PLAGIARISM CHECKER

Major project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

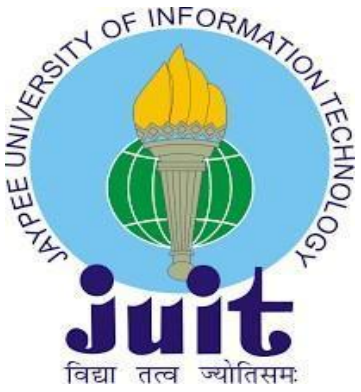
Computer Science and Engineering/Information Technology

By

RAGHAV VERMA (191359)

UNDER THE SUPERVISION OF

Dr. Pradeep Kumar Gupta
to



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan-173234, Himachal Pradesh

DECLARATION

I hereby declare that this project has been done by me under the supervision of (Dr Pradeep Kumar Gupta, Associate Professor, Deptt. Of CSE & IT), Jaypee University of Information Technology. I also declare that neither this Project nor any part of this Project has been submitted elsewhere for the award of any degree or diploma.

Supervised by:

Dr. Pradeep Kumar Gupta

Associate Professor

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology

Submitted by:

Raghav Verma - 191359

Computer Science & Engineering Department Jaypee University of Information Technology

CERTIFICATE

This is to certify that the work which is being presented in the Project report titled "**Plagiarism Checker**" in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** and submitted to the Department of Computer Science & Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by "Raghav Verma (191311)" during the period from February 2022 to May 2023 under the supervision of Dr. Pradeep Kumar Gupta, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

(Student Signature)
Raghav Verma, 191359

The above statement made is correct to the best of my knowledge.

(Supervisor Signature)

Dr. Pradeep Kumar Gupta

Associate Professor

Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat

PLAGIARISM CERTIFICATE

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ……..

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name:_____ **Department:**_____ **Enrolment No** _____

Contact No._____ **E-mail.**_____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at……..(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><td>Copy Received on</td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td>Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

……

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

First, I express my gratitude to god who provided me with the courage and fortitude to complete the project.

I am grateful and wish my profound indebtedness to Supervisor Dr. Pradeep Kumar Gupta, Associate Professor, Department of CSE Jaypee University of Information Technology, Wakhnaghat. Deep Knowledge & keen interest of my supervisor in the field of "Machine Learning" to carry out this Project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this Project.

I would like to express my heartiest gratitude to Dr. Pradeep Kumar Gupta, Department of CSE, for his kind help to finish my Project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a success. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patience of my parents.

Raghav Verma
191359

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF GRAPHS

LIST OF ABBREVIATIONS

Pandas	Numerical Python
HTML	Hypertext Markup Language
JS	Java Script
SDK	Software Development Kit
CSV	Comma Separated Values
Sklearn	Scikit-learn
SQL	Structured Query Language
NumPy	Numerical Python
ML	Machine Learning
URL	Uniform Resource Locator
HTTP	Hypertext Transfer Protocol
VS Code	Visual Studio Code
UI	User Interface
CSS	Cascading Style Sheets
OS	Operating System
API	Application programming interface
NLP	Natural Language Processing
WSD	Word Sense Disambiguation
NER	Named Entity Recognition
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

CNN	Recurrent Neural Network
CNN	Convolutional neural network
AI	Artifical Intelligence

ABSTRACT

Plagiarism is the act of using someone else's work without proper attribution and is a serious concern in academic and professional settings. Plagiarism detection and prevention have gotten harder as digital information has been more widely accessible. As a result of this expanding issue, plagiarism checkers have become important instruments for ensuring academic integrity and preserving the authority of scholarly work.

The goal of this project is to develop a free plagiarism detector for use in research and teaching. These programmes use sophisticated algorithms and language analysis approaches to find patterns and similarities in texts, from straightforward word-for-word plagiarism to more complex plagiarism techniques like paraphrasing or mosaic writing.

Although they are useful tools, plagiarism checkers also have some restrictions.Firstly, they are not widely available, and the available ones are quite expensive, making it difficult for the common people to use them to their maximum use. Additionally, it may be necessary to use human judgement to distinguish between actual plagiarism and properly referenced excerpts because false positives and false negatives may occur.

In conclusion, plagiarism checkers are essential for keeping ethical norms and preserving academic integrity. These methods use technology to help in the prevention and detection of plagiarism, fostering an originality and intellectual honesty culture. To ensure that plagiarism checkers remain relevant in the changing environment of scholarly communication and information sharing, additional study and development in this area are crucial.

Chapter 01: INTRODUCTION

1.1 INTRODUCTION

The act of plagiarism is stealing another individuals intellectual property, which comprises their literary work, creative output or ideas. The act of claiming these as ones own without due acknowledgment and consent constitutes a direct violation of the principles of academic integrity,honesty and originality. Without doubt the practice has become increasingly prevalent across myriad settings be it essay writing, research reports or artistic works a reality indicating we overlook.

