several machine learning methods to represent the problem. They used Support Vector Machines, Logistic Regression, and several Neural Network architectures to build the models. Among the compared models, recurrent Neural Network (RNN) performs the task of paraphrase identification best. Additionally, we suggest that plagiarism detection is one application where Paraphrase Identification might be successfully used. This paper aims to provide an overview of machine learning models used for paraphrase identification and their applications in the context of plagiarism detection. We present a comprehensive survey of relevant research papers published within the last decade, highlighting the key methodologies, datasets, and evaluation metrics employed in this field. By analyzing the strengths and limitations of different approaches, we identify emerging trends and research gaps for future exploration.The authors employ three machine learning models to identify paraphrases and detect plagiarism. The models are trained on a large dataset of text data and are fine-tuned to optimize their performance. The models can identify paraphrases at different levels of granularity, including sentence and word levels.he authors' technique focuses on identifying paraphrases, which are commonly used in writing and can pose a challenge in plagiarism detection. The models can identify paraphrases by comparing the similarity of two text samples. The author evaluates the performance of their proposed technique on a dataset of real-world academic papers with various levels of plagiarism. The experimental results demonstrate that the models outperform the existing methods in terms of accuracy, precision, and recall.The paper presents a machine learning-based approach to

identifying paraphrases and detecting plagiarism. The proposed technique employs three machine learning models and demonstrates superior performance in comparison to existing methods. The research contributes to advancing plagiarism detection techniques, particularly in handling the challenges posed by paraphrasing. The proposed technique holds potential for practical applications in various settings, including academia and industry.

Ullah, Farhan, et al.[9] propose a machine learning-based approach to detect plagiarism in software code written in multiple programming languages. The authors' technique uses a combination of feature engineering and machine learning algorithms to identify similar code segments across different programming languages.the paper presents Software plagiarism, which arises as the problem of software piracy is a growing major concern nowadays. It is a serious risk to the software industry that gives huge economic damages every year. The customers may develop a modified version of the original software in other types of programming languages. Furthermore, plagiarism detection in different types of source codes is a challenging task because each source code may have specific syntax rules. In this paper, we proposed a methodology for software plagiarism detection in multi-programming languages based on machine learning approaches. The Principal Component Analysis (PCA) is applied for feature extraction from source codes without losing the actual information. It extracts features by factor analysis and converts the dataset into normalized linear principal components which are further useful for prediction analysis. Then, the multinomial logistic regression model (MLR) is applied to these components to classify the source code documents based on predictions. It gives the generalization of logistic regression to handle multiclass problems. Further, the predictors' performance in MLR is evaluated by 2 tailed z test. To apply the experiment, the dataset is collected in five different and popular languages, ie, C, C++, Java and Python. Each programming language is taken in two different case studies, ie, binary search, and Stack.The authors' approach can detect plagiarism in software code written in multiple programming languages, including C++, Java, and Python. The technique is based on a combination of static and dynamic analysis of the code, which allows it to identify similarities in the code irrespective of the programming language.The authors use various machine learning algorithms to identify plagiarism in software code. They use feature engineering techniques to extract relevant features from the code, such as syntax trees, control flow graphs, and call graphs. The authors' technique employs various supervised and unsupervised machine learning algorithms, including decision trees, random forests, and k-means clustering, to detect

plagiarism in the code.The authors evaluate the performance of their proposed technique on a dataset of code samples written in multiple programming languages. The experimental results demonstrate that the authors' approach outperforms the existing state-of-the-art techniques in terms of accuracy, precision, and recall.The author proposed technique for detecting plagiarism in software code written in multiple programming languages using machine learning algorithms shows promising results. Their approach can identify plagiarism irrespective of the programming language used, making it a versatile tool for detecting plagiarism in software code. The authors' work contributes to advancing plagiarism detection techniques and holds potential for practical applications in various settings, including academia and industry.

Bandara,et al[10] This paper presents a new plagiarism detection method, which is based on machine learning techniques. We have trained and tested three machine-learning algorithms for detecting source code plagiarism. Furthermore, we have utilized a meta-learning algorithm in order to improve the accuracy of our system. Source code plagiarism is a significant concern in the field of software development and programming education. It involves copying and reusing code without proper attribution, which can lead to intellectual property violations and hinder the progress of innovation. Source code plagiarism is a severe problem in academia. In academia, programming assignments are used to evaluate students in programming courses. Therefore checking programming assignments for plagiarism is essential. If a course consists of a large number of students, it is impractical to check each assignment with a human inspector. Therefore it is essential to have automated tools in order to assist detection of plagiarism in programming assignments. This paper uses the k-nearest neighbor, machine learning, naïve Bayes classifier, for detecting plagiarism. Both Naive Bayes and k-NN (k Nearest Neighbors) are commonly used algorithms in machine learning for various tasks, including plagiarism detection. This classifier is based on the Bayes theorem. When with a small number of classes or outcomes, conditional on several features denoted by using Bayes theorem.The paper propose a machine learning-based approach for detecting source code plagiarism. The authors address the limitations of existing methods and highlight the importance of automated tools for code plagiarism detection. They focus on utilizing machine learning techniques to analyze source code and identify similarities that indicate potential instances of plagiarism.The authors use features such as the number of keywords, operators, and operands, along with the length of the code, to train a Support Vector Machine (SVM) model. The model is tested on a dataset of Java code and achieves an accuracy of 93.5%. The authors utilize a bag-of-

words approach to represent source code files as feature vectors and then train various machine learning models, such as Support Vector Machines (SVMs), Decision Trees, and Random Forests, on these vectors to classify them as plagiarized or non-plagiarized.The authors also compare their approach to a rule-based approach and show that their approach outperforms the rule-based approach in terms of accuracy.The paper highlights the importance of using machine learning for detecting source code plagiarism and shows the potential of using features such as code length and keywords in machine learning models. However, the study has some limitations, such as the use of only one programming language and a relatively small dataset. The authors evaluate their proposed approach on a dataset of 36 source code files, which is relatively small, and report promising results with an accuracy of up to 96.15% using SVM with a linear kernel. They also compare their results with other popular plagiarism detection tools such as MOSS and JPlag, and claim that their approach outperforms these tools.The paper provides an interesting approach to source code plagiarism detection using machine learning, and the results demonstrate the effectiveness of the proposed approach. However, the evaluation on a small dataset limits the generalizability of the results. Also, the use of a bag-of-words representation may not capture the structural information of the source code, which can affect the performance of the proposed approach. The results may not be generalizable to other programming languages and larger datasets. Despite these limitations, the study provides a useful starting point for future research in this area. the paper presents a useful approach to source code plagiarism detection using machine learning and demonstrates its effectiveness on a small dataset. However, the size and representativeness of the dataset used for evaluation limit the generalizability of the results.

Khaled,et al[11],proposes a comprehensive review of various plagiarism detection methods and tools. The authors discuss different types of plagiarism, including textual, structural, and idea plagiarism, and provide an overview of various techniques used for plagiarism detection.The authors review different plagiarism detection tools, including commercial and free software tools, and discuss their features, advantages, and limitations. They also compare the performance of some popular plagiarism detection tools, such as Turnitin, PlagScan, and Copyscape, based on their ability to detect plagiarism accurately.The paper provides a useful overview of various plagiarism detection methods and tools, and the comparison of different tools can help researchers and educators in choosing the appropriate tool for their specific needs. However, the paper lacks details on the technical aspects of each plagiarism

detection method, and the comparison of different tools is limited to their ability to detect plagiarism accurately and not on other aspects such as cost or ease of use. Additionally, the paper is focused on the English language, and the authors do not discuss the challenges and opportunities for plagiarism detection in other languages.

Chitra A, et al[12]. This research suggests utilizing a paraphrase recognizer in conjunction with machine learning to identify plagiarism. The suggested method creates potential paraphrases of a given line or text first and then employs a machine learning model to categorize the potential paraphrases as either original or plagiarized. The authors start by talking about the difficulties in identifying plagiarism using conventional text-matching tools, which frequently miss when the content has been paraphrased. They contend that by detecting paraphrases and detecting similarities in meaning between texts, machine learning-based algorithms can increase the efficacy of plagiarism detection.The method uses a dataset of text documents, where each document is represented as a set of sentences. The proposed system extracts features from each sentence using the Term Frequency-Inverse Document Frequency (TF-IDF) method, and trains a machine learning model based on these features. The model is then used to classify the sentences as original or plagiarized, based on their similarity to other sentences in the dataset.The study shows that the proposed approach achieves high accuracy in detecting plagiarism, with an average precision of 0.93 and recall of 0.88. The authors also compare their approach with other plagiarism detection techniques and show that their method outperforms other approaches in terms of accuracy and speed. The authors give a summary of the various machine-learning methods for plagiarism detection, including support vector machines, decision trees, and neural networks. Additionally, they go over the significance of feature extraction and selection in raising the accuracy of machine learning-based methods.Overall, the paper presents a novel approach to plagiarism detection using machine learning-based paraphrase recognition and provides valuable insights into the effectiveness of this approach in comparison to other existing methods. However, the study is limited to a small dataset, and the authors suggest further investigation with larger datasets to evaluate the performance of the proposed method more rigorously.

