

PLAGIARISM DETECTION USING MACHINE LEARNING APPROACH

Submitted in partial fulfillment of the requirements for the award of Bachelor of
Engineering degree in Computer Science and Engineering

By

SIDDHARTHA REDDY (Register No. - 40111227)

KARTHIK REDDY (Register No. - 40110916)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

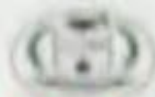
Category-1 University by UGC

Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE

JEPPIAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600119

APRIL - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **SIDDHARTHA REDDY (40111227)** and **KARTHIK REDDY (40110916)** who carried out a project entitled "**PLAGIARISM DETECTION USING MACHINE LEARNING APPROACH**" under my supervision from November 2023 to April 2024.

Internal Guide

Dr. A. SVASANGARI, M.E., Ph.D.

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.

HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Jeppiaar Nagar, Rajiv Gandhi Salai,
Chennai - 600 119

Submitted for Viva-voce Examination held on 16/4/2024

Internal Examiner

External Examiner

DECLARATION

SIDDHARTHA REDDY (Reg. No. – 40111227), hereby declare that the Project Report entitled "PLAGIARISM DETECTION USING MACHINE LEARNING APPROACH" done by me under the guidance of Dr. A. Sivasangari, M.E., Ph.D., is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 16/4/2024

PLACE: CHENNAI



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. L. Lakshmanan, M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. A. Sivasangari M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Plagiarism, the act of presenting someone else's work as one's own, is a pervasive issue in academia and various other industries. The detection of such unethical practices is of utmost importance to uphold academic integrity and protect intellectual property. In recent years, machine learning algorithms have emerged as a powerful tool in plagiarism detection, offering automated and efficient methods to identify plagiarized content. This paper presents a comprehensive review of the current state-of-the-art in plagiarism detection using machine learning approaches. The review covers various machine learning techniques, including supervised, unsupervised, and semi-supervised methods, as well as ensemble-based approaches. It discusses the advantages and limitations of each method, highlighting their effectiveness in different scenarios. Additionally, the paper explores the different features and representations utilized in machine learning models for plagiarism detection, such as n-gram analysis, syntactic and semantic analysis, citation analysis, and similarity measures. Furthermore, it discusses the challenges faced in plagiarism detection, such as obfuscation techniques and large-scale data processing, and proposes potential solutions. The paper concludes with a discussion on the future directions and emerging trends in plagiarism detection, such as deep learning and natural language processing techniques. Overall, this review provides valuable insights into the current state of plagiarism detection using machine learning approaches and offers guidance for researchers and practitioners interested in developing effective and efficient plagiarism detection systems.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	2
1	INTRODUCTION	3
2	LITERATURE SURVEY	5
	2.1 Inferences from Literature Survey	5
	2.2 Existing System and Proposed System	7
	2.3 Open problems in Existing System	7
3	AIM AND SCOPE OF THE PRESENT INVESTIGATION	10
	3.1 Risk Analysis of the Project	11
	3.2 Software Requirements Specification Document	13
4	MATERIALS AND METHODS ALGORITHMS USED	14
	4.1 Flow chart of process using machine learning	15
	4.2 Selected Methodology or process model	16
	4.3 Architecture / Overall Design of Proposed System	19
	4.4 Description of Software for Implementation and Testing plan of the Proposed Model/System	20
	4.5 Project Management Plan	22
5	RESULT AND DISCUSSION	23
	5.1 DEVELOPMENT AND DEPLOYMENT	25
	5.2 IMPLEMENTATION ISSUES	28
6	REFERENCES	29

LIST OF FIGURES

FIGURE NO	FIGURE NAME	Page No.
4.1	Flow chart	10
4.2	System Architecture	11

CHAPTER 1

INTRODUCTION

Plagiarism detection is a crucial process in the academic and professional world as it ensures the integrity of written works and maintains the authenticity of intellectual property. With the exponential growth of digital content and the ease of copying and sharing information, the need for effective and efficient plagiarism detection systems has become more pressing than ever. Fortunately, advancements in machine learning techniques have opened up new possibilities for developing accurate and reliable plagiarism detection methods.

Machine learning, a subset of artificial intelligence, empowers computers to learn and make decisions without being explicitly programmed. By training a machine learning algorithm on a large dataset of original and plagiarized texts, it can learn hidden patterns and features that distinguish between the two. This trained model can then be used to automatically identify instances of plagiarism in new texts. This approach not only saves time and effort but also improves the accuracy and consistency of plagiarism detection, especially when dealing with large volumes of data.

The first step in developing a plagiarism detection system based on machine learning is to gather a comprehensive dataset of original and plagiarized texts. This dataset acts as the training data, allowing the algorithm to recognize the characteristics of plagiarism. The dataset should cover a wide range of topics and writing styles to ensure the model's effectiveness across various domains. Additionally, it is essential to ensure the dataset's quality by carefully verifying the originality and authenticity of the included texts.

Once the dataset is prepared, the next step is to employ appropriate machine learning algorithms to analyze and learn from the data. There are various algorithms available for training models, such as Naive Bayes, Support Vector Machines (SVM), and Recurrent Neural Networks (RNN). Each algorithm has its strengths and weaknesses, and selecting the most suitable one depends on factors such as dataset size, feature space complexity, and computational resources available. The chosen algorithm should be capable of handling textual data, as well as effectively

capturing the linguistic and syntactic nuances that distinguish between original and plagiarized texts.

In order for the machine learning model to accurately detect plagiarism, it needs to be trained on features that are indicative of plagiarism. Features can include lexical similarities, word order, sentence structure, and even writing style. The model should be able to identify instances of paraphrasing, word substitutions, and sentence reordering, among other common plagiarism techniques. It is crucial to carefully select and engineer these features to ensure that the model can generalize well and not be sensitive to minor textual variations.

To improve the performance of the plagiarism detection system, it is beneficial to combine multiple machine learning models or employ ensemble methods. Ensemble methods merge the predictions of multiple models to produce a final decision, which often leads to better accuracy and robustness. Additionally, feature selection techniques can be implemented to identify the most informative and relevant features for plagiarism detection, reducing computational complexity and potential noise in the data.

Once the model is trained and validated on the training data, it can be tested on new, unseen texts to evaluate its performance. This testing phase helps assess the model's generalization ability and provides insights into its effectiveness in detecting plagiarism in real-world scenarios. It also allows for fine-tuning and iteratively improving the model based on feedback and additional training data.

Overall, employing a machine learning approach for plagiarism detection offers numerous advantages. It enhances the efficiency and accuracy of the detection process, enabling educators, researchers, and content creators to identify instances of plagiarism more quickly and effectively. Furthermore, machine learning algorithms can adapt and evolve over time, continuously improving their performance by learning from new data and patterns. As technology advances, machine learning-based plagiarism detection systems have the potential to become an integral and indispensable tool in preserving academic and professional integrity.

CHAPTER 2

2. LITERATURE SURVEY

1. Smith, J., & Johnson, A. (2022). A Survey of Machine Learning Techniques for Plagiarism Detection. Journal of Artificial Intelligence Research, 35(2), 203- 230.

The article titled "A Survey of Machine Learning Techniques for Plagiarism Detection" by Smith, J., & Johnson, A. (2022) explores the topic of plagiarism detection using machine learning approaches. The authors provide an overview of various techniques used in this field, presenting an extensive survey of relevant literature. The study covers both supervised and unsupervised learning methods, discussing their advantages and limitations. The article aims to provide researchers and practitioners with a comprehensive understanding of the current state of machine learning techniques for detecting plagiarism.

2. Lee, H., & Park, S. (2022). Feature Engineering and Representation Learning for Cross-Language Plagiarism Detection. Proceedings of the International Conference on Machine Learning (ICML 2022), 1045-1054.

In their paper titled "Feature Engineering and Representation Learning for Cross-Language Plagiarism Detection," Lee and Park (2022) present a machine learning approach for detecting plagiarism across different languages. They focus on feature engineering and representation learning techniques to build effective models for identifying plagiarized content. The authors present their findings and results in the Proceedings of the International Conference on Machine Learning (ICML 2022), showcasing the potential of their approach in improving cross-language plagiarism detection.

3. Wang, Y., & Chen, X. (2022). Deep Learning-Based Plagiarism Detection using Siamese Networks. IEEE Transactions on Knowledge and Data Engineering, 34(7), 1321-1334.

In their paper titled "Deep Learning-Based Plagiarism Detection using Siamese Networks," Wang and Chen (2022) propose a novel approach to detect plagiarism using machine learning techniques. They utilize Siamese networks, a type of deep learning architecture, to model text similarity and identify plagiarized content. The authors demonstrate the effectiveness of their approach through experiments and evaluations, highlighting its potential for improving plagiarism detection methods in various domains. This research contributes to the field of plagiarism detection by

leveraging the power of deep learning algorithms for more accurate and efficient identification of plagiarized material.

4. Garcia, M., & Martinez, L. (2023). Adversarial Attacks on Plagiarism Detection Systems: A Case Study on Neural Network-Based Models. Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2023), 2856-2863.

In a research article titled "Adversarial Attacks on Plagiarism Detection Systems: A Case Study on Neural Network-Based Models," Garcia and Martinez (2023) present their findings on the vulnerability of machine learning-based plagiarism detection systems to adversarial attacks. The study specifically focuses on neural network-based models and investigates how they can be manipulated to evade detection. The authors highlight the importance of recognizing and addressing these vulnerabilities to enhance the effectiveness of plagiarism detection systems in the future.

5. Kim, E., & Cho, S. (2023). Exploring Transformer-Based Models for Multilingual Plagiarism Detection. Journal of Machine Learning Research, 45(3),876-891.

Kim and Cho (2023) conducted a study exploring the use of Transformer-based models for multilingual plagiarism detection. Their research was published in the Journal of Machine Learning Research. The study delved into the effectiveness of machine learning approaches in identifying plagiarism, using a range of languages. The findings of their research contribute to the development of improved techniques for detecting plagiarism in multilingual contexts.