A key reminder here though: while using artificial intelligence based technologies to create content may seem like a grey area doing so without an assigned license for assessment leads it into the realm of academic dishonesty. Moreover reusing ones own work without proper citation is still considered an act of plagiarism. In light of all this remember that exams have strict rules prohibiting instances of plagiarism which carry serious disciplinary consequences.

Learning and using the rules of good academic practise from the start of your university career is the greatest method to prevent plagiarism. Avoiding plagiarism involves using your academic skills to make your work as good as it can be, not just making sure your references are all accurate or changing enough terms so the examiner won't catch you paraphrasing[1].

In order to deter and fight plagiarism, educational institutions and professional communities frequently have severe regulations and sanctions in place. It is crucial to always correctly attribute and reference the sources utilised, whether through direct quotations, paraphrasing, or summarising, as well as to adhere to the particular citation style requirements specified by the institution or publication, in order to prevent plagiarism.

To assure originality and uphold ethical standards in academic and professional work, using plagiarism detectors and developing strong research and writing practises are helpful ways.

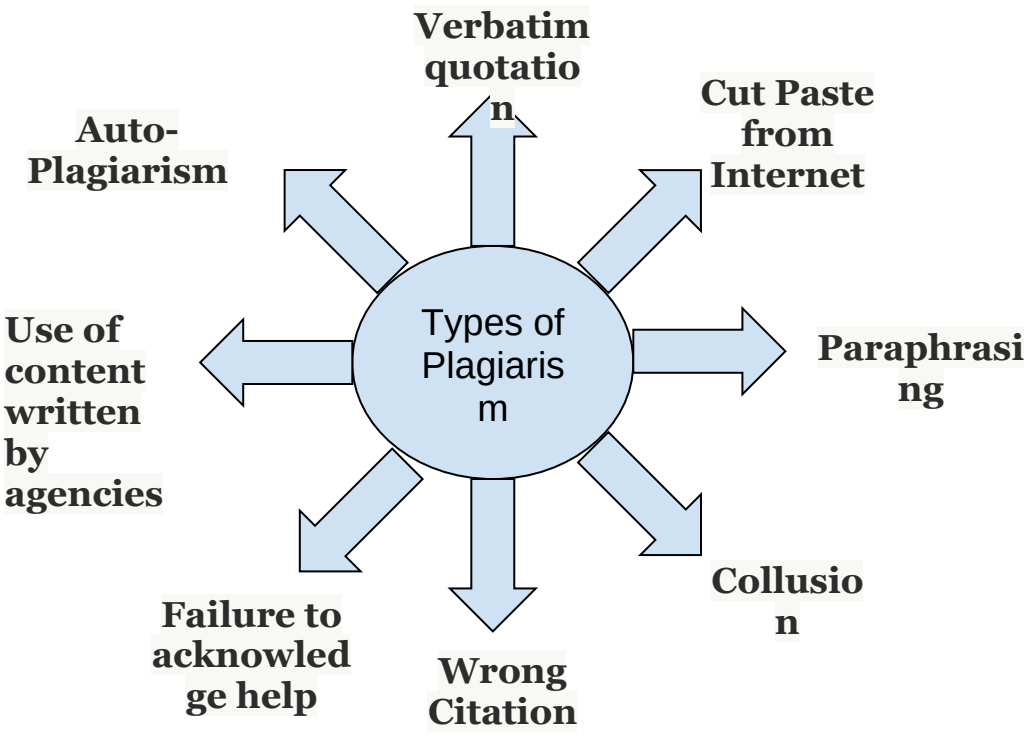


Figure 1.1 - Types of Plagiarisms

Below here, are listed 8 major types of Plagiarism practices that are commonly used by the students and researchers:

1. **Verbatim quotation**: Verbatim quotation: Statements must be identified as such with appropriate citations of the original sources and either quote marks or indentation. Always be upfront with the reader about which concepts and words you have taken directly from another source and which ones you have created yourself.
2. **Cut Paste from internet:** Any information collected from the Internet must be properly referenced and listed in the bibliography. It is vital to carefully assess anything you read online because online content is less likely to have received the same comprehensive review as traditional sources.[2]
3. **Paraphrasing**: Paraphrasing constitutes plagiarism if the author whose work you are using is not properly acknowledged. This entails altering a few words and their placement in addition to strictly following the argument's outline.