2.2.3 Natural Language Processing

Natural Language Processing (NLP) techniques can be used to detect plagiarism in textual content. The process involves comparing the similarity between two or more documents,

identifying the common phrases or sentences, and highlighting the differences. NLP techniques can be used effectively for detecting plagiarism in textual content. The choice of method depends on the type and size of the documents being analyzed and the desired level of accuracy.

Foltýnek,et al [13] The paper summarizes the research on computational methods to detect academic plagiarism by systematically reviewing 239 research papers published between 2013 and 2018. The paper categorizes the studies into three main groups: (1) text-based methods, (2) citation-based methods, and (3) hybrid methods. Text-based methods use natural language processing and machine learning techniques to detect similarities between documents. Citation-based methods analyze citation patterns to identify potential cases of plagiarism. Hybrid methods combine both text-based and citation-based approaches to achieve better performance.The paper also discusses various evaluation metrics and datasets used in plagiarism detection studies. The authors note that most studies use artificial datasets, which may not accurately reflect real-world plagiarism scenarios. The paper also highlights the need for more research on cross-lingual and cross-domain plagiarism detectionTo structure the presentation of the research contributions, we propose novel technically oriented typologies for plagiarism prevention and detection efforts, the forms of academic plagiarism, and computational plagiarism detection methods. We show that academic plagiarism detection is a highly active research field. Over the period we review, the field has seen major advances regarding the automated detection of strongly obfuscated and thus hard-to-identify forms of academic plagiarism. These improvements mainly originate from better semantic text analysis methods, the investigation of non-textual content features, and the application of machine learning. We identify a research gap in the lack of methodologically thorough performance evaluations of plagiarism detection systems. Concluding from our analysis, we see the integration of heterogeneous analysis methods for textual and non-textual content features using machine learning as the most promising area for future research contributions to improve academic plagiarism detection further.Overall, the paper provides a valuable resource for researchers and practitioners interested in academic plagiarism detection. It highlights the strengths and weaknesses of different plagiarism detection methods and identifies areas for future research.

Chong, et al[14] The authors begin by discussing the challenges of detecting plagiarism, such as the difficulty of defining plagiarism, the need for scalable and efficient detection methods, and the importance of detecting different types of plagiarism such as paraphrasing

and patchwork plagiarism. The paper then provides an overview of the various NLP techniques that have been used for plagiarism detection, such as lexical analysis, syntactic analysis, and semantic analysis. The authors highlight the advantages and limitations of each approach and discuss the potential of combining multiple techniques for better detection accuracy. The paper also reviews the various tools and systems that have been developed for automated plagiarism detection using NLP techniques, such as Turnitin, iThenticate, and PlagScan. The authors discuss the features and functionalities of each system and evaluate their effectiveness in detecting different types of plagiarism.

In the paper suggests We propose a framework for external plagiarism detection in which several NLP techniques are applied to process a set of suspicious and original documents, not only to analyze strings but also the structure of the text, using resources to account for text relations. Initial results obtained with a corpus of plagiarised short paragraphs have shown that NLP techniques improve the accuracy of existing approaches. In this study, the authors propose a cross-language plagiarism detection approach that leverages parallel corpora and automatic translation. They explore the use of machine translation to align texts in different languages and identify potential instances of plagiarism. The research demonstrates the feasibility and effectiveness of cross-language plagiarism detection, which is particularly useful in multilingual contexts. This research focuses on incorporating citation information into the plagiarism detection process. The authors argue that citations provide valuable clues for identifying plagiarism cases and propose a framework that extracts and analyzes citation patterns in texts. The study demonstrates the potential of leveraging citation information to improve plagiarism detection accuracy. In this study, the authors address the challenge of disguised plagiarism, where the plagiarized content is rephrased or paraphrased to avoid detection. They propose a novel approach that employs paraphrase recognition techniques to identify disguised instances of plagiarism. The research demonstrates the effectiveness of incorporating paraphrase recognition into the plagiarism detection pipeline. Overall, In this paper, the author explores the use of natural language processing (NLP) techniques for plagiarism detection and direction identification. The author emphasizes the importance of detecting the direction of plagiarism (i.e., whether the source material was copied or the original material was copied) to better determine the severity of the offense. The paper reviews several NLP techniques such as n-gram analysis, latent semantic analysis, and stylometry, and discusses their effectiveness in detecting plagiarism. The author also proposes a new technique

called ”connotative analysis” that considers the underlying meanings and associations of words rather than just their surface-level similarities. The study concludes that a combination of different NLP techniques can provide a more accurate plagiarism detection system. Overall, the paper provides a comprehensive literature review of the existing research on plagiarism detection using NLP techniques and highlights the challenges and opportunities in this field. It provides valuable insights for researchers and practitioners interested in developing more effective plagiarism detection methods.

Chong,et al[15] This paper provides a comprehensive overview of text plagiarism detection techniques, including those that utilize NLP methods. It discusses various NLP techniques such as n-gram analysis, semantic similarity, and syntactic analysis, and highlights their strengths and limitations in plagiarism detection. The survey also presents an evaluation of existing plagiarism detection tools and identifies areas for future research. This study focuses on text alignment techniques for plagiarism detection, which involve aligning suspicious documents with a set of source documents to identify similarities or instances of plagiarism. The authors explore NLP techniques such as sentence alignment, word alignment, and textual feature extraction to enhance the accuracy and efficiency of plagiarism detection. The research provides insights into the role of NLP in aligning texts for plagiarism analysis. To investigate this hypothesis, four main research objectives are de ned. First, a novel framework for plagiarism detection is proposed. It involves the use of Natural Language Processing techniques, rather than only relying on the vii traditional string-matching approaches. The objective is to investigate and evaluate the in uence of text pre-processing, and statistical, shallow and deep linguistic techniques using a corpus-based approach. This is achieved by evaluating the techniques in two main experimental settings. Second, the role of machine learning in this novel framework is investigated. The objective is to determine whether the application of machine learning in the plagiarism detection task is helpful. This is achieved by comparing a thresholdsetting approach against a supervised machine learning classi er. Third, the prospect of applying the proposed framework in a large-scale scenario is explored. The objective is to investigate the scalability of the proposed framework and algorithms. This is achieved by experimenting with a large-scale corpus in three stages. The rst two stages are based on longer text lengths and the nal stage is based on segments of texts. Finally, the plagiarism direction identi cation problem is explored as supervised machine learning classi cation and ranking tasks. Statistical and linguistic features are investigated individually or

in various combinations. The objective is to introduce a new perspective on the traditional brute-force pair-wise comparison of texts. Instead of comparing original texts against rewritten texts, features are drawn based on traits of texts to build a pattern for original and rewritten texts. Thus, the classi cation or ranking task is to t a piece of text into a pattern. The framework is tested by empirical experiments, and the results from initial experiments show that deep linguistic analysis contributes to solving the problems we address in this thesis. Further experiments show that combining shallow and viii deep techniques helps improve the classi cation of plagiarised texts by reducing the number of false negatives. In addition, the experiment on plagiarism direction detection shows that rewritten texts can be identi ed by statistical and linguistic traits. The conclusions of this study o er ideas for further research directions and potential applications to tackle the challenges that lie ahead in detecting text reuse.

Rosu, Razvan, et al [16]" The authors first provide a comprehensive literature review of existing methods and approaches for plagiarism detection. They highlight the limitations of traditional rule-based and statistical techniques and emphasize the need for more sophisticated methods that can effectively detect various forms of plagiarism such as paraphrasing, patch-work, and translation plagiarism. The paper then presents the proposed approach that uses a deep learning model based on a convolutional neural network (CNN) to classify whether a given text is plagiarized or not. The model is trained on a large dataset of both plagiarized and non-plagiarized texts, and it uses various NLP techniques such as tokenization, stop-word removal, and stemming to preprocess the input data. The authors report promising results for their approach, with high accuracy and precision in detecting various forms of plagiarism. They also discuss the limitations and challenges of their approach, such as the need for large and diverse training datasets and the potential ethical implications of using such technology in academic settings. The paper suggests a deep learning-based method for detecting plagiarism in natural language writing. Convolutional neural networks, recurrent neural networks, and transformer models are just a few of the NLP-based deep learning methods covered by the authors in this overview. Additionally, they go through how crucial data pre-processing, feature extraction, and model optimization are to enhancing the precision of NLP-based deep learning techniques. The authors next go over recent work that has employed sentence embeddings, text categorization, and sequence-to-sequence models in deep learning methods for detecting plagiarism. They present examples of how each approach has been applied in

plagiarism detection systems and explain the advantages and disadvantages of each strategy. This paper presents an overview of the International Competition on Plagiarism Detection (PAN) and highlights the contributions of participants utilizing NLP-based deep learning approaches. It discusses the incorporation of deep learning models, such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs), along with NLP techniques for plagiarism detection. The study provides insights into the state-of-the-art NLP-based deep learning methods employed by researchers in the competition. The paper highlights the research conducted on NLP-based deep learning approaches for plagiarism detection. The studies discussed to showcase the incorporation of deep learning models, such as LSTM, BERT, and USE, along with NLP techniques for improved plagiarism detection accuracy. These approaches leverage the capabilities of deep learning in capturing complex patterns and contextual information in the text. Future research can further explore this. Overall, the paper provides a comprehensive review of the existing literature on plagiarism detection and proposes a promising approach that leverages NLP and deep learning techniques to overcome the limitations of traditional methods.