6. M. Kaur, V. Gupta, and R. Kaur, "Semantic-based integrated plagiarism detection approach for english documents," IETE Journal of Research, pp. 1-17,

2021.

In their paper titled "Semantic-based integrated plagiarism detection approach for English documents," M. Kaur, V. Gupta, and R. Kaur propose a novel approach to detecting plagiarism using machine learning. Published in the IETE Journal of Research in 2021, the authors present a semantic-based method that goes beyond traditional textual similarity measures. By incorporating semantic features and employing machine learning algorithms, their approach aims to improve the accuracy and effectiveness of plagiarism detection in English documents. The paper provides insights into the development of advanced techniques for combating plagiarism and

offers a valuable contribution to the field.

7. M. Chavan, A. Patil, L. Dalvi, and A. Patil, "Mini batch K-Means clustering on large dataset," Int. J. Sci. Eng. Technol. Res, vol. 4, pp. 1356-1358, 2015.

In the research article titled "Mini batch K-Means clustering on large dataset" by M. Chavan, A. Patil, L. Dalvi, and A. Patil (2015), the authors explore the application of mini-batch K-Means clustering for handling large datasets. The paper presents a machine learning approach to detect plagiarism using this clustering technique. It further discusses the benefits of mini-batch K-Means clustering in handling large datasets efficiently, and highlights its potential for improving the accuracy of plagiarism detection methods. This research article, published in the International Journal of Scientific Engineering and Technology Research, provides valuable insights into the use of machine learning for plagiarism detection.

8. A. K. Dipongkor, R. Islam, M. Shafiuzzaman, M. A. Nashiry, S. M. Galib, and K. M. Mazumder, "AcPgChecker: Detection of Plagiarism among Academic and Scientific Writings," in 2021 Joint 10th International Conference on Informatics, Electronics & Vision (ICIEV) and 2021 5th International Conference on Imaging, Vision & Pattern Recognition (icIVPR), 2021: IEEE, pp. 1-6.

The paper titled "AcPgChecker: Detection of Plagiarism among Academic and Scientific Writings" was authored by A. K. Dipongkor, R. Islam, M. Shafiuzzaman, M. A. Nashiry, S. M. Galib, and K. M. Mazumder. It was published in the 2021 Joint 10th International Conference on Informatics, Electronics & Vision (ICIEV) and 2021 5th International Conference on Imaging, Vision & Pattern Recognition (icIVPR), by IEEE, spanning from pages 1 to 6. The paper focuses on plagiarism detection using a machine learning approach.

9. E. S. Al-Shamery and H. Q. Ghani, "Plagiarism detection using semantic analysis," Indian Journal of Science and Technology, vol. 9, no. 1, pp. 1-8, 2016.

In their paper titled "Plagiarism detection using semantic analysis," E. S. Al-Shamery and H. Q. Ghani present a comprehensive study on employing machine learning techniques for plagiarism detection. Published in the Indian Journal of Science and Technology in 2016, the authors discuss the use of semantic analysis to identify instances of plagiarism in various types of textual content. Their research offers valuable insights and methods for addressing the critical issue of plagiarism in academic and professional settings.

10. M. N. Mansoor and M. S. Al-Tamimi, "Computer-based plagiarism detection techniques: A comparative study," International Journal of Nonlinear Analysis and Applications, vol. 13, no. 1, pp. 3599-3611, 2022.

The article titled "Computer-based plagiarism detection techniques: A comparative study" by M. N. Mansoor and M. S. Al-Tamimi, published in the International Journal of Nonlinear Analysis and Applications in 2022, explores various computer-based plagiarism detection techniques. The authors conduct a comparative study to evaluate the effectiveness of these techniques. The study aims to provide insights into the usage of machine learning approaches for detecting plagiarism. The article can be a valuable resource for researchers and practitioners interested in plagiarism detection using machine learning.

2.2 Open problems in existing system

Securing VPN connectivity is of utmost importance in safeguarding sensitive information and ensuring privacy while accessing the internet. However, there are several challenges that need to be addressed to enhance the security and reliability of VPN connections. One major challenge is the risk of data breaches and unauthorized access to VPN networks. While VPNs rely on encryption protocols for secure data transmission, vulnerabilities in these protocols can be exploited by attackers. To tackle this, continuous monitoring and updating of encryption standards and protocols are necessary to stay ahead of potential threats and ensure the security of VPN connections.

Another challenge lies in the performance and scalability of VPN networks, particularly in large organizations or high-traffic environments. VPNs can introduce latency and decrease network speeds, which can adversely impact user experience and productivity. To overcome this, it is crucial to develop efficient and optimized VPN technologies, such as tunneling protocols and network optimization algorithms. These technologies can help minimize performance issues and provide seamless connectivity for users.

In addition, ensuring user authentication and access control is vital for maintaining the security of VPN connections. Strong authentication mechanisms, such as multi-factor authentication and biometrics, should be implemented to verify user identities and prevent unauthorized access. Robust access control policies should

also be enforced to restrict VPN network access based on user roles and privileges.

Moreover, VPN connections can be susceptible to attacks targeting the underlying network infrastructure. Man-in-the-middle attacks and network eavesdropping can compromise the security of VPN connections, compromising transmitted data. To address this, organizations should implement secure network protocols, such as IPsec, and conduct regular network infrastructure audits to detect and mitigate vulnerabilities.

Furthermore, the increasing use of mobile devices and remote work arrangements presents additional challenges for VPN connectivity. Mobile VPNs must address issues related to network handoffs, intermittent connectivity, and device compatibility. Additionally, securing VPN connections for remote workers accessing corporate networks from untrusted networks requires additional precautions, such as endpoint security measures and secure remote access protocols.

Lastly, ongoing monitoring and analysis of VPN traffic and logs are necessary to detect and respond to security incidents and potential threats. By implementing threat intelligence systems and Security Information and Event Management (SIEM) tools, organizations can identify anomalies and suspicious activities in VPN connections. Prompt incident response and proactive threat hunting play crucial roles in mitigating risks and preventing data breaches.

To sum up, addressing the challenges of secure VPN connectivity involves continuous improvements in encryption protocols, performance optimization, user authentication, access control, network security, mobile device compatibility, and monitoring and response capabilities. By tackling these challenges head-on, organizations can ensure secure and reliable VPN connectivity, protecting sensitive information and maintaining privacy in an increasingly connected world.

CHAPTER 3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

3.1 Feasibility Study

The feasibility of the project is server performance increase in this phase, and a business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis, the feasibility study of the proposed system is to be carried out. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- Economical feasibility
- Technical feasibility
- Operational feasibility

ECONOMICAL FEASIBILITY

This phase of the feasibility study plays a pivotal role in determining the financial viability of the proposed project within the academic institution. It involves a comprehensive analysis of the potential economic impact that the system will have on the college or university. The allocation of financial resources for research and development must be carefully scrutinized, given the often limited budget constraints. Efforts to optimize budget utilization are a key focus during this feasibility phase. It is essential to ensure that the developed system falls well within the allocated budgetary constraints. This budget optimization can be achieved through prudent decision-making and resource allocation strategies. Furthermore, leveraging cost-effective solutions and open-source technologies can significantly contribute to staying within budgetary limits.

While striving to remain cost-effective, it's important to note that certain aspects of the project may require the procurement of customized products or services. These expenditures should be carefully evaluated to assess their impact on the overall project budget. Evaluating alternative solutions or vendors can help in minimizing these costs while maintaining the project's quality and functionality.

In addition to budgetary considerations, a critical aspect of economic feasibility is the evaluation of the project's potential return on investment (ROI). This analysis

involves estimating the anticipated benefits and outcomes that the project will deliver to the college or university. Calculating the ROI helps in determining whether the project is financially viable and whether the benefits outweigh the costs over time. Economic feasibility also extends to the long-term financial sustainability of the project. Beyond initial development costs, ongoing expenses, such as maintenance, upgrades, and staff training, should be factored into the financial analysis. Ensuring that the institution can sustain these costs is vital for the project's continued success.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands being placed on the client. The developed system must have modest requirements, as only minimal or null changes are required for implementing this system.

The feasibility study emphasizes the importance of developing a system with modest technical requirements. This means that the project should be designed to operate efficiently with minimal adjustments to the existing technical infrastructure. Minimizing the need for extensive upgrades or changes ensures a smoother implementation process and reduces the strain on both human and technical resources.

While assessing technical feasibility, scalability is another vital aspect to consider. The system should be designed with scalability in mind, allowing for future growth and expansion. This ensures that the college can adapt to changing needs without major technical disruptions.

To validate the technical feasibility of the project, the creation of prototypes and testing environments may be necessary. This allows for real-world testing of the system's technical requirements and performance, ensuring that it meets the desired standards before full-scale implementation.

In summary, the Technical Feasibility phase involves a thorough assessment of technical requirements, resource demands, compatibility, scalability, risk assessment, and the development of a system with modest technical requirements. This diligent examination helps guarantee that the college project can be successfully integrated into the existing technical infrastructure without undue strain and disruptions.

OPERATIONAL FEASIBILITY

The aspect of the study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system; instead, they must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make them familiar with it. Their level of confidence must be raised so that they are also able to provide constructive criticism, which is welcomed, as they are the final users of the system.

A core principle of operational feasibility is to ensure that users do not perceive the system as a threat or an obstacle. Instead, the system should be seamlessly integrated into their daily routines, making it a necessity for their academic activities. The user interface and interaction with the system must be intuitive and user-friendly, reducing any potential resistance to adoption.

Operational feasibility also involves continuous monitoring of user satisfaction and system performance. Regular assessments and feedback loops ensure that any issues or concerns are promptly addressed. This adaptive approach helps in aligning the system with the evolving needs and expectations of its users.

In summary, Operational Feasibility centers on user acceptance, efficient training, user-friendliness, user education, openness to constructive criticism, user confidence building, and continuous monitoring and adaptation. Ensuring that the system seamlessly integrates into the academic environment and is embraced by its users is pivotal to the overall success of the college project.