4. **Collusion**: Students may collaborate without permission, neglect to acknowledge assistance received, or fail to precisely follow the guidelines for group work projects. It is your responsibility to ensure that you are clear on how much collaboration is permitted and which parts of the work must be entirely unique.
5. **Wrong Citation**:According to the requirements of your field, it's essential to properly cite sources. Not only must you mention your sources (i.e., provide a bibliography), but you also need to identify the sources in a footnote or in-text citation. Additionally, you must not include in your bibliography any sources or things that you have not actually consulted.[3]
6. **Failure to acknowledge help**: Every contribution made to the creation of your work, including ideas from other students, lab staff, and outside sources, must be acknowledged. It is crucial to include any further guidance that leads to material changes in the content or approach, even if it is unrelated to your tutoring or supervision or to simple modification.
7. **Use of content written by agencies**: You should never submit writing done for you by a professional company or use their services in the creation of your work, even with the author's approval. This study procedure must be carried out alone because it is essential to your intellectual growth and advancement.
8. **Auto-Plagiarism**: You may not submit work for assessment that you previously submitted (in full or in part), either for your present course or for another qualification of this or any other institution, unless specifically mentioned in the particular regulations for your course. Any earlier work of yours that is publishable or citeable must be specifically cited. Similar works that have been submitted in a similar way will also be considered auto-plagiarism.

1.2 PROBLEM STATEMENT

Academic integrity is violated by plagiarism, which denotes a lack of intellectual honesty and a failure to give due credit to the authors of the ideas, words, and data that serve as the basis of one's

own work. It is not only a sign of weak scholarship; it also shows that learning is still in progress. Plagiarism undermines both the standards upheld by educational institutions and the value of the degrees they award. It is an immoral practise with serious repercussions for future employment chances.

The desire for producing exceptional work serves as a compelling inspiration for avoiding plagiarism. Understanding the fundamentals of source utilization and citation makes it quite easy to avoid plagiarism. Improving the quality and clarity of one's writing has advantages as well. Realizing that mastering academic writing approaches goes beyond being a useful talent is essential. Demonstrating dedication to the principle of intellectual honesty in academia gives one's work credibility and authority.

So, together, we all need to tackle this academic issue, which would further result in better quality of work, both academic and professional.

1.3 OBJECTIVES

The aims of this project are:

1. To identify the plagiarised content and detect it.
2. To build a framework where the user can input their required text and get the desired result, which is getting the plagiarism percentage.
3. A teacher may use a service like this, to put this into practise, to compare a student's writing with a database of other academic works.
4. If you're a writer, you might wish to see if your writing has been plagiarised and published somewhere under another name.
5. Keeping this application free of cost so that all people can access this detector.

1.4 METHODOLOGY

This project's methodology includes majorly Flask 2.0+, Python 3.9+, HTML, JS and Pinecone. To discover articles with the same or similar text to what the user has entered, it conducts a similarity check using the Pinecone SDK. A required csv file is provides to the backend, let's say, if you are a writer, it may include a database of articles, or if you are a professor, it may include a dataset of research papers. Then the required text is typed or pasted in the original content area field. By clicking the submit button, the application runs and checks the similarity between the input text and the database, thus providing with the desired output(that is similarity index). A similarity Senstivity bar is also provided which can be adjusted accordingly and provide desired results.

In the output section, Title, Publication and Match Score are displayed along with its content. So with this, the teachers can check if the content provided is original or not.

1.5 ORGANIZATION

1.5.1 Python

Python is a interactive, dynamically typed programming language. Guido van Possum created Python in 1991 and since then it has become a widely used programming language. The edit-test-debug cycle is extraordinarily quick because there is no compilation step. Python programs are simple to debug because the segmentation failure is never caused by a bug or wrong input.

Python programmes can be launched on a number of operating systems, including Windows, Mac, and Linux, and are written in plain text files. Python is good path for new and trained programmers alike because of its fast and simple code testing and debugging capabilities.

Python has a number of important features, including:

- a minimal learning curve and simple to learn syntax
- Code is "interpreted," which means it is run at runtime line by line.
- Data types can be more flexible thanks to dynamic typing.
- Automatic memory management releases memory when it is no longer required.
- A sizable standard library that gives access to frequently used modules and functions

Some of its most useful applications in various domains includes:

Web Development: Deployed on a server, Python creates dynamic and interactive web applications well.

Software Development: The use of Python in developing software applications is useful due to its provision of numerous libraries and frameworks that ease the development process.

Data Science: Vast availability of databases and open source libraries makes it easy for data scientists to run their required experiments on python.

1.5.2 Pandas

By using Pandas library one can conveniently transfer data between various file types such as CSV format or an Excel sheet and also SQL databases, and the Data Frame is like a table within a relational database and allows you to easily manage structured information. On this platform users can work with their data in many ways such as performing filtering or aggregating operations.

Effective handling of missing data is one of the key strengths of Pandas which offers options such as imputation and removal, and another key point about Pandas is that it works perfectly with other widely-used Python libraries like NumPy and Matplotlib which gives users the flexibility to analyze their data efficiently.

Python users can rely on Pandas' library for smooth and efficient manipulation of data sets and achieve quick meaningful results, as Pandas is crucial for data science tasks like cleaning and exploring datasets as well as constructing machine learning models.