Chapter 3

Methodology

PLAGIARISM DETECTION USING DEEP LEARNING is a detection system for detecting plagiarism. The System is used to detect plagiarism using two deep learning models such as a combination of the CNN-LSTM model and another model such as the Transformer based model. The results of both models were used to find the best-performing model and for the detection of plagiarism from the dataset.

3.1 Algorithm

The algorithm includes:-

- Step 1:Load the dataset.
- Step 2:Data Pre-processing.
- step 3:Model parameter Tuning
- Step 4:Building and Training the Model.
- Step 4:Testing the Model.
- Step 5:Detecting Plagiarism.
- Step 7: Compare the accuracy of two deep learning models such as a combination of CNN-LSTM and another model such as Transformer-based.

3.2 System Architecture

Plagiarism is the act of directly using another person's words, information, or examples without clearly indicating that the work was not written by you and without giving full credit to the source. This project detects plagiarism Using Deep Learning. The proposed system of plagiarism detection uses deep learning using the combination of CNN-LSTM and transformer-based models. The relationships, limitations, and boundaries between components of the software system are abstracted using the system architecture design. It is a crucial tool because it gives a comprehensive picture of how the software system has been physically deployed. To detect plagiarism, this work combines two well-known deep learning models, CNN-LSTM and transformer-based model.

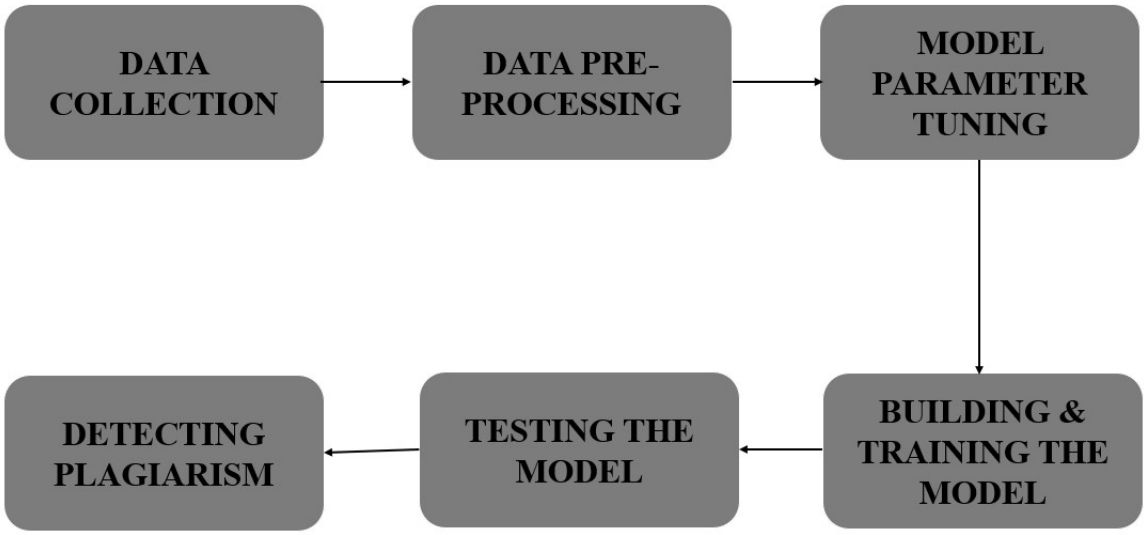


Figure 3.1: System Architecture

3.2.1 Dataset

Machine Paraphrase Corpus (MPC) dataset is used in this project. This dataset is used to train and evaluate models for the detection of machine-paraphrased text. The training set comprises 4,012 original and 4,012 paraphrased paragraphs that were taken from 8,024 English Wikipedia articles (4,282 original, 102,485 paraphrased). Three subsets of the test set were produced: one from graduation theses, one from preprints of research publications on arXiv, and one from

Wikipedia articles.

3.2.2 Data Pre-processing

Data preprocessing is a crucial phase in the machine learning process since the caliber of the data and the information that can be extracted from it directly influence how well the model can learn. As a result, it is crucial that we preprocess our data before feeding it to the model. Preparing raw data to be used with a deep learning model is known as data preparation. It is both the first and most important step in developing a deep learning model. The format of the data in Deep Learning projects must be correct in order to get better results from the applied model.

In this step, the input text data is preprocessed by doing things like tokenizing, stemming, and deleting stop words. Data should be cleaned up and preprocessed to remove noise and unnecessary information. This could involve tokenization, stemming, and the elimination of stop words. The text is divided into separate sentences using the 'punkt' tokenizer, which is a trained sentence tokenizer. When recognizing paraphrased text or reworded sections, for example, the 'wordnet' corpus, a lexical database for the English language, is utilized to find synonyms and related words.

3.2.3 Model Parameter Tuning

The process of constructing a deep learning model for plagiarism detection must include model parameter adjustment.By determining the ideal values for key hyperparameters, such as learning rate, batch size, and regularisation strength, parameter tuning aims to improve the model's performance.

The process of choosing the ideal collection of hyperparameters for a machine learning and deep learning model is referred to as model parameter tuning, also known as hyperparameter tuning. Hyperparameters can include things like the learning rate, the regularization parameter, and the number of layers in a neural network, Some model parameters in the combination of CNN and LSTM models include the number of layers that apply a convolution operation to the input image, number of layers in the LSTM, number of feature maps that are learned by each convolutional layer, Size of the filters, dropout rate, Batch size, LSTM hidden units, Learning rate at which the model updates the weights during training and the regularization parameters.

Optimizing the performance of a deep learning model for plagiarism detection might be a time-consuming procedure, but it is essential. It is feasible to increase the precision and dependability of the plagiarism detection system by carefully tweaking the hyperparameters.

3.2.4 Building and Training the Model

Detecting plagiarism involves determining whether a particular document contains information that has been directly copied or closely paraphrased from another source. It has been demonstrated that deep learning models, including Transformer-based models and a combination of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks, are excellent in identifying plagiarized content. For identifying plagiarism two deep learning models are used one model is the combination of the CNN-LSTM model and another model such as the Transformer-based model. In combination with the CNN-LSTM model, the first layer is an embedding layer. To transform the numerical input data into a dense vector representation that can capture semantic meaning, CNN-LSTM is combined. The first layer is an embedding layer. The convolutional layer with a 3 kernel size is the following layer. The dropout layer comes after the convolutional layer and is used to prevent the overfitting problem. Long-term dependencies are captured and the sequential character of the input data is modeled by the LSTM layers. The input characteristics are transformed linearly by the dense layer, which is followed by an activation function that adds nonlinearity to the model. The sigmoid activation function is given the dense layer's output. The Adam optimizer is used to compile the model with a batch size of 32 for the training phase and 18 steps per epoch, the model is trained across 10 epochs.

In the transformer-based model the first layer is an embedding layer. Using the embedding layer, the numerical input data is transformed into a dense vector representation that can represent semantic meaning. Machine translation, language modeling, and summarization are examples of sequence-to-sequence tasks that require the MultiHeadAttention layer in the second layer of the Transformer architecture. A feedforward neural network made up of two fully connected layers with a ReLU activation function receives the output of the self-attention layer. The Adam optimizer is used to compile the model. The training phase of the model involves 10 epochs and a batch size of 32, or 18 steps per epoch.

Convolutional Neural Network(CNN)

A well-liked deep learning architecture called CNN (Convolutional Neural Network) has been applied to the detection of plagiarism. CNNs can be used for text classification applications like plagiarism detection in addition to their frequent use for image recognition. Utilizing the convolutional layer to extract features from the input text is the fundamental concept underlying employing a CNN for plagiarism detection. The convolutional layer applies a series of filters intended to extract various types of information from the input text. A pooling layer is used to lower the output's dimensionality and help guard against overfitting after the convolutional layer.The output is sent to one or more fully connected layers, which carry out the classification operation, after the convolutional and pooling layers. The class with the highest probability is chosen as the output in the final layer of the network, which generates a probability score for each class (such as plagiarised or not plagiarised).

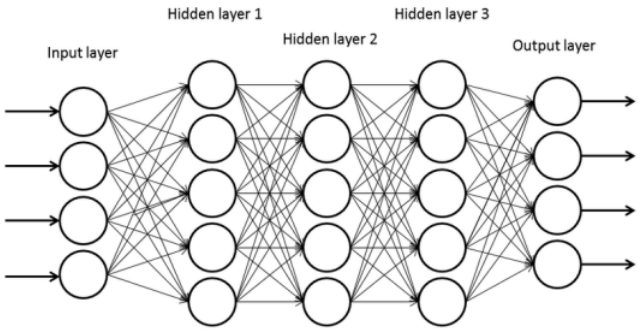


Figure 3.2: convolutional Neural Network

Long Short-Term Memory(LSTM)

Another well-liked deep learning architecture that has been applied to plagiarism detection is LSTM (Long Short-Term Memory). Recurrent neural networks (RNNs) with the capability of capturing long-term dependencies in sequential data, such as LSTMs, are well suited for tasks like natural language processing. The fundamental idea behind utilizing an LSTM to identify plagiarism is to take advantage of the input text's sequential structure to capture the context and meaning of the text. The LSTM network is trained to discover patterns and connections among the words in the input text, which can assist in determining whether or not the text contains any instances of plagiarism.