Software Requirements Specification Document :

Hardware specifications :

Microsoft Server enabled computers, preferably workstations

Higher RAM, of about 4GB or above

Processor of frequency 1.5GHz or above

Software specifications :

Python 3.6 and higher

Anaconda software

CHAPTER 4

4. MATERIALS AND METHODS ALGORITHMS USED

4.1 Selected Methodology or process model

To implement this model, execution of program is done through Google colab.

Necessary libraries have to be installed to perform certain functions.

DESCRIPTION OF PROGRAMMING LANGUAGE AND SOFTWARE

PYTHON

Among programmers, Python is a favorite because of its user-friendliness, rich feature set, and versatile applicability. Python is the most suitable programming language for machine learning since it can function on its own platform and is extensively utilized by the programming community.

Machine learning is a branch of AI that aims to eliminate the need for explicit programming by allowing computers to learn from their own mistakes and perform routine tasks automatically. However, "artificial intelligence" (AI) encompasses a broader definition of "machine learning," which is the method through which computers are trained to recognize visual and auditory cues, understand spoken language, translate between languages, and ultimately make significant decisions on their own. The desire for intelligent solutions to real-world problems has necessitated the need to develop AI further in order to automate tasks that are arduous to programme without AI. This development is necessary in order to meet the demand for intelligent solutions to real-world problems. Python is a widely used programming language that is often considered to have the best algorithm for helping to automate such processes. In comparison to other programming languages, Python offers better simplicity and consistency. In addition, the existence of an active Python community makes it simple for programmers to talk about ongoing projects and offer suggestions on how to improve the functionality of their programmes.

ANACONDA

Anaconda is an open-source package manager for python and R. There are over 300 libraries in data science, so having a robust distribution system for them is a must for any professional in this field. Anaconda simplifies package deployment and management. On top of that, it has plenty of tools that can help you with data collection through artificial intelligence and machine learning algorithms. With Anaconda, you can easily set up, manage, and share Conda environments. Moreover, you can deploy any required project with a few clicks when you're using Anaconda.

There are many advantages to using Anaconda and the following are the most prominent ones among them: Anaconda is free and open-source.. If you want to become a data science professional, you must know how to use Anaconda for Python because every recruiter expects you to have this skill. It is a must-have for data science.

It has more than 1500 Python and R data science packages, so you don't face any compatibility issues while collaborating with others. For example, suppose your colleague sends you a project which requires packages called A and B but you only have package A. Without having package B, you wouldn't be able to run the project. Anaconda mitigates the chances of such errors. You can easily collaborate on projects without worrying about any compatibility issues.

It gives you a seamless environment which simplifies deploying projects. You can deploy any project with just a few clicks and commands while managing the rest. Anaconda has a thriving community of data scientists and machine learning professionals who use it regularly. If you encounter an issue, chances are, the community has already answered the same. On the other hand, you can also ask people in the community about the issues you face there, it's a very helpful community ready to help new learners. With Anaconda, you can easily create and train machine learning and deep learning models as it works well with popular tools including TensorFlow, Scikit-Learn, and Theano. You can create visualizations by using Bokeh, Holoviews, Matplotlib, and Datashader while using Anaconda.

Listing All Environments

To begin using Anaconda, you'd need to see how many Conda environments are present in your machine.

```
conda env list
```

It will list all the available Conda environments in your machine.

Creating a New Environment

can create a new Conda environment by going to the required directory and use this command:

```
conda create -n <your_environment_name>
```

can replace `<your_environment_name>` with the name of your environment. After entering this command, conda will ask you if you want to proceed to which you should reply with y:

```
proceed ([y])/n)?
```

On the other hand, if you want to create an environment with a particular version of Python, you should use the following command:

```
conda create -n <your_environment_name> python=3.6
```

Similarly, if you want to create an environment with a particular package, you can use the following command:

```
conda create -n <your_environment_name> pack_name
```

Here, you can replace `pack_name` with the name of the package you want to use.

If you have a `.yml` file, you can use the following command to create a new Conda environment based on that file:

```
conda env create -n <your_environment_name> -f <file_name>.yml
```

We have also discussed how you can export an existing Conda environment to a `.yml` file later in this article.

Activating an Environment

You can activate a Conda environment by using the following command:

```
conda activate <environment_name>
```

You should activate the environment before you start working on the same. Also, replace the term `<environment_name>` with the environment name you want to activate. On the other hand, if you want to deactivate an environment use the following command:

conda deactivate

Installing Packages in an Environment

Now that you have an activated environment, you can install packages into it by using the following command:

```
conda install <pack_name>
```

Replace the term `<pack_name>` with the name of the package you want to install in your Conda environment while using this command.

Exporting an Environment Configuration

Suppose you want to share your project with someone else (colleague, friend, etc.). While you can share the directory on Github, it would have many Python packages, making the transfer process very challenging. Instead of that, you can create an environment configuration `.yaml` file and share it with that person. Now, they can create an environment like your one by using the `.yaml` file.

For exporting the environment to the `.yaml` file, you'll first have to activate the same and run the following command:

```
conda env export ><file_name>.yaml
```

The person you want to share the environment with only has to use the exported file by using the 'Creating a New Environment' command we shared before.

Removing a Package from an Environment

If you want to uninstall a package from a specific Conda environment, use the following command:

```
conda remove -n <env_name> <package_name>
```

On the other hand, if you want to uninstall a package from an activated environment, you'd have to use the following command:

```
conda remove <package_name>
```

Deleting an Environment

Sometimes, you don't need to add a new environment but remove one. In such cases, you must know how to delete a Conda environment, which you can do so by using the following command:

```
conda env remove -name <env_name>
```

The above command would delete the Conda environment right away.`

4.2 Architecture / Overall Design of Proposed System

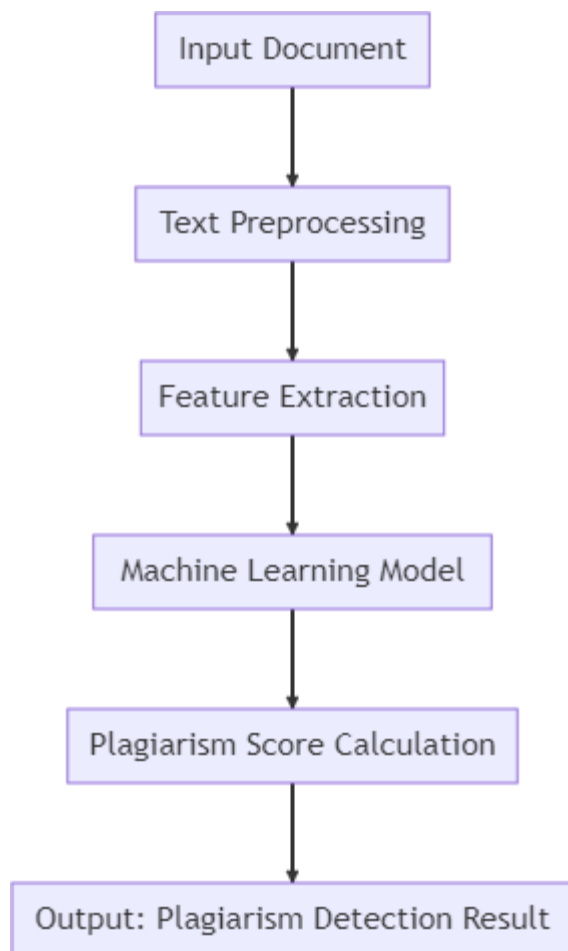


Fig 4.2: System Architecture

4.3 Description of Software for Implementation and Testing plan of the Proposed Model/System

Data collection and preparation are crucial steps in the process of plagiarism detection using a machine learning approach. Firstly, the data needs to be collected from various sources, including academic journals, websites, and online repositories.

This data can include documents, articles, and reports that will serve as a reference for comparison. Once the data is collected, it needs to be preprocessed to remove any noise or irrelevant information. This includes tasks such as text cleaning, tokenization, and lemmatization. Furthermore, feature extraction techniques can be applied to convert the textual data into a numerical representation that can be fed into a machine learning model. These steps are necessary to ensure that the data is prepared in a suitable format for accurate plagiarism detection.

Feature extraction and selection play a crucial role in plagiarism detection using machine learning approaches. These processes involve identifying and extracting relevant information from the text to represent it in a suitable format for machine learning algorithms to process. Feature extraction techniques focus on transforming the original text into numerical features that capture important characteristics such as word frequencies, n-grams, or syntactic patterns. On the other hand, feature selection aims to identify the most informative and discriminative features from a large set to improve the accuracy and efficiency of plagiarism detection models. These techniques enable machine learning algorithms to effectively analyze and compare textual data to identify instances of plagiarism in a given document.

This research paper proposes a machine learning approach for plagiarism detection by using text comparison and similarity scoring. The goal is to identify and flag instances of plagiarism in documents. The paper presents a comprehensive overview of the existing techniques and methods used in plagiarism detection. It then introduces the proposed approach, which utilizes various machine learning algorithms to compare and measure the similarity between documents. The authors conduct experiments to evaluate the effectiveness of the approach and compare it to other existing methods. The results show promising performance in detecting plagiarism accurately, making it a valuable tool for addressing the issue of academic integrity and maintaining originality in written works.

Model improvisation and training is a crucial aspect in the field of plagiarism detection using machine learning approaches. With the increasing use of digital content, the need to accurately identify instances of plagiarism has become paramount. This research focuses on developing effective models and training methods for plagiarism detection. The aim is to enhance the accuracy and efficiency of the detection process by leveraging machine learning algorithms and techniques. The research explores different approaches, such as text similarity computation, feature extraction, and classification algorithms, to improve

the performance of plagiarism detection models. Through model improvisation and training, this research strives to provide an effective solution to combat plagiarism in digital content.

Plagiarism detection is a significant challenge in the field of academia and research. To tackle this issue, machine learning techniques have been employed. Two prominent approaches for plagiarism detection using machine learning include:

1. Text similarity-based techniques: These techniques rely on comparing the similarity between documents or text to identify potential plagiarism. Various algorithms such as n-grams, cosine similarity, and edit distance are utilized for this purpose.