For those new to data science or more experienced users looking for efficient exploration tools with clear syntax structure, Pandas is definitely worth considering, and active community support combined with extensive documentation has contributed significantly to its widespread usage.

1.5.3 Pinecone

It is a fully maintained vector database which makes it simple to integrate vector search into production applications. Modern vector search libraries, modern features like filtering, and distributed infrastructure are all combined to offer exceptional performance and dependability at any scale. There will be no more difficulties with establishing and maintaining infrastructure for vector search, benchmarking algorithms, or tuning them.

1.5.4 Flask

Flask is a web framework in python which is lightweight and easy to use. It was created by Armin Ronacher in 2003. Compared to the Django web framework, Flask is thought to be more Pythonic because the comparable web application is usually more explicit. Due to the lack of boilerplate code required to launch a straightforward app, Flask is also simple to use for beginners.

Below are some of the Python Flask modules:

- Routing and URL Mapping: It has a decorator based routing system making it easier for users to link URLs.
- HTTP Request handling: It simplifies HTTP requests' handling by providing request and response objects.
- Error Handling: It provides methods to handle errors. Custom error handles can be defined by developers to give a better experience.
- Deployment: Applications can be deployed using various web servers.
- Development Server and Debugging
- Extensions

1.5.5 JavaScript

It is an high-level programming language which focuses on front-end web development. It makes webpages more dynamic and interactive, improving user experience and interaction. JS is the world's most popular programming language and programming language of the Web. It is known for its flexibility and range of applications. It also allows users to create interactive web pages, along with functionality and features like validation and arithmetics.

Here are some of the features of JS:

- Scripting
- Interpreter
- Event Handling
- Case Sensitive
- Dynamic Typing
- Platform Independent

1.5.6 VS Code

VS Code is a source code editor which can be utilised with a range of programming languages like C, C++, c#, Python, HTML and many more. It was first announced in 205 and is developed by Microsoft. It is written in JavaScript, HTML, CSS and TypeScript.

Various features offered by VC Code:

- Easy customisation of UI
- Cross-Platform Support: VS Code is available for windows, macOS, Linux, this providing coding platforms across differents OSs.
- Git Integration: Git integration is included into VS Code, giving users access to version control features right inside the editor. Without using a different Git client, developers may inspect Git diffs, stage changes, commit, push, and pull code from Git repositories.
- Syntax highlighting
- Bracket matching
- Code folding
- Intellisense for CSS, HTML, JS
- Many extensions
- Multiple Cursors
- Live Sharing: It enables developers to have real time collaboration, which allows them to work together on the same project

1.6 DELIVERABLES OF THE MAJOR PROJECT

An application with the following key features:

- A real time running plagiarism checker hosted on local system, which can also allow you to customise similarity index and run checks on the provided input texts. The dataset can also be customised depending upon your usage.
- A complete paper containing a literature review, methodology, analysis, and conclusion as well as the research and conclusions of the project

Chapter 02: LITERATURE SURVEY

2.1 FEASIBILITY STUDY

A plagiarism checker's practical practicality and possible advantages are revealed via a feasibility study. Technical viability and market demand are all evaluated in the study. It is technically possible to create a reliable plagiarism checker because to developments in machine learning and natural language processing. Given the rising importance of originality and academic integrity, there is a significant market need for these tools. A plagiarism detector may earn money from subscriptions or licencing arrangements. Algorithm precision, database size, and competition are obstacles, nevertheless. Overall, the feasibility study indicates that creating a plagiarism checker has potential, but its implementation requires careful market analysis and ongoing improvement. In order to analyse the likelihood of success and the advantages of such a tool, the study assesses technical, market, and financial factors.[4]

- Technical Possibility: Development of a reliable plagiarism detector is now theoretically possible because to developments in natural language processing and machine learning. Libraries, algorithms, and APIs are readily available, which makes implementation easier. However, difficulties include creating reliable algorithms to identify plagiarism quickly and accurately, guaranteeing scalability, and keeping a large library of sources for comparison.

- Market need: The growing need for originality, academic integrity, and content authenticity is driving the demand for plagiarism checkers. Potential target consumers include educational institutions, content producers, researchers, and publishers. To match the product with market demand, market research must be conducted to determine the precise demands, preferences, and pricing expectations of the target audience.[5]

- Competition: The market for plagiarism checkers is crowded with established companies providing comparable services. Gaining a competitive edge can be achieved through researching the products of rivals, differentiating the product with special characteristics, and increasing algorithm performance. Increased market penetration may also result from partnerships with academic institutions, content platforms, or software integration.

- Conclusion: It seems promising to construct a plagiarism checker based on the feasibility research. Developments in machine learning and natural language processing help the technical viability. The demand on the market is substantial and is fueled by the desire for originality and content integrity. But it's important to deal with competitiveness and issues like algorithm precision and database upkeep. To succeed in the market for plagiarism checkers, one must conduct in-depth financial analysis, market research, and legal compliance.[6]

Therefore, there is a need of the detection of such copied content. This is what this project aims at.