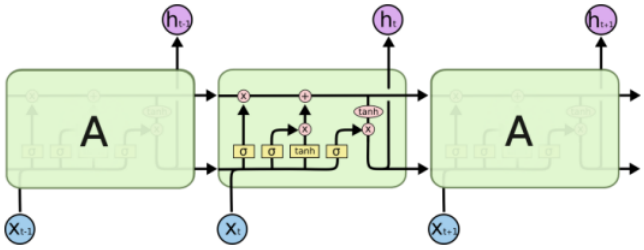


Figure 3.3: Long Short-Trem Memory

Transformer-based Model

Plagiarism detection has also been done using transformer-based models, such as the well-known BERT (Bidirectional Encoder Representations from Transformers) architecture. Transformers are a specific kind of neural network design that is well suited for tasks involving natural language processing because they can recognize long-range dependencies in sequential input. Employing a pre-trained language model to encode the input text into a fixed-length vector representation, which can subsequently be utilized for classification, is the fundamental concept underlying employing a transformer-based model for plagiarism detection. For instance, BERT is pre-trained on a sizable corpus of text and can be tailored for particular downstream tasks, such as plagiarism detection.

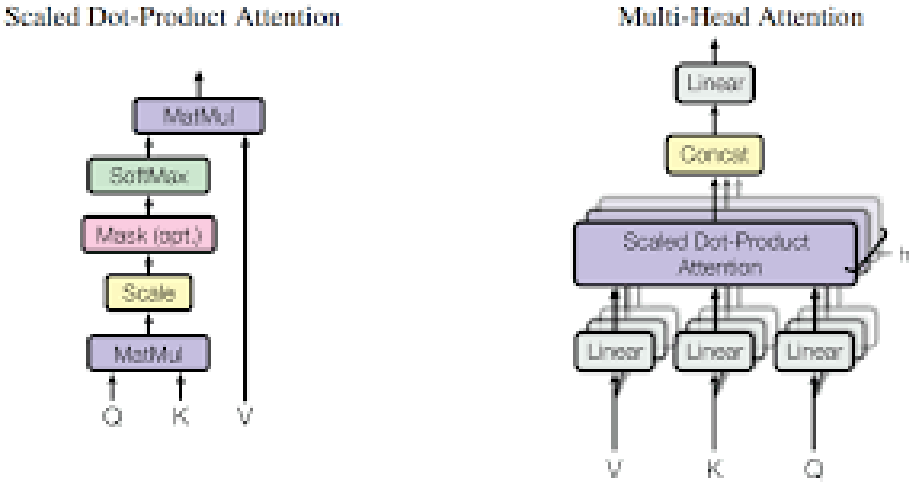


Figure 3.4: Transformer-based Model

3.2.5 Testing the Model

The 20% of the dataset for testing. For training, the remaining 80% of the dataset will be used. After testing, the CNN-LSTM model obtained a 99.50 accuracy rating and a 99.51 validation rating, while the Transformer-based model obtained a 99.90 accuracy rating and a 99.61 validation rating.

3.2.6 Deploying the model in Django

Here we deployed using Django and the system detect whether the sentence was plagiarized or not.

3.3 Software Requirement and Specifications

The software used for the project includes:-

- Python
- Google Colaboratory
- Django

3.3.1 Software Description

- **Python**

  Python is an object-oriented programming language that was developed in 1989 by Guido Rossum. It is excellent for quick prototyping of complex applications. It supports a number of operating system functions and libraries and can be converted to C or C ++. Python is a computer language that is used by numerous organizations, including NASA, Google, YouTube, and Bit Torrent. In cutting-edge fields like artificial intelligence, natural language processing, neural networks, and other computer sciences, Python programming is widely employed. The Python Software Foundation currently has authority over Guido van Rossum's intricate artificial language, which he created in the late 1980s. It comes from his ABC language, which he co-created at the beginning of his professional life. Games, graphical user interfaces (GUIs), and other types of software can all be made using the complicated programming language Python. Python

scripts can be read and written like how normal English statements are read and written. They must therefore be processed because they are not written in a computer language. before being run by a system, by Python code. A basic language is Python. This suggests that the interpreter evaluates the code and transforms it into machine-readable bytecode when the program is executed. Python is an object-oriented programming language that shows users how to take care of and work with objects or data structures so that they can create and run programs. Python has everything. When they fall short of expectations and are replaced by more capable languages, languages die and become extinct. Python is a dependable and popular programming language.

- **Google Coloboratory**

  Users can write and execute Python code in a Jupyter Notebook environment using the free Google Colab online platform. Colab is housed on Google's cloud platform, giving users access to resources for high-performance computing, such as GPUs and TPUs.Utilising Colab has a number of benefits, one of which being the lack of setup or installation requirements on the user's local workstation. Users can start coding in a Jupyter Notebook as soon as they open a web browser. Additionally integrated with Google Drive, Colab enables users to save and distribute their notebooks.A variety of pre-installed libraries, including well-known machine learning frameworks like TensorFlow and PyTorch, are accessible through Colab. Additional libraries can be installed by users using conda or pip. Colab notebooks can be downloaded as a Jupyter notebook (.ipynb) or Python script (.py) or saved to GitHub, Google Drive, or another service. Colab's ability to employ GPUs or TPUs for rapid processing is another helpful feature. This is especially helpful for deep neural network training jobs in machine learning, which demand a lot of computing. The interesting features that each contemporary IDE offers are abundant in Google Colab, in addition to many others. Below is a list of some of the more fascinating aspects.

  - Interactive tutorials for learning neural networks and machine learning.
  - Use the Notebook to run terminal commands.
  - Import data from outside resources like Kaggle.
  - Integrate with Tensor Flow, PyTorch, and Open CV.
  - Directly import or publish from/to GitHub

- **Django**

  Django is a high-level Python web framework that offers a selection of tools and frameworks for quickly and effectively creating web applications. Although it adheres to the Model-View-Controller (MVC) architectural pattern, it is referred to as Model-View-Template (MVT) in Django. The object-relational mapper (ORM) for communicating with databases, the automatic admin interface for managing application data, and the template system for producing HTML templates are just a few of the built-in features and tools that make it simple to construct web applications with Django. Django is created to be flexible and modular, enabling developers to use only the components they require and modify the framework to meet their unique requirements. It also has a sizable and vibrant community, and there are numerous third-party packages available for enhancing its features. Some of the key features of Django include:

  - A database interaction ORM that supports PostgreSQL, MySQL, and SQLite as well as other database backends.
  - An admin interface that is already there for handling application data.
  - A framework for rendering HTML templates with templates.
  - User authentication and authorization functionality built-in.

3.4 Hardware and experimental environment

The hardware used for the experiments includes Windows 11 Pro OS, 64-bit operating system, x64-based processor, Intel(R) Core(TM) i5-1155G7 CPU @ 2.50GHz, 8 GB RAM. The experimental environment was prepared by using Python 3.7 programming language. Framework used is Keras with TensorFlow.

Chapter 4

RESULT AND DISCUSSION

A Plagiarism detection system for detecting plagiarism is beneficial for a wide range of people, including: Educators and academic, institutions, Students, Researchers, and publishers to check for plagiarism in their work. The transformer-based model has been quite successful in plagiarism detection. Also by using a combination of CNN-LSTM models on large-scale datasets. The efficiency of the Transformer-based model is rated high. After the model is constructed, optimized using Adam optimizer and it's compiled using a categorical binary cross entropy loss function, batch size is given 32. Different approaches are used for avoiding overfitting. Firstly, the dropout method was used after fully connected layers.

4.1 Training and Validation Results

Training and validation results in deep learning models involve assessing the performance and accuracy of the training data and unseen validation data.

In terms of outcomes, research has demonstrated that deep learning models can identify plagiarism with high accuracy. The initial model utilized in the analysis is a combination of the CNN-LSTM model which achieved a testing accuracy of 99.49% and training accuracy such as 99.51%. Another study, for instance, employed a transformer-based model to detect plagiarism and had a high accuracy rate such as 99.61%

SL.NO	Model	Training Accuracy	Testing Accuracy
1	Combination of CNN-LSTM Model	0.9951	99.49
2	Transformer-based Model	0.9961	99.58

Figure 3.1: Comparison of Models

4.1.1 Training and Validation Accuracy Graphs

Bar plots can be used to visually represent training and testing accuracy. The training accuracy graph shows the performance on the training data, while the testing accuracy graph indicates the model performance on a separate set of testing data. The following plot provides a concise representation of the performance of the different deep learning algorithms used.