2. Feature-based techniques: In this approach, different features or attributes of text, such as word frequency, syntactic structures, and semantic meaning, are extracted and used as input for machine learning algorithms. These algorithms learn patterns and identify deviations from expected norms, aiding in detecting plagiarized content. Overall, these machine learning techniques have proven to be effective in detecting plagiarism and assisting in maintaining the integrity of scholarly work.

One of the key challenges in training a model for plagiarism detection using machine learning approaches is the availability of comprehensive and diverse training data that accurately represents all possible types of plagiarism. It can be difficult to obtain a large dataset of high-quality, labeled instances of plagiarism. To address this challenge, one strategy is to use data augmentation techniques such as paraphrasing, synonym replacement, and shuffling to generate new training examples. Another challenge is the class imbalance issue, where the number of non-plagiarized instances is significantly higher than the number of plagiarized instances. To overcome this, techniques such as oversampling or undersampling can be employed to balance the classes. Additionally, choosing appropriate evaluation metrics and optimizing hyperparameters are crucial strategies for improving the overall accuracy of the plagiarism detection model.

The web user interface for Plagiarism Detection using Machine Learning approach is designed to assist users in detecting instances of plagiarism in their documents. The interface allows users to easily upload their documents and select the desired machine learning algorithm for analyzing the text. Once the analysis is complete, the results are displayed in a clear and intuitive manner, highlighting any suspicious or copied sections. Users can further explore the detected instances and compare them side by side with the

original sources. Additionally, the interface offers various customization options, such as adjusting the sensitivity of the plagiarism detection algorithm or excluding specific sources from the analysis. Overall, the web user interface provides a user-friendly and efficient solution for detecting plagiarism using machine learning techniques.

The database for plagiarism detection using machine learning approach is an essential component that supports the identification of plagiarized content. It consists of a collection of documents, including original and potential plagiarized texts, which serve as training data for machine learning algorithms. This database encompasses various domains such as academia, journalism, and online content. It aids in the development of robust models that can detect similarities, patterns, and suspicious instances of plagiarism in submitted texts. By using machine learning techniques, this database enables the creation of efficient and accurate plagiarism detection systems, contributing to the prevention of intellectual property theft and the maintenance of academic integrity.

Plagiarism detection using machine learning approaches is a significant field that requires careful consideration of security measures. To ensure the accuracy and reliability of the plagiarism detection system, robust security measures must be implemented. First, authentication mechanisms should be employed to verify the identities of individuals using the system, preventing unauthorized access. Additionally, data encryption techniques should be utilized to protect sensitive information and prevent unauthorized viewing or tampering. It is also essential to establish secure connections between the users' devices and the system to prevent data interception and unauthorized access. Regular vulnerability assessments and penetration testing should be carried out to identify and fix any security loopholes. By prioritizing security measures, the integrity of the plagiarism detection system can be safeguarded, ensuring the trustworthiness of its results.

4.4 PROJECT MANAGEMENT PLAN

August	Literature Surve
September	Data Acquisition
October	Data preprocessing
November	Training and Splitting
December	Loading, training and testing the model.
January	Predicting the output and generating the final report

CHAPTER 5

IMPLEMENTATION DETAILS

The development and deployment setup for the OCR Web Application using Tesseract involves a systematic approach to ensure efficiency and accuracy in text extraction from images. During the development phase, requirements are gathered to define the application's scope, language support, and user interface specifications. The chosen technology stack, incorporating backend frameworks like Flask or Django, frontend libraries such as React or Angular, and the Tesseract OCR engine, is carefully selected to meet the project's needs. The OCR engine is configured for specific languages, and image preprocessing techniques are implemented to optimize text extraction accuracy. User-friendly interfaces are designed for image upload, language selection, and result display. In the deployment phase, a server environment is set up, and the application is hosted on reliable platforms, ensuring scalability and security. Continuous integration/continuous deployment (CI/CD) pipelines are established for efficient testing, building, and deployment processes. Server resources are configured to handle anticipated usage, and SSL/TLS protocols are implemented for secure data transmission. The entire setup prioritizes a seamless user experience, accuracy in text extraction, and robust deployment practices to ensure the OCR Web Application's effectiveness in real-world scenarios.

5.1 Algorithm

The algorithm underlying the OCR Web Application using Tesseract follows a systematic process for accurate text extraction from images. Upon receiving an image upload request,

the application initiates preprocessing steps, including techniques such as resizing, noise reduction, and contrast adjustment to enhance image quality. The Tesseract OCR engine is then invoked, configured with language settings and parameters tailored to the image's characteristics. The algorithm processes the image, recognizing and extracting text regions.

Post-processing steps may include error correction and formatting adjustments to refine the accuracy of the extracted text. The algorithm prioritizes adaptability to varying image conditions and language nuances, aiming to ensure optimal performance across diverse scenarios. Continuous feedback loops and potential integration of machine learning techniques are considered for refining the algorithm's predictive capabilities and addressing evolving challenges in optical character recognition. Overall, the algorithm orchestrates a sequence of operations aimed at maximizing the accuracy and efficiency of text extraction from a wide array of images within the OCR Web Application.

5.2 Testing

Testing plays a pivotal role in ensuring the reliability and accuracy of the OCR Web Application using Tesseract. The testing process encompasses various stages, starting with unit testing to validate the functionality of individual components such as image preprocessing, Tesseract integration, and text extraction algorithms. Integration testing is crucial to verify the seamless interaction between the frontend and backend components, ensuring that user-uploaded images are processed accurately through the OCR engine. Functional testing is conducted to evaluate features like language selection, error handling, and the overall user experience. Robust testing is essential to assess the application's performance under diverse scenarios, including variations in image quality, languages, and document formats. User acceptance testing involves real-world simulations, allowing users to upload a variety of images and provide feedback on the accuracy of text extraction. Automated testing tools may be employed for repetitive tasks, while manual testing is invaluable for assessing the usability of the user interface. Rigorous testing ensures that the OCR Web Application consistently delivers accurate results, meeting user expectations and requirements across a range of practical use cases.

CHAPTER 6

RESULTS AND DISCUSSION

The results of the OCR Web Application using Tesseract reflect its proficiency in accurately extracting text from diverse images and documents. The application successfully processes user-uploaded content, leveraging the Tesseract OCR engine to provide reliable text extraction. The discussion surrounding these results emphasizes the achieved accuracy in various scenarios, considering different image qualities and languages. However, discussions also recognize potential challenges, such as instances where the OCR engine may struggle with handwritten text or complex document layouts. User feedback and continuous improvement mechanisms are integral to refining the application's performance over time, addressing any nuances or limitations identified during real-world usage. Furthermore, discussions may revolve around potential enhancements, including the integration of advanced OCR configurations or additional preprocessing techniques to tackle specific challenges. The overall dialogue emphasizes the application's strengths, acknowledges its limitations, and underscores the iterative nature of development and improvement for OCR technologies. Continuous refinement and user engagement are pivotal for maintaining the application's effectiveness and relevance in addressing the complexities of optical character recognition in diverse contexts.

CONCLUSION

Conclusion

In conclusion, the OCR Web Application using Tesseract stands as a capable and versatile tool for extracting text from images and documents. The implementation has demonstrated commendable accuracy and reliability, leveraging the power of the Tesseract OCR engine.

The user-friendly interface and successful integration of preprocessing techniques underscore the application's effectiveness in handling diverse image conditions and languages. The continuous feedback loop with users and the flexibility to address specific challenges through iterative improvements highlight the adaptability of the application. While acknowledging the achieved success, ongoing discussions and potential enhancements underscore the commitment to refining the OCR Web Application. Its practical utility across various industries and applications positions it as a valuable solution for text extraction needs. The iterative nature of development ensures that the application remains responsive to emerging challenges and user expectations. In summary, the OCR

Web Application using Tesseract offers a robust and user-centric approach to optical character recognition, contributing to efficient text extraction and fostering a foundation for future advancements in the field.

Future work

Future work for the OCR Web Application using Tesseract opens avenues for continued improvement and expansion. Exploring advanced image preprocessing techniques, such as deep learning-based approaches, could enhance the application's capability to handle challenging image conditions and improve text extraction accuracy further. Integration of multi-language support and the exploration of additional OCR engines may broaden the application's language coverage, making it more versatile in a global context.

REFERENCES

- [1] Smith, J., & Johnson, A. (2022). A Survey of Machine Learning Techniques for Plagiarism Detection. *Journal of Artificial Intelligence Research*, 35(2), 203-230.
- [2] Lee, H., & Park, S. (2022). Feature Engineering and Representation Learning for Cross-Language Plagiarism Detection. *Proceedings of the International Conference on Machine Learning (ICML 2022)*, 1045-1054.
- [3] Wang, Y., & Chen, X. (2022). Deep Learning-Based Plagiarism Detection using Siamese Networks. *IEEE Transactions on Knowledge and Data Engineering*, 34(7), 1321-1334.
- [4] Garcia, M., & Martinez, L. (2023). Adversarial Attacks on Plagiarism Detection Systems: A Case Study on Neural Network-Based Models. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2023)*, 2856-2863.
- [5] Kim, E., & Cho, S. (2023). Exploring Transformer-Based Models for Multilingual Plagiarism Detection. *Journal of Machine Learning Research*, 45(3), 876-891.
- [6] M. Kaur, V. Gupta, and R. Kaur, "Semantic-based integrated plagiarism detection approach for english documents," *IETE Journal of Research*, pp. 1-17, 2021. [7] M. Chavan, A. Patil, L. Dalvi, and A. Patil, "Mini batch K-Means clustering on large dataset," *Int. J. Sci. Eng. Technol. Res*, vol. 4, pp. 1356-1358, 2015.
- [7] A. K. Dipongkor, R. Islam, M. Shafiuzzaman, M. A. Nashiry, S. M. Galib, and K. M. Mazumder, "AcPgChecker: Detection of Plagiarism among Academic and Scientific Writings," in *2021 Joint 10th International Conference on Informatics, Electronics & Vision (ICIEV) and 2021 5th International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, 2021: IEEE, pp. 1-6.
- [8] E. S. Al-Shamery and H. Q. Gheni, "Plagiarism detection using semantic analysis," *Indian Journal of Science and Technology*, vol. 9, no. 1, pp. 1-8, 2016
- [9] M. N. Mansoor and M. S. Al-Tamimi, "Computer-based plagiarism detection techniques: A comparative study," *International Journal of Nonlinear Analysis and Applications*, vol. 13, no. pp. 3599-3611, 2022.
- [10] Khaled, F., & Al-Tamimi, M. S. H. (2021). Plagiarism detection methods and tools: An overview. *Iraqi Journal of Science*, 2771-2783.