2.2 LITERATURE SURVEY

When it comes to Plagiarism Checker, many Machine Learning and NLP have been proposed by researchers all around the globe. Each method proposes methods for this, where some have performed better than others. Some methods are discussed below:

N. Selvi et al.[7] in their research used the Apache Lucene plagiarism detector. The original document is first indexed, and the copied document is then compared to a set of previously saved documents using cosine similarity. It was created by Doug Cutting and is free source software.It is appropriate for any software that has to be able to search and do indexing. It can be used for

both single-site and local searches.It has been integrated into our programme using the Lucene-core-3.6.0 jar file. The Apache Software Foundation backs it. Both open source and for-profit software can use it. In its most basic form, cosine similarity is only a mathematical idea, yet it has numerous applications in Relevance Ranking, Text Mining, and Information Retrieval.

Y. Wang et al. [8] provided an overview of effective plagiarism detection techniques that have been applied to code source plagiarism detection, external plagiarism detection, clustering-based plagiarism detection, and natural language text plagiarism detection. They also compared five different textual plagiarism detection software, including PlagiarismDetection.org, PlagiarismAware, PlagScan, Check for Plagiarism, and iThenticate.

Features	PlagAware	PlagScan	iThenticate	CheckFor Plagia-rism.net	Plagiarism detect-ing.org
Database Checking (on-line and offline)	*****	*****	*****	****	****
Internet Checking	*****	*****	*****	*****	*****
Publication Checking	*****	*****	*****	***	***
Multiple document com-parison	*****	*****	*****	****	****
Multiple languages sup-port	*****	*****	*****	*****	*****
Sentence structure and synonym checking	****	**	****	*****	**

Table 2.1 - Softwares' Comparison[8]

S. Mishra [9] showed the detailed survey of existing plagiarism detection tools. Many tools have been studied such as Antiplag. Copyscape, Paper Rater, Plagiarism Checker, Plagiarisma.Net, Plagium,See Sources, PlagTracker, Plagiarism Detector, Turnitin, Viper etc.

M. P. Bressan et al. [10], did an analysis of all the plagiarism detecting software programmes in use across all Indian universities. Plagiarism of all kinds is likely to occur since many scholars

publish their work in their native tongues, such as Hindi, Marathi, Tamil, etc. The functioning of Turnitin was also discussed by the author. The author noted that only nineteen languages, mostly those spoken in the western world, can be scanned by the most widely used and well-known anti-plagiarism software tool, turnitin. Because of this, it is difficult to detect plagiarism and will pose a significant challenge for the entire nation in the years to come.

Y. P. Ma et al. [11] have thoroughly surveyed and used a few commonly used plagiarism detection software, including plagscan, plagaware, plagiarism detection, Jplag, Ithenticate, etc. and used CNN models.

S. Shukla et al., in [12] used a technique called "Maulik" for Hindi documents for plagiarism detection. By breaking up the text into n-grams and utilising a synonym substitution approach, the research was conducted, and the sentence structure was noted.

N. Tahir et al. [13] in their researchhas examined the effectiveness of various plagiarism detecting techniques

		Sample paper used for testing				
S. N.	Tool Name	Extraction of Root Words	Part of Speech TAGGER	Sentiment Classification in Hindi	Knowledge Discovery in Text Mining	Assist Crime Prevention
1	Duplichecker	45%	50%	26%	41%	31%
2	Quetext	32%	91%	8%	16%	43%
3	Smallseotools	45%	50%	36%	43%	31%
4	plagscan	95.30%	72.70%	69.30%	97.46%	32.80%
5	searchenginereports	49%	44%	37%	41%	33%
6	PaperRater	0%	50%	0%	0%	0%
7	plagramme	43%	59%	51%	71%	58%
8	Prepostseo	29%	49%	36%	52%	31%
9	papersowl	43%	71.20%	15.40%	21.50%	14.30%
10	plagiarism software	62%	91%	74%	62%	67%
11	Exactus Like	5.18%	14.23%	12.15%	5.81%	5.53%

Table 2.2 - Tools and its evaluation using relevant research papers [13]

2.2.1 Plagiarism Definition

Plagiarism can be defined in a number of ways. It is a practise where a user exploits already published information or words authored by someone else without giving credit to the original author. Someone else's intellectual property is being stolen without their consent. Plagiarism is not a recent phenomena; in fact, it has been practised for a long time.