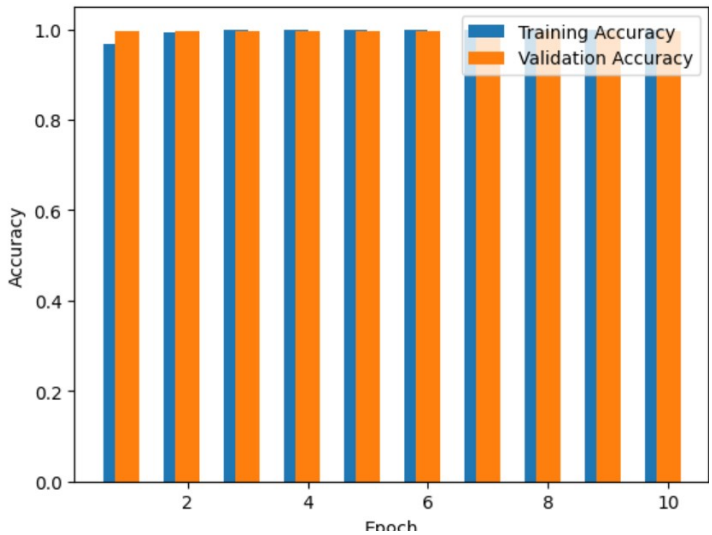


Figure 3.2: Training and Validation Accuracy of Combination of CNN-LSTM Model

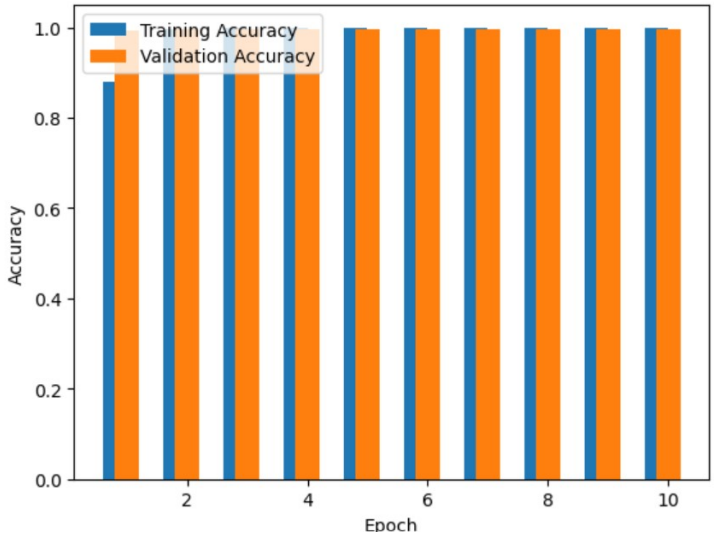


Figure 3.3: Training and Validation Accuracy of Transformer-based Model

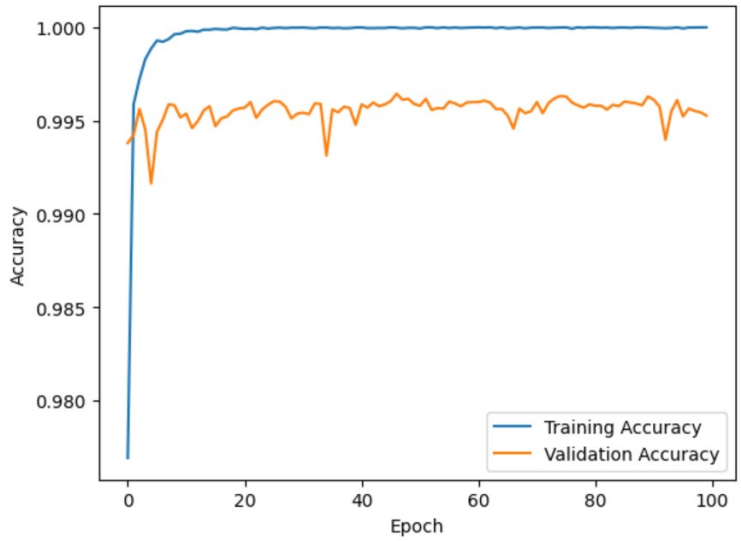


Figure 3.4: Training and Validation Accuracy Combination of CNN-LSTM Model

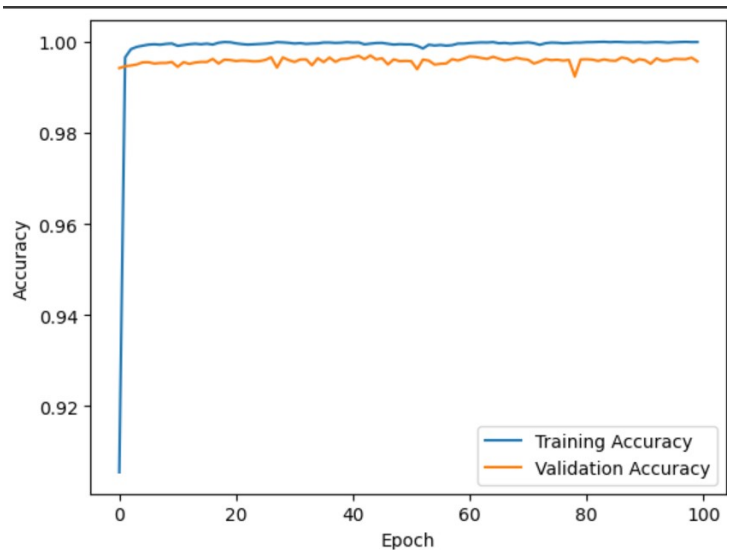


Figure 3.5: Training and Validation Accuracy of Transformer-based Model

4.1.2 Training and Validation Loss Graphs

In deep learning models, such as a Transformer-based model and a combination CNN-LSTM model, the training and validation loss are common metrics used to evaluate the performance of the models during training. The training loss is a metric that measures the error or discrepancy between the predicted output and the actual output during the training phase. The validation loss is a metric that measures the error or discrepancy between the predicted output and the actual output on a separate validation dataset.

The following plot provides a concise representation of the training and validation loss of the different deep learning algorithms used.