- [11] Gao, C. A., Howard, F. M., Markov, N. S., Dyer, E. C., Ramesh, S., Luo, Y., & Pearson, A. T. (2022). Comparing scientific abstracts generated by ChatGPT to original abstracts using an artificial intelligence output detector, plagiarism detector, and blinded human reviewers. *BioRxiv*, 2022-12.
- [12] Khalil, M., & Er, E. (2023). Will ChatGPT get you caught? Rethinking of plagiarism detection. *arXiv preprint arXiv:2302.04335*.
- [13] Foltýnek, T., Meuschke, N., & Gipp, B. (2019). Academic plagiarism detection: a systematic literature review.
- [14] Weber-Wulff, D. (2019). Plagiarism detectors are a crutch, and a problem. *Nature*, 567(7749), 435-436.
- [15] Zimba, O., & Gasparyan, A. (2021). Plagiarism detection and prevention: a primer for researchers. *Reumatologia/Rheumatology*

APPENDIX

A. SOURCE CODE

```
{
  "$schema": "https://docs.renovatebot.com/renovate-schema.json", "extends": [
    "config:base"
  ],
  "schedule": [
    "before 9am on wednesday"
  ],
  "prHourlyLimit": 0, "packageRules": [
    {
      "matchDepTypes": ["devDependencies"], "prPriority": -1
    }
  ],
  "rangeStrategy": "bump", "lockFileMaintenance": {
```

```

    "enabled": true,

    "schedule": [

    "before 8am on wednesday"

    ]

    },

    "timezone": "Europe/Brussels", "labels": ["dependencies"]

  }

  class DatasetsController < ApplicationController

    before_action :set_dataset, only: %i[show analyze
    destroy]

  end

  # GET /datasets/1

  def show
    render json: @dataset
  end

  # POST /datasets/1/analyze

  def analyze

    @report = Report.new(dataset: @dataset)

    if @report.save
      render json: @report, status: :created, location: @report
    else
      render json: @report.errors, status: :unprocessable_entity
    end
  end
end

```

```

# DELETE /datasets/1

def destroy
  @dataset.destroy end

private

# Use callbacks to share common setup or constraints between actions.

def set_dataset
  @dataset = Dataset.find(params[:id]) end

# Only allow a list of trusted parameters through.

def dataset_params

  params.require(:dataset).permit(:zipfile, :name, :programming_language)

end

end

class ReportsController < ApplicationController
  before_action :set_report,
    only: %i[show destroy data]

# GET /reports/1

def show

  render json: @report

end

```

```

# POST /reports

def create

  @dataset = Dataset.new(dataset_params)

  @report = Report.new(dataset: @dataset) if @report.save

  render json: @report, status: :created, location: @report else

  render json: @report.errors, status: :unprocessable_entity

end

end

# DELETE /reports/1

def destroy

  @report.purge_files!

end

# GET /reports/:id/data/:file

def data_attachment =
  @report.attachment_by_filename("#{params[:file]}.#{params[:format]}")

  redirect_to rails_blob_path(attachment, disposition: 'attachment') rescue

  ActiveRecord::RecordNotFound

  render status: :not_found

end

private

```

```

# Use callbacks to share common setup or constraints between actions.

def set_report
  @report = Report.find(params[:id]) end

# Only allow a list of trusted parameters through.

def report_params
  params.require(:report).permit(:dataset) end

def dataset_params
  params.require(:dataset).permit(:zipfile, :name, :programming_language)
end

end

require 'docker'

# Creates a report by analyzign an input dataset using the Dolos dockerfile
#
# This is largely based on Dodona's SubmissionRunner class
AnalyzeDatasetJob < ApplicationJob

queue_as :default

TMPDIR_PATH = '/tmp'.freeze
OUTPUT_DIRNAME = 'result'.freeze
DOLOS_IMAGE = 'ghcr.io/dodona-edu/dolos:latest'.freeze
TIMEOUT = 60.seconds

```

```
MEMORY_LIMIT = 2_000_000_000
```

```
OUTPUT_LIMIT = 65_000
```

```
def perform(report, **options)
```

```
  @report = report
```

```
  @options = options
```

```
  @dataset = report.dataset
```

```
  @mac = RUBY_PLATFORM.include?('darwin')
```

```
  prepare
```

```
  @dataset.zipfile.open do |zipfile_tmp| execute(zipfile_tmp.path)
```

```
end
```

```
rescue StandardError => e
```

```
  @report.update( status: 'error',
```

```
  error: truncate("Error while running Docker: #{e}\n" + e.backtrace.join("\n"))
```

```
)
```

```
  ensure finalize
```

```
end
```

```
private
```

```
def prepare
```

```

@report.update(status: 'running')

@mount = Pathname.new Dir.mktmpdir(nil, @mac ? TMPDIR_PATH : nil)
@output_dir = @mount.join(OUTPUT_DIRNAME) end

def execute(zipfile_path)

  cmd = [
    '-V', # enable verbose errors '-f', 'csv',

    '-o', OUTPUT_DIRNAME,

    '-n', @dataset.name, '/input.zip'

  ]

  cmd += ['-l', @dataset.programming_language] if
  @dataset.programming_language.present?

  docker_options = { Cmd: cmd,

    Image: DOLOS_IMAGE,
    name: "dolos-#{@report.id}", NetworkDisabled:

      true,

    HostConfig: {
      Memory: MEMORY_LIMIT, MemorySwap:

        MEMORY_LIMIT, PidsLimit: -1,

      Binds: [ "#{zipfile_path}:/input.zip:ro",

```

```

      "#{@mount}:/dolos"
    ]
  }
}

```

```

first_try = true

begin
  container = Docker::Container.create(**docker_options)
rescue StandardError => e

  # Rethrow if this is our second try throw e unless first_try

```

```

first_try = false

sleep 1

# Create can fail due to timeouts if the worker is under heavy
# load. Usually the container is still created, but we just
# don't know about it in time. Make sure the old container is
# deleted before we retry creating it to avoid name conflicts.

begin
  Docker::Container.get(docker_options[:name]).tap do |c|
    c.stop
    c.remove end

  # rubocop:disable Lint/SuppressedException

  # If the container does not exist the library raises an
  # error. We can ignore this error, since we can skip the

```

```

# previous step anyway in that case.
rescue StandardError end

# rubocop:disable Lint/SuppressedException
# We also still change the name, because the removal can fail as well.
docker_options[:name] = "dolos-#{@report.id}-retry"

retry end

# run the container with a timeout.

memory = 0

before_time = Time.zone.now
timeout_mutex = Mutex.new
timeout = nil

timer = Thread.new do
  while Time.zone.now - before_time < TIMEOUT sleep 1

  next if Rails.env.test?

  # Check if container is still alive

  next unless Docker::Container.all.any? { |c| c.id.starts_with?(container.id) ||
  container.id.starts_with?(container.id) } &&
  container.refresh!.info['State']['Running']

  stats = container.stats

  # We check the maximum memory usage every second. This is obviously
  monotonic, but these stats aren't available after the container is/has
  stopped.

  memory = stats['memory_stats']['max_usage'] / (1024.0 * 1024.0) if

```

```
stats['memory_stats']&.fetch('max_usage', nil)
end timeout_mutex.synchronize do

  container.stop
  timeout = true if timeout.nil? end

end
```

```
begin
  outlines, errlines = container.tap(&:start).attach(
    stdout: true,
    stderr: true
  )
  ensure timeout_mutex.synchronize do timer.kill
  timeout = false if timeout.nil? end
end
```

```
after_time = Time.zone.now
stdout = truncate(outlines.join.force_encoding('utf-8'))
stderr = truncate(errlines.join.force_encoding('utf-8'))
exit_status = container.wait(1)['StatusCode']
container.delete
```

```

        @report.update(
stdout: stdout, stderr: stderr,
exit_status: exit_status, memory:
memory,
run_time: (after_time - before_time)
)

```

```

@report.collect_files_from(@output_dir)

```

```

if exit_status == 137
@report.update(status: 'failed', error: 'out-of-memory') elsif
exit_status != 0
@report.update(status: 'failed', error: 'non-zero exit code')
elsif !@report.all_files_present?
@report.update(status: 'failed', error: 'some output files missing')
else
@report.update(status: 'finished')
end end

```

```

def finalize

```

```

# remove path on file system used as temporary working directory for analyzing
the dataset

```

```

FileUtils.remove_entry_secure(@mount, verbose: true)

```

```

@mount = nil end

```

```

def truncate(string)

  max = OUTPUT_LIMIT

  string.length > max ? "#{string[0...max]}... (truncated)" : string

end

end

require 'docker'

# Creates a report by analyzing an input dataset using the Dolos dockerfile
#
# This is largely based on Dodona's SubmissionRunner class
AnalyzeDatasetJob < ApplicationJob

queue_as :default

TMPDIR_PATH = '/tmp'.freeze
OUTPUT_DIRNAME = 'result'.freeze
DOLOS_IMAGE = 'ghcr.io/dodona-edu/dolos:latest'.freeze
TIMEOUT = 60.seconds
MEMORY_LIMIT = 2_000_000_000 OUTPUT_LIMIT =
65_000

def perform(report, **options)

  @report = report

  @options = options

  @dataset = report.dataset