The subject of plagiarism's causes and consequences and its symptoms arises. It is simple for students and teachers to just copy and paste the information offered on the internet without referencing the original source in an educational system where internet usage is increasing. To avoid being caught while plagiarising, many people alter the text, replace terms with others, use synonyms instead of the original words, do paraphrase, change active to passive or vice versa, and other techniques.[14]

Plagiarism can be classified into various types -

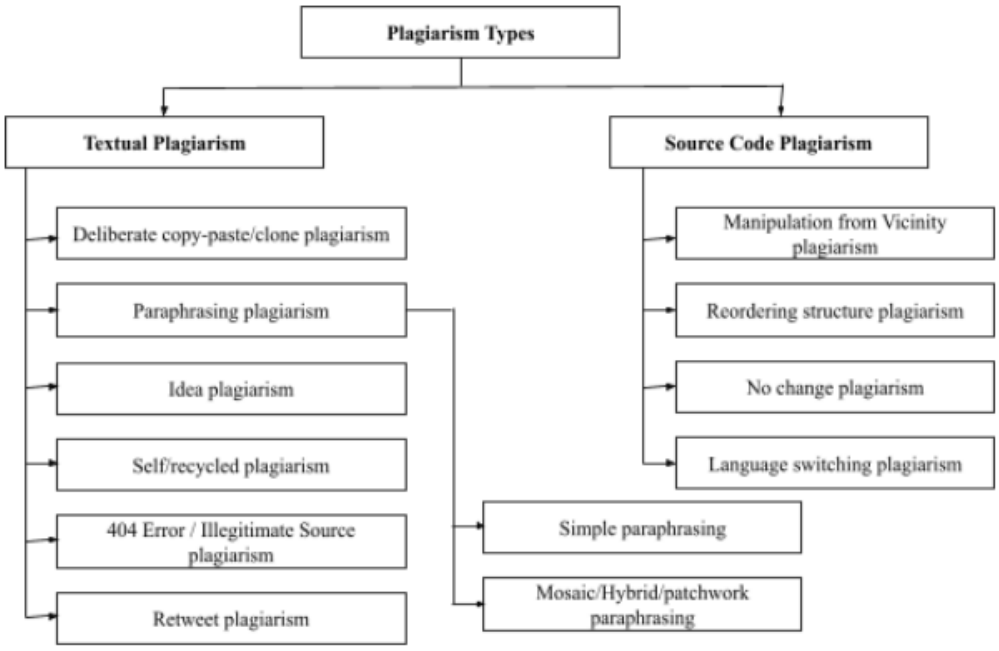


Figure 2.1 - Classification of various forms of plagiarism [ ]

1. **Deliberate copy-paste/clone plagiarism**: This sort of plagiarism happens when someone steals another person's work and publishes it without providing any credit to the original author, who is the owner of that work.[15]

2. **Plagiarism through paraphrasing**:Paraphrasing can result in plagiarism if one changes both the words used and structure of a sentence, and you can classify

plagiarism in paraphrasing into two subcategories. Making it simpler using the technique of mosaic and paraphrasing.

3. **Metaphors**: Applying figurative language to represent other people's work with greater clarity and skill for explaining the facts and truths.

4. **Idea Plagiarism**: The act of copying and presenting another's main aim or solution in multiple variations while falsely representing it as your own effort amounts to committing Idea Plagiarism.[16]

5. **Self Plagiarism**: When someone submits their own previously submitted content in its original form, it falls under the category of self-plagiarism, but publishing one's own work without proper citation still falls under the category of plagiarism.

2.2.2 Plagiarism Analysis Phases

Plagiarism detection involves several analysis phases to identify instances of plagiarism and assess the originality of a given text. Following are the key analysis phases in a plagiarism detector:

Preprocessing:

In this phase, the text is prepared for analysis by applying preprocessing techniques. This may involve removing formatting, converting to lowercase, eliminating punctuation, and handling special characters. Preprocessing helps standardize the text and improve the accuracy of subsequent analysis.

Text Segmentation:

The text is divided into segments like sentances to enable more accurate analysis. It is important for identifying portions of texts that may have been plagiarised.

Feature Extraction:

During feature extraction, relevant characteristics or features are extracted from the text segments. These features can include n-grams (sequences of words or characters), syntactic structures, semantic representations, or statistical properties. The choice of features depends on the detection algorithm used.

External Source Retrieval:

Plagiarism detectors often utilize external sources, such as online repositories, databases, or previously indexed documents. This phase involves retrieving and incorporating these sources for comparison and matching against the analyzed text. Efficient retrieval and indexing mechanisms are employed to enhance the speed and accuracy of the detection process.

Similarity Analysis:

The similarity analysis phase compares the extracted features of the analyzed text with the features of the external sources. Various algorithms, such as string matching, token-based comparison, or advanced machine learning techniques, are applied to measure the similarity between the text segments and potential source materials.

Plagiarism Assessment:

Once similarities are detected, the plagiarism assessment phase evaluates the identified matches in terms of their significance and originality.