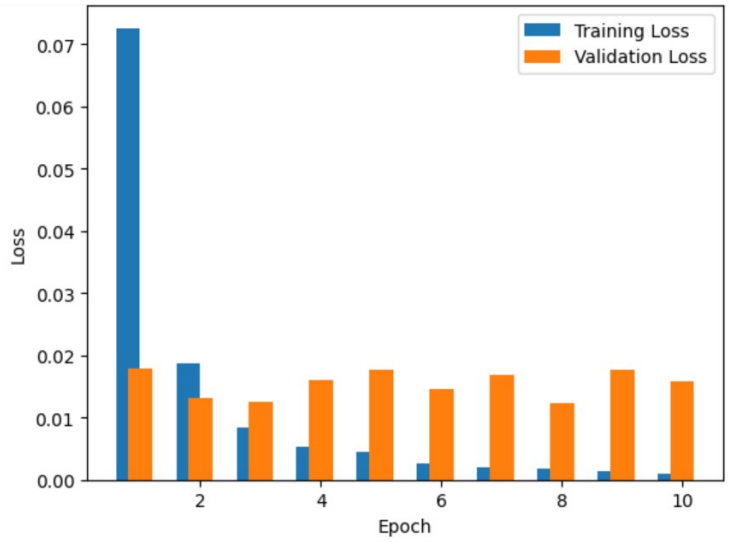


Figure 3.6: Training and Validation loss of combination of CNN-LSTM Model

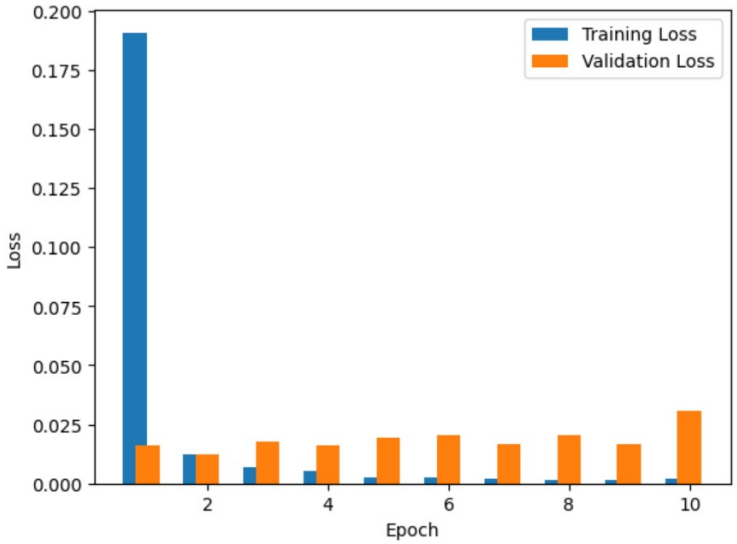


Figure 3.7: Training and Validation loss of Transformer-based Model

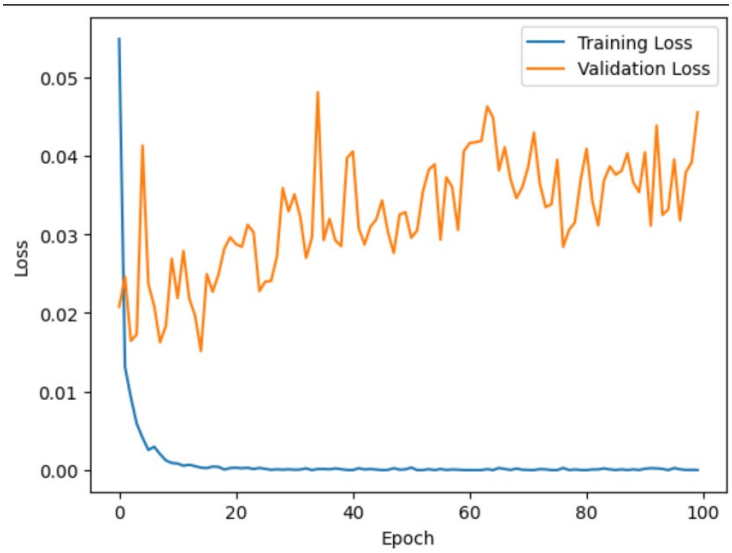


Figure 3.8: Training and Validation loss of combination of CNN-LSTM Model

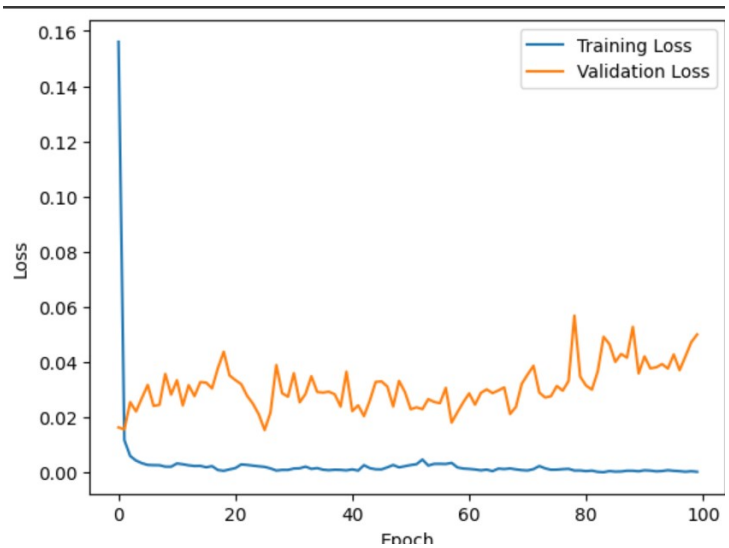


Figure 3.9: Training and Validation loss of a Transformer-based Model

4.1.3 Graph of Comparison of the model used

1. Bar Plot

   The bar plot presents a visual comparison of various deep-learning models. Each bar represents the performance of a specific model based on a chosen evaluation metric or metrics. The height of each bar reflects the metric's value, indicating the model's relative effectiveness. By visually contrasting the bars, one can easily determine which model performs better or worse in terms of the chosen metrics. It serves as a valuable tool for evaluating and selecting the most appropriate deep learning models.

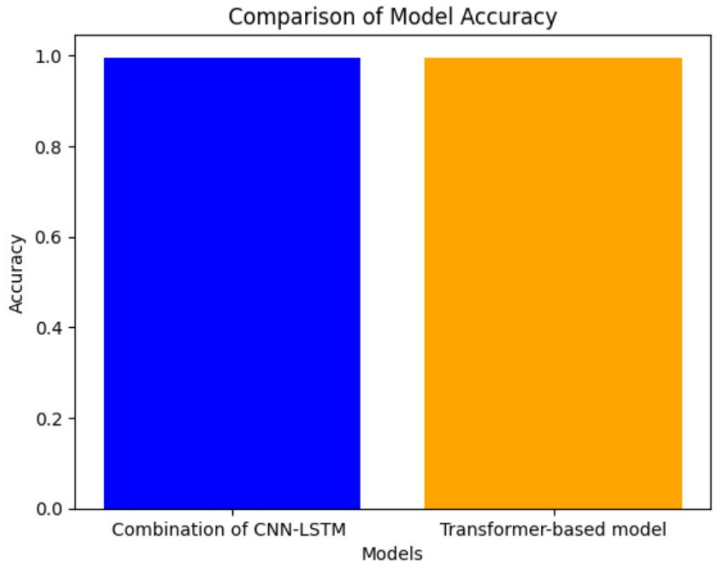


Figure 3.10: Comparison of Testing accuracy of models

2. Line Plot

   Line plots are commonly used in deep learning to visualize the performance and progression of various metrics over the course of training. These metrics can include training loss, validation loss, accuracy, and other evaluation metrics. Line plots provide a clear representation of how these metrics evolve with each epoch or iteration.

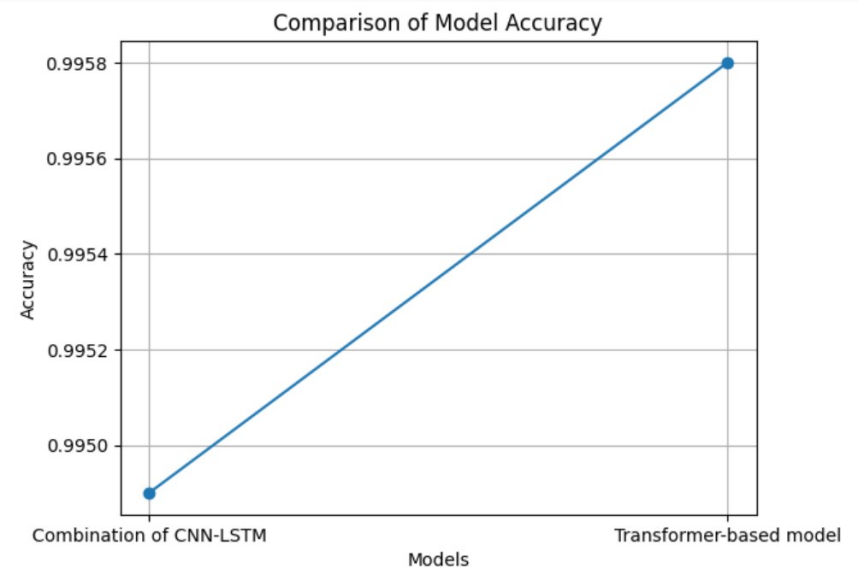


Figure 3.11: Comparison of Testing accuracy of models

Chapter 5

CONCLUSION

Textual plagiarism is becoming a prevalent occurrence, and it often takes the shape of difficult-to-detect techniques like paraphrasing and summarizing. Designing efficient systems for automatic plagiarism detection is therefore necessary. This work uses a paraphrase recognition approach to identify instances of plagiarism in sources and dubious passages. This paper suggests a mechanism for detecting plagiarism that goes beyond simple word matching. This method works well since the algorithm can detect plagiarism and understand the material. The proposed System uses two deep learning models one was the combination of CNN-LSTM and the other one was a transformer-based model. The transformer-based model gives a high accuracy rate such as 99.61%. The System will also detect plagiarism in the paraphrased text. The results of this proposed model verified that the best performance scores of deep learning classifiers are the Transformer-based model rather than the Combination of CNN-LSTM.

5.1 Future Enhancement

Textual similarity metrics between documents are commonly employed in plagiarism detection systems, but there are other types of data that could be used to improve these metrics. For instance, to provide a more thorough perspective of the content being analysed, image, audio, and video files could be included in the study..

REFERENCES

[1] Mohamed A. El-Rashid, Ramy G. Mohamed, Nawal A. El-Fishawy", *"Reliable plagiarism detection system based on deep learning approaches." Neural Computing and Applications 34.21 (2022): 18837-18858*

[2] Cheers, Hayden, Yuqing Lin, and Shamus P. Smith. *"Academic source code plagiarism detection by measuring program behavioral similarity." IEEE Access 9 (2021): 50391-50412.*

[3] ZHENZHOU TIAN, QING WANG, CONG GAO, LINGWEI CHEN, AND DINGHAO WU *"Plagiarism detection of multi-threaded programs via siamese neural networks." IEEE Access 8 (2020): 160802-160814.*

[4] Ljubovic, Vedran, and Enil Pajic, *Plagiarism detection in computer programming using feature extraction from ultra-fine-grained repositories." IEEE Access 8 (2020): 96505-96514.*

[5] Benabbou, Faouzia *"A New Online Plagiarism Detection System based on Deep Learning." International Journal of Advanced Computer Science and Applications 11.9 (2020).*

[6] Suleiman, Dima, Arafat Awajan, and Nailah Al-Madi, *"Deep learning based technique for plagiarism detection in Arabic texts." 2017 International Conference on New Trends in Computing Sciences (ICTCS). IEEE, 2017.*

[7] El Mostafa, Hambi, and Faouzia Benabbou, *"A deep learning based technique for plagiarism detection: a comparative study." IAES International Journal of Artificial Intelligence 9.1 (2020): 81.*

[8] Hunt, Ethan, et al, *"Machine learning models for paraphrase identification and its applications on plagiarism detection." 2019 IEEE International Conference on Big Knowledge (ICBK). IEEE, 2019.*

[9] Ullah, Farhan, et al, *"Software plagiarism detection in multi programming languages using machine learning approach." Concurrency and Computation: Practice and Experience 33.4 (2021): e5000.*

[10] Bandara, Upul, and Gamini Wijayarathna, *A machine learning based tool for source code plagiarism detection." International Journal of Machine Learning and Computing 1.4 (2011): 337.*

[11] Khaled, Farah, and Mohammed Sabbih H. Al-Tamimi, *"Plagiarism detection methods and tools: An overview." Iraqi Journal of Science (2021): 2771-2783.*