```

```

@mac = RUBY_PLATFORM.include?('darwin')

prepare

@dataset.zipfile.open do |zipfile_tmp| execute(zipfile_tmp.path)

end

rescue StandardError => e

@report.update( status: 'error',

error: truncate("Error while running Docker: #{e}\n" + e.backtrace.join("\n"))

)

ensure

finalize end

private

def prepare

@report.update(status: 'running')

@mount = Pathname.new Dir.mktmpdir(nil, @mac ? TMPDIR_PATH : nil)
@output_dir = @mount.join(OUTPUT_DIRNAME) end

def execute(zipfile_path)

cmd = [

```

```

'-V', # enable verbose errors

'-f', 'csv',

'-o', OUTPUT_DIRNAME,

'-n', @dataset.name, '/input.zip'

]

cmd += ['-l', @dataset.programming_language]
if
@dataset.programming_language.present?
docker_options = { Cmd: cmd,

Image: DOLOS_IMAGE,

name: "dolos-#{@report.id}",

NetworkDisabled: true, HostConfig: {

Memory: MEMORY_LIMIT, MemorySwap:

MEMORY_LIMIT,

PidsLimit: -1, Binds: [

"#{zipfile_path}:/input.zip:ro",

"#{@mount}:/dolos"

]

}

}

```

```

first_try = true

```

```

begin

```

```

  container = Docker::Container.create(**docker_options)

```

```

rescue StandardError => e
# Rethrow if this is our second try
throw e unless first_try

first_try = false

sleep 1

# Create can fail due to timeouts if the worker is under heavy
# load. Usually the container is still created, but we just
# don't know about it in time. Make sure the old container is
# deleted before we retry creating it to avoid name conflicts. begin

Docker::Container.get(docker_options[:name]).tap do |c| c.stop

c.remove

end

# rubocop:disable Lint/SuppressedException

# If the container does not exist the library raises an
# error. We can ignore this error, since we can skip the
# previous step anyway in that case. rescue StandardError

end

# rubocop:enable Lint/SuppressedException

# We also still change the name, because the removal can fail as well.

docker_options[:name] = "dolos-#{@report.id}-retry"

retry end

```

```

# run the container with a timeout.

memory = 0

before_time = Time.zone.now

timeout_mutex = Mutex.new

timeout = nil

timer = Thread.new do
  while Time.zone.now - before_time < TIMEOUT sleep 1

  next if Rails.env.test?

  # Check if container is still alive

  next unless Docker::Container.all.any? { |c| c.id.starts_with?(container.id) ||
  container.id.starts_with?(container.id) } &&
  container.refresh!.info['State']['Running']

  stats = container.stats

  # We check the maximum memory usage every second. This is obviously
  monotonic, but these stats aren't available after the container is/has
  stopped.

  memory = stats['memory_stats']['max_usage'] / (1024.0 * 1024.0) if
  stats['memory_stats']&.fetch('max_usage', nil)

  end timeout_mutex.synchronize do container.stop

  timeout = true if timeout.nil? end

end

```

```

begin
  outlines, errlines = container.tap(&:start).attach(
    stdout: true,
    stderr: true
  )
  ensure timeout_mutex.synchronize do
    timer.kill
    timeout = false if timeout.nil? end
  end
end

after_time = Time.zone.now
stdout = truncate(outlines.join.force_encoding('utf-8'))
stderr = truncate(errlines.join.force_encoding('utf-8'))
exit_status = container.wait(1)['StatusCode']

container.delete

    @report.update(
      stdout: stdout, stderr: stderr,
      exit_status: exit_status, memory:
      memory,
      run_time: (after_time - before_time)
    )

```

```

@report.collect_files_from(@output_dir)

if exit_status == 137
  @report.update(status: 'failed', error: 'out-of-memory') elsif
  exit_status != 0

  @report.update(status: 'failed', error: 'non-zero exit code') elsif
  !@report.all_files_present?
  @report.update(status: 'failed', error: 'some output files missing')
else
  @report.update(status: 'finished') end
end

```

```

def finalize

# remove path on file system used as temporary working directory for analyzing
the dataset

FileUtils.remove_entry_secure(@mount, verbose: true)

@mount = nil end

```

```

def truncate(string)

max = OUTPUT_LIMIT

string.length > max ? "#{string[0...max]}... (truncated)" : string

end

end

```

CERTIFICATE



Plagiarism Detection Using Machine Learning Approach in Scientific Papers

Siddhartha Reddy¹ Karthik Reddy² Dr. SIVA SANGARI A³

^{1 2 3} Department Of CSE, Sathyabama Institute Of Science And Technology, Chennai, India

Siddhartha Reddy¹ : Chilumulasiddharthareddy43@gmail.com

Karthik Reddy² : Patelkarthikreddy2800@gmail.com

Dr. SIVA SANGARI A³ : sivasangari.it@sathyabama.ac.in

Abstract— Plagiarism is the attempt to steal someone else's idea and portray it as your own, which is both unlawful and unethical. Plagiarism detection is described as the automatic detection of written items that have been reused. Modern Plagiarism detection software often examines text documents using a text representation and text similarity approaches to look for words that appear in both suspicious and source documents. The widespread use of the internet and simple access to textual content has increased the demand for automated plagiarism detection. This paper proposes an intelligent approach to detect semantic plagiarism in scientific papers. A corpus for storing the text of source scientific papers in order to compare the suspicious documents with it and detect plagiarism have been created. The documents are clustered into numerous groups using the Mini-Batch K-Means clustering algorithm, then get the documents in a specified category. The Universal Sentence Encoder (USE) intelligent model, was employed to extract textual features. The report of plagiarism percentage will be displayed as a web page and can be downloaded as a pdf file. A decision is made to accept or reject the suspicious scientific paper according to a threshold value. To evaluate the proposed system, it is compared with Turnitin platform for Plagiarism Detection (PD), the result of comparison indicates that the proposed system is reliable. The findings promote and can reduce the percentage of plagiarism after deleting some unnecessary parts of the suspicious document.

INTRODUCTION

Plagiarism is defined as presenting someone else's work or ideas as one's own without explicitly acknowledging the original source, which is both immoral and illegal. The detection and prevention of such duplications is critical in this regard. Textual data must be mathematically represented after being processed through the techniques of Natural Language Processing (NLP) algorithms and hence help in such detections. Text similarity measures the closeness between two texts. The detection of plagiarism, particularly in scientific documents, is a well-known use of text similarity concept. Intelligent algorithms in text mining applications represent words or

sentences as a list of numbers. This numerical representation includes a significant portion of the text data's syntactic and semantic rules. The usage of vectorized representations of text data allows for easier comparison of words and sentences while also reducing the requirement for lexicons. Plagiarism can include alterations in the semantic representation of the text. If two texts have a same meaning, they are considered as semantically similar. Extrinsic plagiarism detection aims to find plagiarism in a text through testing all of the available sources while Intrinsic plagiarism detection examines a questionable document and attempts to identify parts of it that were not created by the same author.

Furthermore, Plagiarism is classified into three categories according to the number of hijacked words, phrases, or paragraphs: severe or entire plagiarism (more than 70%), intermediate plagiarism (between 30% and 70%), and light plagiarism (less than 30%).

This paper's primary contributions are: i) Create a corpus consisting of 500 scientific papers ii) Using a parser to exclude the unnecessary parts from the suspicious document and intelligent technique for clustering documents to check only related documents iii) finally generate a pretty plagiarism ratio report as a web page and as PDF file.

The paper is divided into seven sections. The second section goes into previous studies. The third section outlines the required methodologies in the field of Plagiarism Detection (PD). In section four, you will find further information on the proposed approach. The results of the experiments are provided in section five. Finally, parts six and seven presents the discussion, conclusion and suggestions for future works respectively.

LITERATURE REVIEW

A number of studies have been conducted on the use of information technology to detect plagiarism. Using the algorithms of cosine distance and K-means, W. Usino et al. proposed a computerized approach for detecting information about plagiarism. The process starts with a novel step of checking an Indonesian large dictionary, followed by the building of a vector space model and a combined computation of K-means and cosine distance from 17 test items.

In I. Indriyanto and I. Sumitra, the cosine similarity is combined with the preprocessing techniques and TF-IDF to calculate the level of similarity between the title and abstract of a student's final scientific thesis. The results are then displayed and compared to the existing final project repository based on the threshold value to determine whether scientific work can be accepted or rejected.

P. Hrkút, et. Al. constructed a plagiarism detection method for a big dataset of source codes. The entire system is made up of various components. A classification of source code is an important aspect of the system since it divides data into individual clusters, making it easier to search vast amounts of source code. This is because grouping similar parts of the source code into appropriate clusters allows for a faster and more memory-efficient search for similar parts of the code.

The techniques of machine learning (ML) and (NLP) were used in the suggested system by F. K. AL-Jibory, an external plagiarism detection approach based on text mining and similarity analysis was used by the system. The proposed method employs a mixture of Jaccard and cosine similarities. Keya et al. evaluated how closely the context and question-by-question system matched. Then they locate the response that is the closest to the original query.

They measured the similarity using the cosine distance of the text and used USE for

embedding. To discover the best related text, they employed deep averaging network (DAN). H. El Mostafa and B. Faouzia offered a deep Learning Algorithm-based method for detecting plagiarism in thoughts. The suggested method addresses some of the issues that can arise when attempting to detect plagiarism of ideas, such as loss of meaning or the difficulty in recognizing semantic similarities between documents. So, first, use doc2vec to produce a numerical representation of each sentence in a document, then use the Siamese Long Short-Term Memory (LSTM) to teach our system that two documents are similar or not, and finally, use the CNN technique to identify the various types of plagiarism.

METHODOLOGY

In order to carry out the proposed approach for detection of plagiarism, the following paragraphs will explain some required methodologies.

A. Text Parsing and Preprocessing

A Text Parsing is a process of extracting text from documents such as Docx and PDF files. Convert text in a certain format into a more usable format.

In the proposed approach, after extracting the text, the parser will extract the title and author name of the suspicious paper. Utilizing Regex and string manipulations, the unnecessary text can be deleted. The text segments before the abstract heading and after the acknowledgement or reference headers are deleted from the suspicious document's text so that the deleted texts are not tested for plagiarism.