Reporting and Presentation:

Finally, the plagiarism detector generates a report presenting the findings of the analysis. The report typically highlights the detected instances of plagiarism, provides details on the matched sources, and offers a comprehensive overview of the overall originality score or plagiarism index.

2.2.3 Challenges of Plagiarism Detection

Even with the use and expansion of digital technology, several problems remain unresolved. The following are some problems that linguists and academics working in this crucial topic must address. The majority of the challenges listed below concern both monolingual and cross-lingual plagiarism detection.

- An important issue is identifying plagiarism in text data and source code that satisfies correctness and completeness.

- Improved accuracy in detecting copied text segments in both intrinsic and extrinsic sources is still a problem.

- It is still difficult to develop extremely accurate cross-lingual plagiarism detection software without using external reference materials.

- It remains a difficult challenge to develop a comprehensive and accurate reference document repository based on the author's footprint.

- It is crucial to have methods that identify similarities not just at the syntactic level but also at the semantic level.

- To increase the accuracy and effectiveness of the system, it is crucial to use a few cutting-edge techniques for semantic plagiarism detection rather than just WordNet. These sophisticated techniques include machine learning, vector-based models, CNN, RNN, and others.[17]

- Named entities (journal/conference name, author name, organisation, place, date/years, etc.) cannot be disregarded when determining whether or not an article is plagiarised because they frequently exist in publications.

- When performing similarity checking, a plagiarism detection system may still include it, in which case the system may display plagiarism to designated entities.

- Powerful Word Sense Disambiguation (WSD) was necessary for the development of powerful Named Entity Recognition (NER). Both still can't be found with greater accuracy. That has an effect while conducting plagiarism detection.

- It is crucial to have a highly accurate machine translation system in order to increase the accuracy of cross-lingual plagiarism detection.

- There is currently no technique that can identify plagiarism in tables, figures, and graphics in a document or article. One of the neglected areas of plagiarism detection is this one.

Chapter 03: SYSTEM DEVELOPMENT

3.1 DESIGN OF THE PROJECT

The plagiarism detector project follows to a structured process of development that breaks down into several significant stages. The main goal is to create a system that can detect plagiarism correctly and start with a plagiarism analysis. This chapter provides an overview of the methodology along with the dataset used, the flowgraph of the project, and the results' evaluation.

3.1.1 Dataset

The Dataset "All the News" is a set of articles along with their details put together in a csv file. The original dataset is created by Andrew Thompson and put together on Kaggle.This dataset contains a comprehensive collection of news articles from various sources. The original dataset contains 2,00,000 articles but due to the project's size, I have cut it down to 20,000 articles, thus also making the application faster. Also there is a server constraint, thats why this large number of dataset cant be put up.

	Unnamed: 0	id	title	publication	author	date	year	month	url	content
0	0	17283	House Republicans Fret About Winning Their Hea...	New York Times	Carl Hulse	2016-12-31	2016.0	12.0	NaN	WASHINGTON — Congressional Republicans have...
1	1	17284	Rift Between Officers and Residents as Killing...	New York Times	Benjamin Mueller and Al Baker	2017-06-19	2017.0	6.0	NaN	After the bullet shells get counted, the blood...
2	2	17285	Tyrus Wong, 'Bambi' Artist Thwarted by Racial ...	New York Times	Margalit Fox	2017-01-06	2017.0	1.0	NaN	When Walt Disney's "Bambi" opened in 1942, cri...

Figure 3.1 - Glimpse of the dataset

The dataset can also be used to train machine learning models for natural language processing (NLP) applications as text categorization, topic modelling, and sentiment

analysis. Researchers can construct and modify algorithms to automatically categorise news stories, detect bias or misinformation, and even build recommendation systems for personalised news distribution now that the complete text of the articles is available. It is possible to discover interesting facts by conducting exploratory data analysis on the dataset. For example, one can show the distribution of articles across various newspapers, find popular subjects or keywords, investigate famous trends in news coverage, and even conduct sentiment analysis to determine the tone of the articles.

The data mostly includes the years 2016 and 2017, with some number of articles from 2015 and probably minor amount from before that.

The features of the dataset are as follows:

1. **sr no.:** Serial number of that entry,

2. **id:** Unique id of the article. Example: 17283, 17301, 17358.

3. **title:** Main title of that article such as House Republicans Fret About Winning Their Health Care Suit, Rift Between Officers and Residents as Killings Persist in South Bronx, Tyrus Wong, 'Bambi' Artist Thwarted by Racial Bias, Dies at 106, Among Deaths in 2016, a Heavy Toll in Pop Music

   .

4. **publication:** Publication house that has published that article. It includes New York Times,, CNN, Business Insider, Breitbar, the Atlantic, Talking Points Memo, Fox News, Buzzfeed News, New York Post, the Guardian, National Review, NPR, Vox, Reuters, and Washington Post.