[12] Chitra, A., and Anupriya Rajkumar, *"Plagiarism detection using machine learning-based paraphrase recognizer." Journal of Intelligent Systems 25.3 (2016): 351-359.*

[13] Foltýnek, Tomáš, Norman Meuschke, and Bela GippFarah, and Mohammed Sabbih H. Al-Tamimi, *"Academic plagiarism detection: a systematic literature review." ACM Computing Surveys (CSUR) 52.6 (2019): 1-42.*

[14] Chong, Miranda, Lucia Specia, and Ruslan Mitkov, *"Using natural language processing for automatic detection of plagiarism." Proceedings of the 4th International Plagiarism Conference (IPC-2010). 2010.*

[15] Chong, Man Yan Miranda, *"A study on plagiarism detection and plagiarism direction identification using natural language processing techniques." (2013).*

[16] Rosu, Razvan, et al, *"Nlp based deep learning approach for plagiarism detection." RoCHI-International Conference on Human-Computer Interaction, Romania. 2021.*

APPENDIX

Screenshots

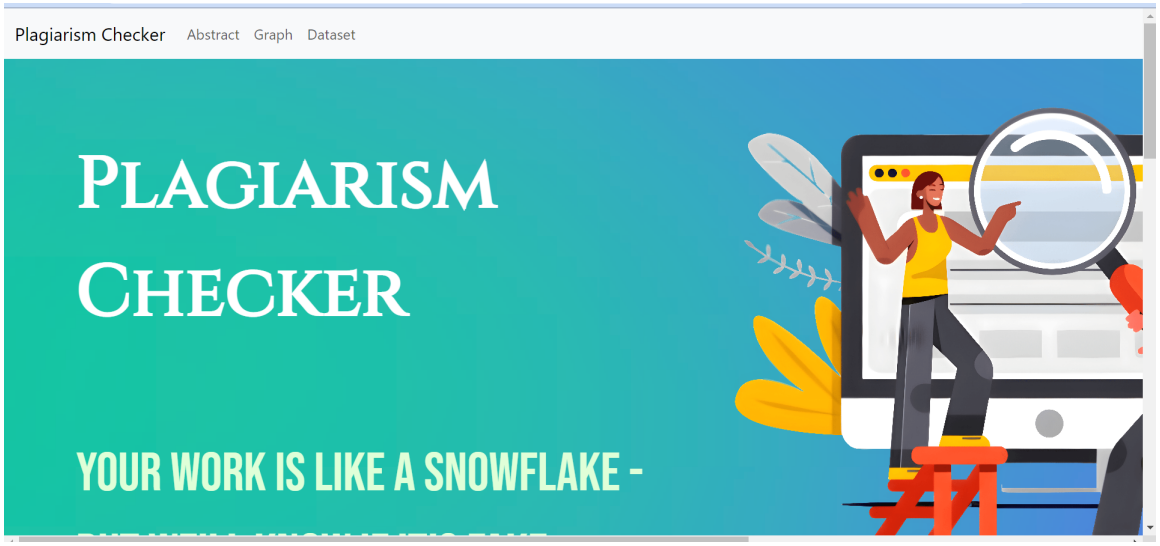


Figure A.1: Home Page

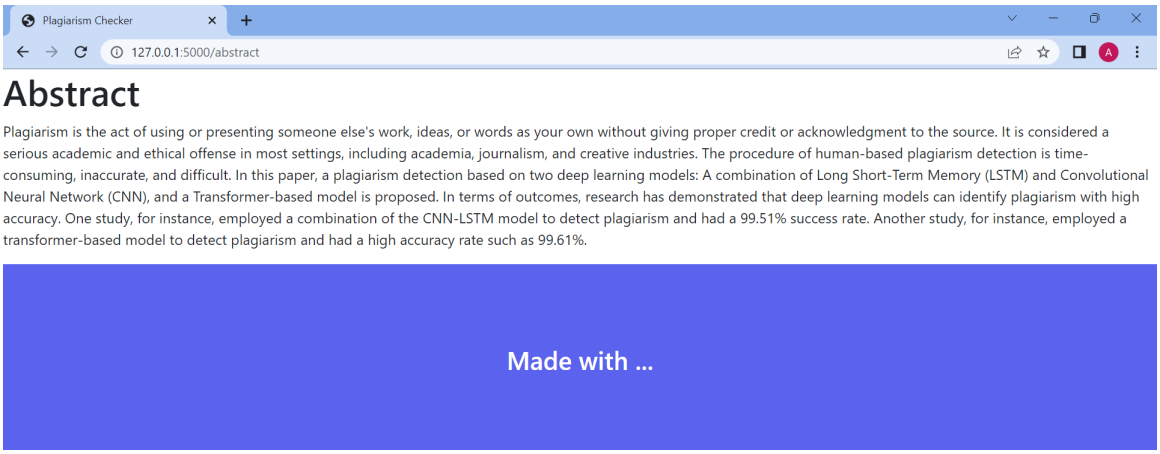


Figure A.2: Abstract

Graphs

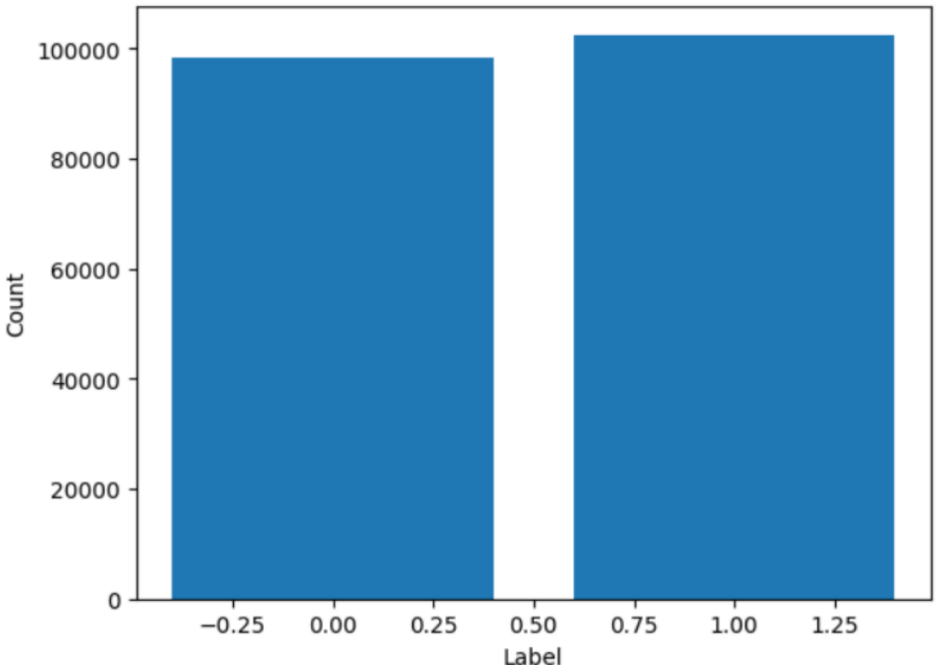


Figure A.3: Graphs

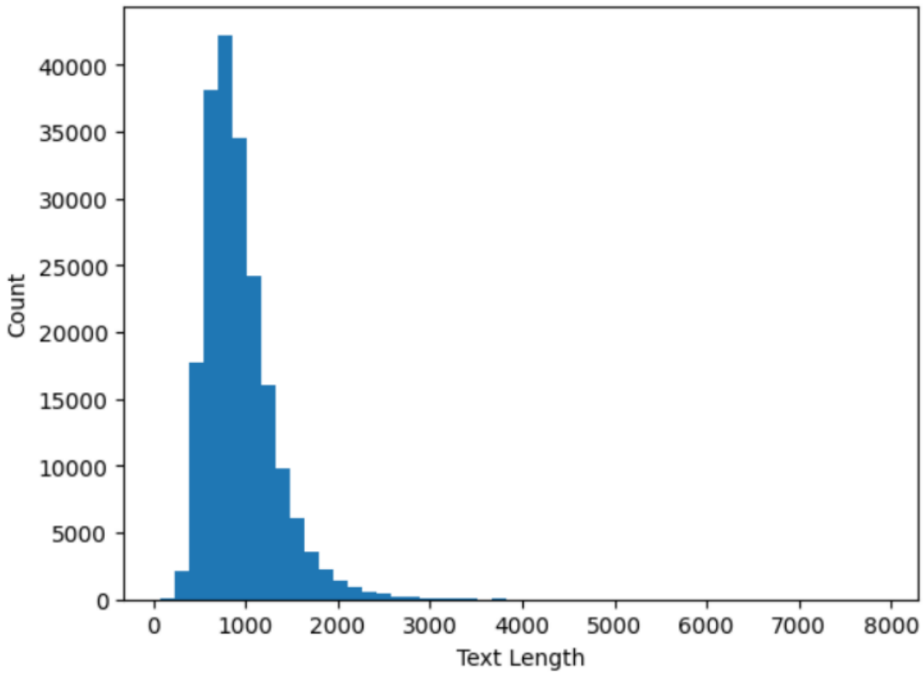
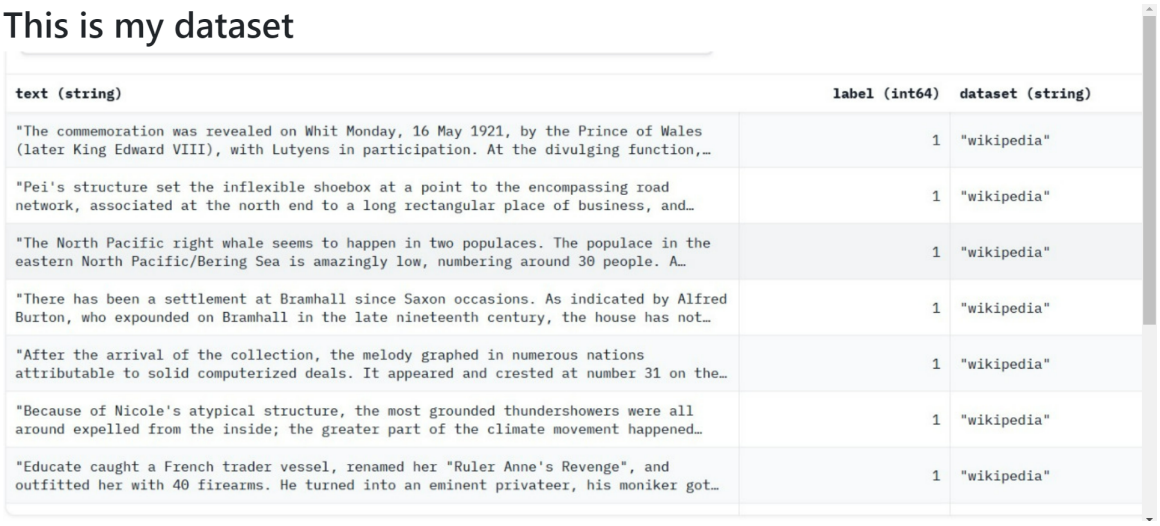