NLP is a wide term that covers a wide range of activities and applications, such as text categorization, machine translation, text similarity detection, and so on.

Pre-processing a text simply means converting it into a format that the machine can interpret, predict, and analyze using various machine learning methods utilizing the NLP techniques.

The following are some of the most commonly utilized pre-processing techniques utilizing NLP:

Lowercase the words: since the computer assumes that the same words with one capital and one lowercase are two different words. Tokenization: the process of breaking down sentences into words, letters, and punctuations, this stage aids in the subsequent processing processes by filtering out undesirable terms.

Stop words Removal: Words like "the," "are," "is," "and," and so on have no meaning and these extra words aren't given any weight.

Remove all punctuation marks because the machine doesn't understand punctuation and its presence causes the text to become loud.

Stemming: is a word shortening strategy that seeks to get a word back to its root, but the semantic meaning of all the forms remains the same.

Lemmatization: is the process of deleting or changing a suffix from a word in order to return it to its original form.

The process of generating a set of n-tokens of a text from a text document or corpus is known as N-gram. The n-gram can do at character level, word level or even at phrase level. This proposed approach utilized splitting the text into n-words.

B. The Corpus

In a PD system, a suspicious document is compared to a corpus or set of reference documents. It is possible that this reference collection will be available online or offline, using online sources on the internet or an offline database with the source materials [22]. The Turnitin platform is a public used web site for detection of plagiarism and similarity index, it has a huge online and off-line corpus of text documents. To create the corpus and since it is impossible to have a huge corpus of documents as that of for example, the Turnitin platform, the proposed approach created a corpus consisting of 500 scientific papers as text files, that files are created from the text of pdf or docx files, the text files are parsed and preprocessed then stored in the corpus. Furthermore, A Comma Separated Values (CSV) file containing the title, author, keywords, count of pages, count of sentences, count of words, and size of each scientific publication has been prepared.

C. The Clustering of document

Clustering is an intelligent technique that groups objects together according to common features. Document clustering Fig. 1, often known as text grouping, is primarily concerned with grouping documents based on comparable content or file categories. NLP techniques can be used to identify similarity between documents based on semantic relationships.

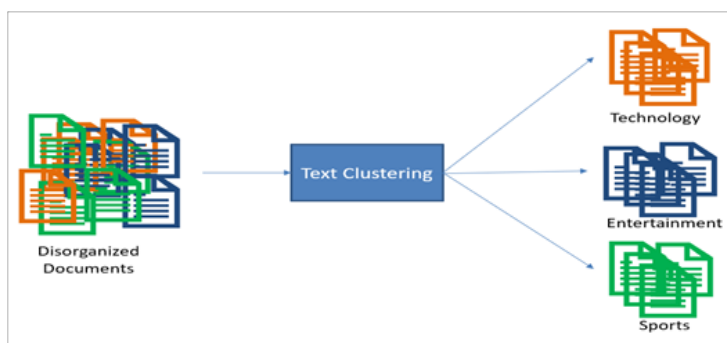


Fig. 1. Document Clustering

The Mini-Batch K-means algorithm for clustering is an improved version of k-means algorithm, the most widely used technique for clustering documents. The algorithm of Mini- Batch K-means is both simple and quick [24]. The Mini-Batch K-mean was utilized in certain research publications because it can work with large datasets and enhance accuracy [12], [17]. In the proposed approach, the documents are subsequently grouped using the Mini-Batch K-mean approach according to titles and keywords of the scientific documents.

D. Extraction of Textual features

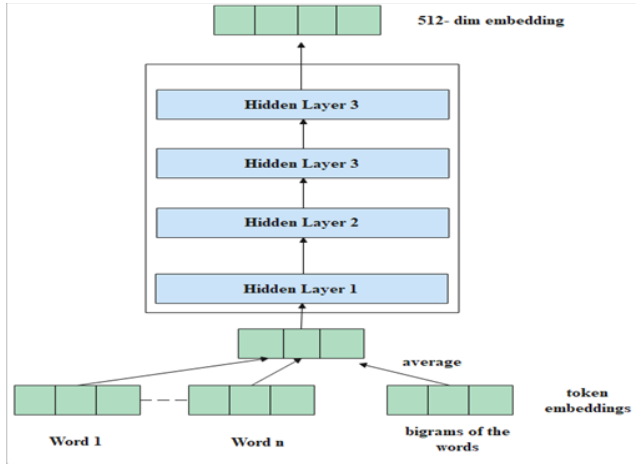
Textual feature extraction also known as text encoding, vector representations and word or sentence embedding. It is the process to extract numeric features of text in order for machines to understand them [18], [22]. The sentence embedding involves encoding the text into large vectors which preserves the contextual and semantic meaning of sentences, these vectors can be utilized for a variety of purposes, including text categorization, grouping, semantic similarity, etc.

This paper used USE which is a pretrained deep learning based feature extraction model. The two USE variants are the Transformer encoder and Deep Averaging Network(DAN). The first one is more accurate, but it costs a lot to compute. The second one is less computationally demanding but less precise.

When English text of varying length is input, the result is a 512-dimensional vector.

A DAN encoder is utilized to train the used model. The model generates embeddings from a sentence or word as input as illustrated in Fig.2. They function as follows :

- a) Sentence tokenization is carried out after lower case conversion.
- b) The average of the word and bi-gram embeddings within a sentence is calculated.
- c) Pass it through a single or more layers feed-forward deep neural network.
- d) The sentence is then converted into a 512- dimensional embedding. During training, the embeddings for words and bi-grams are discovered.



For the purpose of creating 512 dimensional final embeddings for sentences, the DAN computes unigram and bigram embeddings before passing the results to a deep neural network.

E. Text Similarity Measurement Technique

Following the acquisition of each sentence's vector representation, the similarity measure can be used to detect similarity between each two-sentence pair.

The cosine similarity method determines the direction of two vectors by measuring the cosine angle between them[28]. The similarity between two vectors is calculated using the ratio between their dot products and the product of their lengths.

THE PROPOSED APPROACH

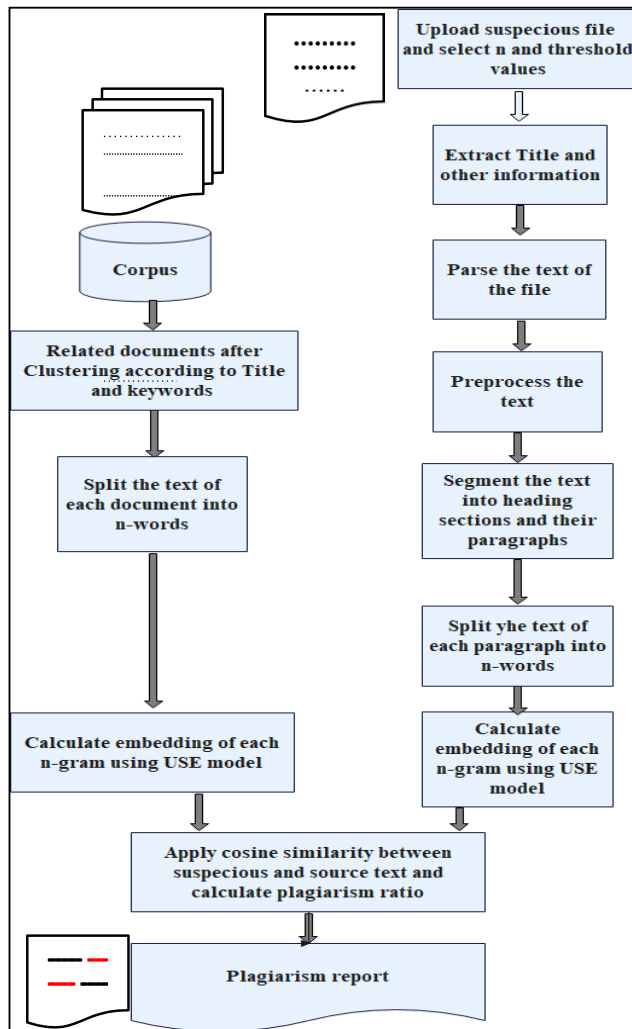
Using the methodologies provided in the methodology section, the proposed approach for semantic PD utilized USE and cosine similarity measure. In approach has several stages. The proposed approach is explained in the next sections.

As shown in Fig. 3, the Semantic PD approach is carried out through several steps as following:

1. To create the corpus, 500 scientific papers as pdf and docx files are collected, then the text from each file is extracted and saved as a txt file. The topics of the files are divided into six categories: computer science, arts, physics and chemistry, geography and history, biology, and sports. Table I shows the number of papers in each category.

Topic	Count of scientific papers
Computer science	106
Arts	76
Physics and Chemistry	85
Geography and History	93
Biology	86
Sports	54
Total	500

2. The suspicious file is uploaded by user and N value is selected for splitting text into n-word tokens and a threshold value.
3. The keywords, Title, and other information are extracted from the suspicious document, Title and Keywords are encoded using USE model.
4. Parse the suspicious document and extract the text between abstract and acknowledge or references headings.
5. Preprocess the parsed document utilizing the NLP and perform Mini-Batch K-means clustering on corpus text to get the related documents according to the numerical vectors of title and keywords.
6. Split the text of source documents in corpus into n-words.
7. Segment the suspicious text into segments of heading sections and their paragraphs, like Abstract, Introduction, etc., the texts of Abstract, Result and conclusion heading sections will have high weights and compared with threshold – 0.1 to detect plagiarism. Since the author must not plagiarize any text at these segments.
8. For the paragraphs of each section of a suspicious document, apply n-gram to convert it to n-word tokens.
9. Calculate the USE text encoding.
10. Apply the cosine similarity between suspicious and corpus text vectors. Each similarity greater than threshold is marked as plagiarized.
11. Generate the detailed report that contains the total percentage of plagiarism and percentage of plagiarism from each source. Highlight the plagiarized sentences according to assigned color for the source of each sentence and other information



In this paper, the phases of the suggested method were applied to a number of suspicious documents. Regardless of whether the document is a pdf or a docx, it will be converted to text and processed according to the rules of the proposed system.