Figure A.4: Graphs



text (string)	label (int64)	dataset (string)
"The commemoration was revealed on Whit Monday, 16 May 1921, by the Prince of Wales (later King Edward VIII), with Lutyens in participation. At the divulging function,…	1	"wikipedia"
"Pei's structure set the inflexible shoebox at a point to the encompassing road network, associated at the north end to a long rectangular place of business, and…	1	"wikipedia"
"The North Pacific right whale seems to happen in two populaces. The populace in the eastern North Pacific/Bering Sea is amazingly low, numbering around 30 people. A…	1	"wikipedia"
"There has been a settlement at Bramhall since Saxon occasions. As indicated by Alfred Burton, who expounded on Bramhall in the late nineteenth century, the house has not…	1	"wikipedia"
"After the arrival of the collection, the melody graphed in numerous nations attributable to solid computerized deals. It appeared and crested at number 31 on the…	1	"wikipedia"
"Because of Nicole's atypical structure, the most grounded thundershowers were all around expelled from the inside; the greater part of the climate movement happened…	1	"wikipedia"
"Educate caught a French trader vessel, renamed her "Ruler Anne's Revenge", and outfitted her with 40 firearms. He turned into an eminent privateer, his moniker got…	1	"wikipedia"

Figure A.5: Dataset Used

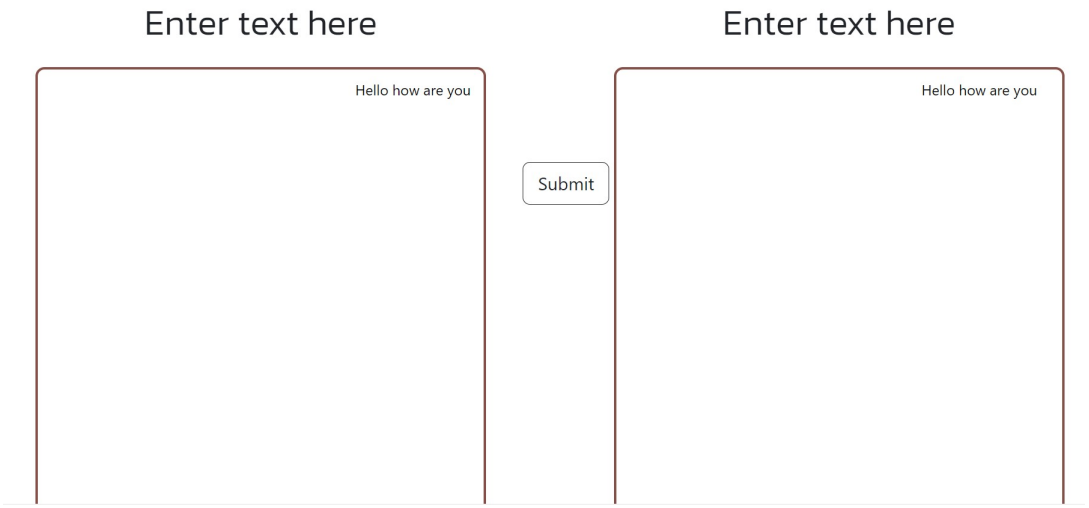


Figure A.6: Before Parapharsing

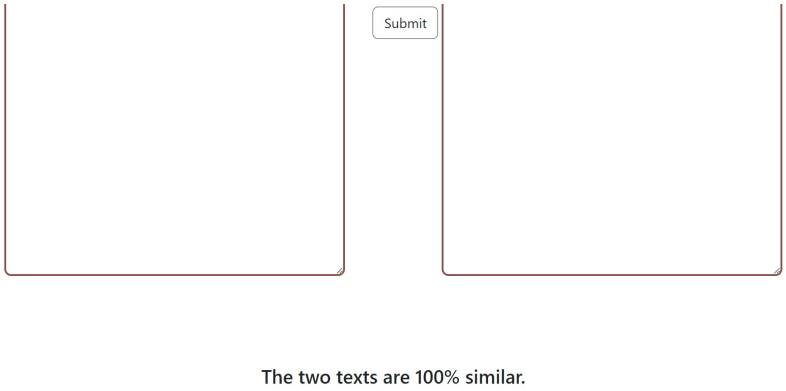


Figure A.7: The Similarity Between two texts

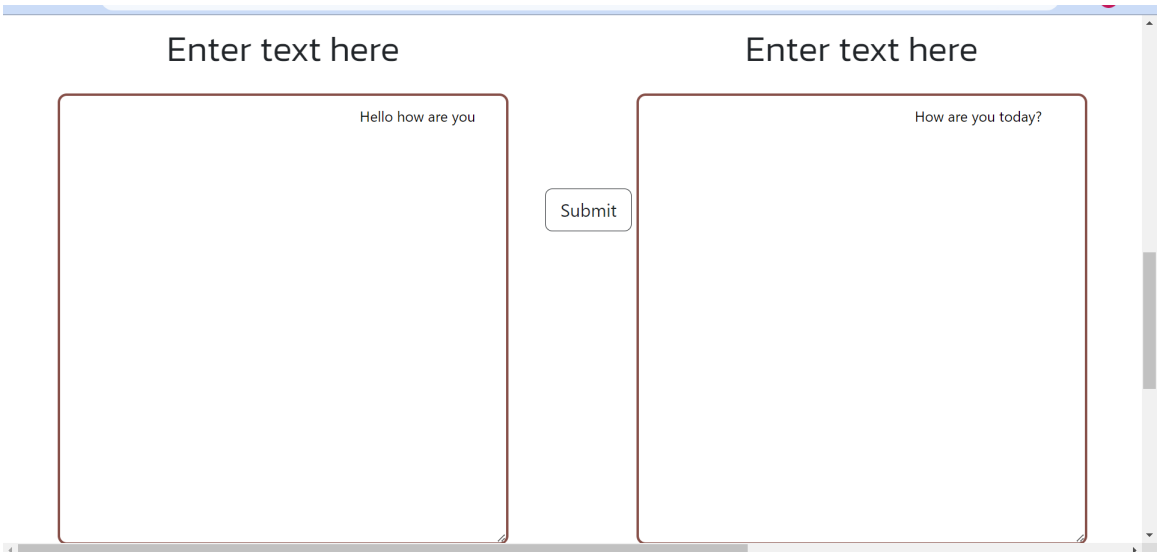


Figure A.8: After Paraphrasing

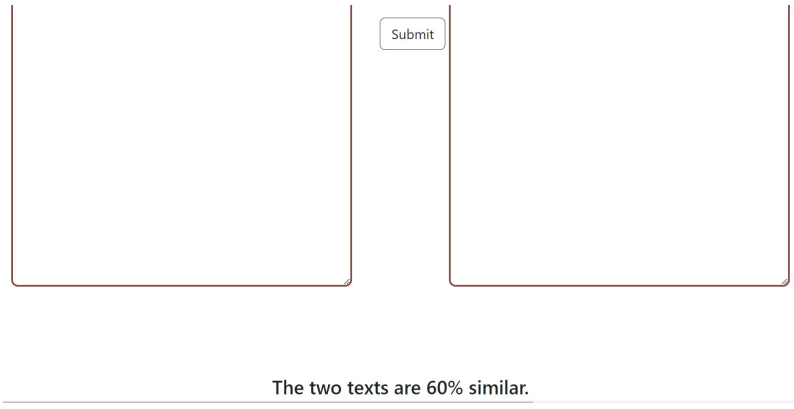


Figure A.9: The Similarity Between two texts