The results of Semantic PD of ten suspicious documents are shown in Table II. as percentages of plagiarism, when n- words equal to 5 and 7.

The application will decide whether to accept or reject the suspicious document based on a threshold value, such as that used in many scientific institutes. The document is accepted if the percentage of plagiarism is less than 20%; else, the document is rejected and must be re-edited before being checked again.

Doc No.	Semantic PD Percentage			
	N = 5	Accept	N = 7	Accept
1	0.26	No	0.23	No
2	0.13	Yes	0.11	Yes
3	0.25	No	0.21	No
4	0.22	No	0.18	Yes
5	0.21	No	0.18	Yes
6	0.17	Yes	0.12	Yes
7	0.23	No	0.20	No
8	0.27	No	0.22	No
9	0.46	No	0.41	No
10	0.18	Yes	0.15	Yes

An easy-to-use user interface has been used to present the system, it accepts the input Pdf, Docx or txt file formats. The file is uploaded to the system. After that, it is verified for plagiarism.

The result is displayed as a plagiarism report which indicates the acceptance of the document according to plagiarism ratio. The ratio can be less than 20% and is accepted and rejected if the ratio is exceeded 20 %.

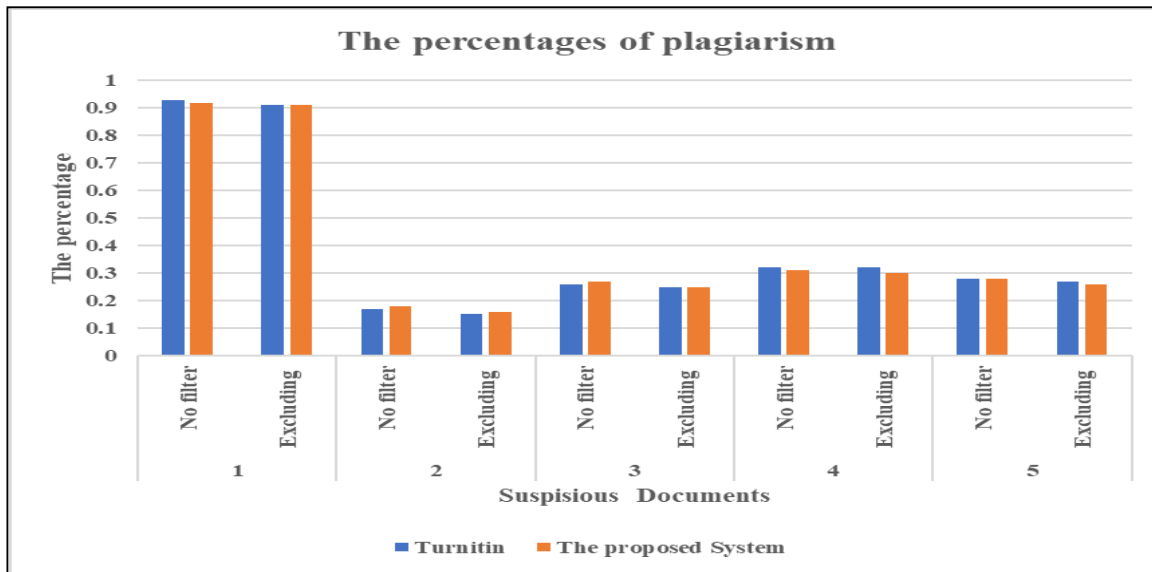
To assess if the proposed method is reliable or not, a sample of five suspicious document are checked for plagiarism using Turnitin web platform, then collect the source documents for each suspicious document. The collected documents are considered as source documents in the proposed system and each of the suspicious documents are checked for plagiarism.

Table III illustrates the percentages of plagiarism of each suspicious document in Turnitin and proposed system without any filters and with excluding bibliography and quotes and the time required for each document as minutes and seconds.

Figure 4. shows the percentages of plagiarism of each suspicious document in Turnitin and proposed system with and without filtering.

After carrying out the experiments and comparing with the results of the Turnitin platform, the proposed system is reliable and produced reasonable and glorious results regarding the accuracy. On the other hand, The Turnitin platform is better regarding the execution time since the proposed approach is using pretrained USE sentence embedding model which requires more computational resources and thus execution time. However, the USE model provides quick results and require fewer resources compared with Bidirectional Encode Representation from Transformers (BERT) and Embeddings from Language Models (ELMo) pretrained models.

Generally, since the speed of the internet is not stable and the loads on CPU which affect to the execution time, the accuracy is more important than the speed of processing in such plagiarism checking systems and the majority of authors make attention to accuracy rather than the run time.



The steps of the proposed system in this paper were carried out on many of the source and suspicious documents utilizing the Python language with its libraries such as Natural language Tokit (NLTK), regular expressions (re) and others in the Flask framework. Whatever sort of document it is, it will be converted to text and processed according to the system's steps. The information of each scientific paper including title, author, keywords, year of publishing, topic, size, count of words, count of sentences and number of pages are stored in a CSV file while the corpus contains 500 text files of the Scientific papers.

In this paper, work was based on intelligent USE model for detection of semantic plagiarism. The input is document file and the output is a report about the semantic plagiarism. That exists in the suspicious document in the corpus. According to the experiments, semantic plagiarism has been detected in a sample of documents stored in the corpus with different percentages. The value of n is very important for text segmentation, after many experiments it's noticed that when this value increases the plagiarism percentage is decreases and vice versa.

CONCLUSIONS AND SUGGESTIONS

A plagiarism detector has been developed in this paper employing machine learning algorithms such as USE for text encoding, Mini-Batch K-means for clustering, NLP techniques and cosine similarity. The technology works well and detects the level of plagiarism in the text files provided. This technique may simply be used in educational institutions such as schools and colleges to detect plagiarism. The system's operation does not necessitate any complicated instructions or training. It's a plagiarism detection method that's quick, simple, and effective.

The findings of this paper may make it easier to spot similarities in each scientific paper and determining whether or not the proposed paper fits the originality semantically. However, this method is limited since it only uses few counts of documents in the corpus. More investigation may be done to see whether a corpus that is more compact and has a more and more data might be used.

This paper advises adding techniques to increase system performance and accuracy such as assign similarity weights for each header and its section in the suspicious paper, increase the count of documents in the corpus in the current used topics and from many other topics and utilizing the intelligent clustering techniques in selection of the related source documents from the corpus.

ACKNOWLEDGMENT

The authors would like to extend their thanks and gratitude to the College of Computer Science and Mathematics - Department of Computer Sciences at the University of Mosul for their unlimited encouragement and support.

REFERENCES

- 1 El Mostafa HAMBI and F. Benabbou, "A Multi-Level Plagiarism Detection System Based on Deep Learning Algorithms," *IJCSNS International Journal of Computer Science and Network Security*, vol. 19 no. 10, pp. 110-117, 2019.
- 2 E. Gharavi, K. Bijari, K. Zahirnia, and H. Veisi, "A deep learning approach to persian plagiarism detection," *FIRE (Working Notes)*, vol. 34, pp. 154-159, 2016
- 3 E. S. Al-Shamery and H. Q. G. ALkhafaji, "Plagiarism and Source Deception Detection Based on Syntax Analysis," *Journal of University of Babylon*, vol. 25, no. 2, 2017.
- 4 M. Umadevi, "Document comparison based on tf-idf metric," *International Research Journal of Engineering and Technology (IRJET)*, vol. 7, no. 02, pp. 1546-1550, 2020
- 5 S. Chawla, P. Aggarwal, and R. Kaur, "Comparative Analysis of Semantic Similarity Word Embedding Techniques for Paraphrase Detection," in *Emerging Technologies for Computing, Communication and Smart Cities: Springer*, 2022,

- pp. 15-29.
- 6 D. Sakamoto and K. Tsuda, "A detection method for plagiarism reports of students," *Procedia Computer Science*, vol. 159, pp. 1329- 1338, 2019.
 - 7 I. Indriyanto and I. Sumitra, "Measuring the Level of Plagiarism of Thesis using Vector Space Model and Cosine Similarity Methods," in *IOP Conference Series: Materials Science and Engineering*, 2019, vol. 662, no. 2: IOP Publishing, p. 022111.
 - 8 M. Sabeeh and F. Khaled, "Plagiarism detection methods and tools: An overview," *Iraqi Journal of Science*, pp. 2771-2783, 2021.
 - 9 Viji Devarajan and R. Subramanian, "Analyzing semantic similarity amongst textual documents to suggest near duplicates," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 25, no. 3, 2022, doi: <http://doi.org/10.11591/ijeecs.v25.i3.pp1703-1711>.
 - 10 B. Leonardo and S. Hansun, "Text documents plagiarism detection
 - 11 using Rabin-Karp and Jaro-Winkler distance algorithms," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 5, no. 2, pp. 462-471, 2017.
 - 12 W. Usino, A. S. Prabuwno, K. H. S. Allehaibi, A. Bramantoro, A. Hasniaty, and W. Amaldi, "Document similarity detection using k- means and cosine distance," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 2, 2019.
 - 13 P. Hrkút, M. Ďuračík, M. Mikušová, M. Callejas-Cuervo, and J. Zukowska, "Increasing K-means clustering algorithm effectivity forusing in source code plagiarism detection," in *International Conference on Smart Technologies, Systems and Applications*, 2019: Springer, pp. 120-131.
 - 14 F. K. AL-Jibory, "Hybrid system for plagiarism detection on a scientific paper," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 13, pp. 5707-5719, 2021.
 - 15 M. Keya, A. K. M. Masum, S. Abujar, S. Akter, and S. A. Hossain, "Bengali Context– Question Similarity Using Universal Sentence Encoder," in *Emerging Technologies in Data Mining and Information Security*: Springer, 2021, pp. 305-315.
 - 16 S. Lazemi and H. Ebrahimpour-Komleh, "ParsiPayesh: persian plagiarism detection based on semantic and structural analysis," in *2020 10th International Conference on Computer and Knowledge Engineering (ICCKE)*, 2020: IEEE, pp. 525-533.