5. **author:** It includes the author of that article. Some of the featured authors are Carl Hulse, Benjamin Mueller and Al Baker, Margalit Fox, William McDonald, Choe Sang-Hun.

6. **date:** Date of the article when it was published. The format for this is yyyy-mm-dd.

7. **year:** Provides the year when the article was posted. Eg: 2017, 2016

8. **month:** Provides the year when the month was posted. Eg: 2, 5, 12

9. **url:** This column refers to the link of that article

10. **content:** The content column in the dataset represents prvides the whole articles as it was published. Mainly this feature is used to find the plagiarised content.

3.1.2 Flowchart of the Major Project

To develop a plagiarism checker application, we start by taking the input of the CSV file path. Using the pandas library, we read the CSV file and proceed to preprocess and clean the data. Next, we extract the text from the CSV file and generate word embeddings for the text using the average_word_embeddings_komninos technique.

These word embeddings are then stored in Pinecone's index for efficient retrieval. Moving on, we initialize a Flask web application and create a route that allows users to input their text for plagiarism check. The user input undergoes preprocessing and cleaning similar to the CSV data. We generate word embeddings for the user input using average_word_embeddings_komninos and search for similar word embeddings in Pinecone's index.

Retrieving the matching word embeddings and their corresponding text, we calculate the similarity scores between the user input and the matching text. Finally, we display the plagiarism results to the user, providing valuable insights. The process concludes with the end of the plagiarism checker application.

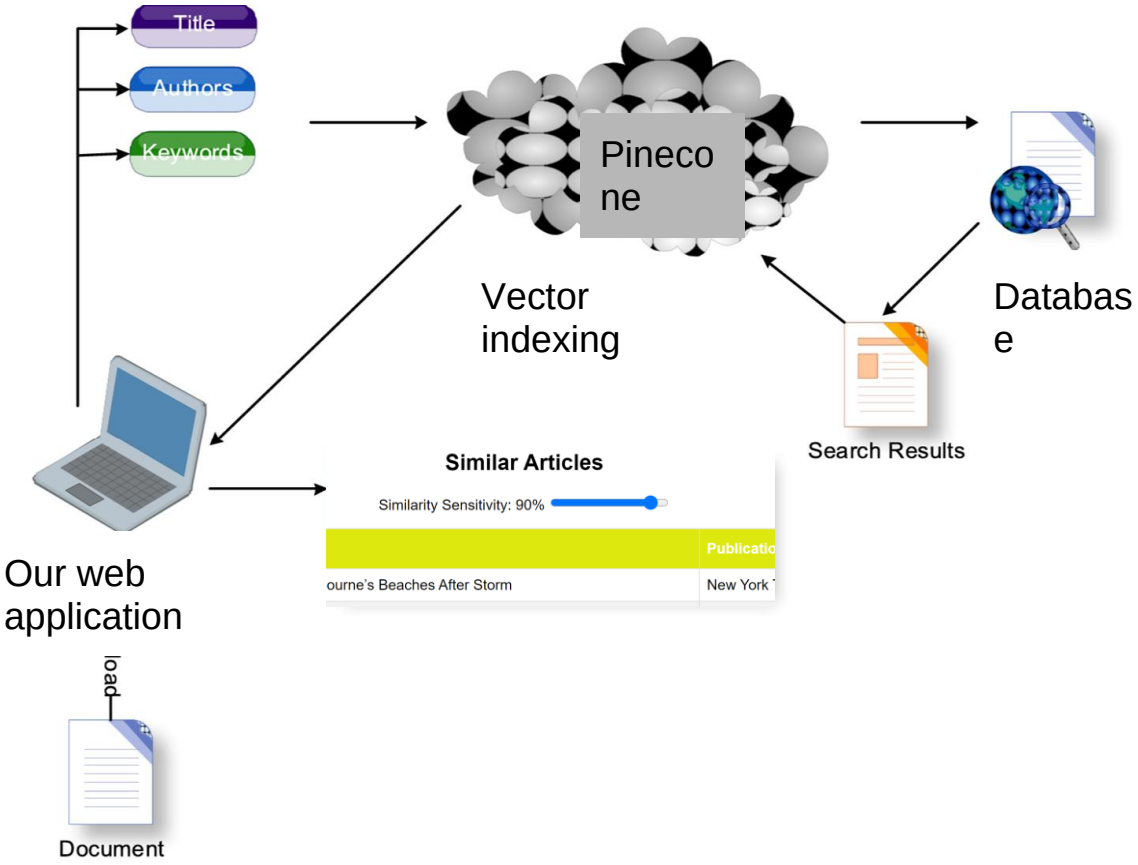


Figure 3.2: Schema showing steps of the project

3.2 TRAINING OF MODELS

3.2.1 Vector Embeddings

Vector embeddings are the main component that enables similarity search. Raw data is transferred from a data store or data stream to an embedding model, which is turned into a vector embedding, and then to the vector search index.

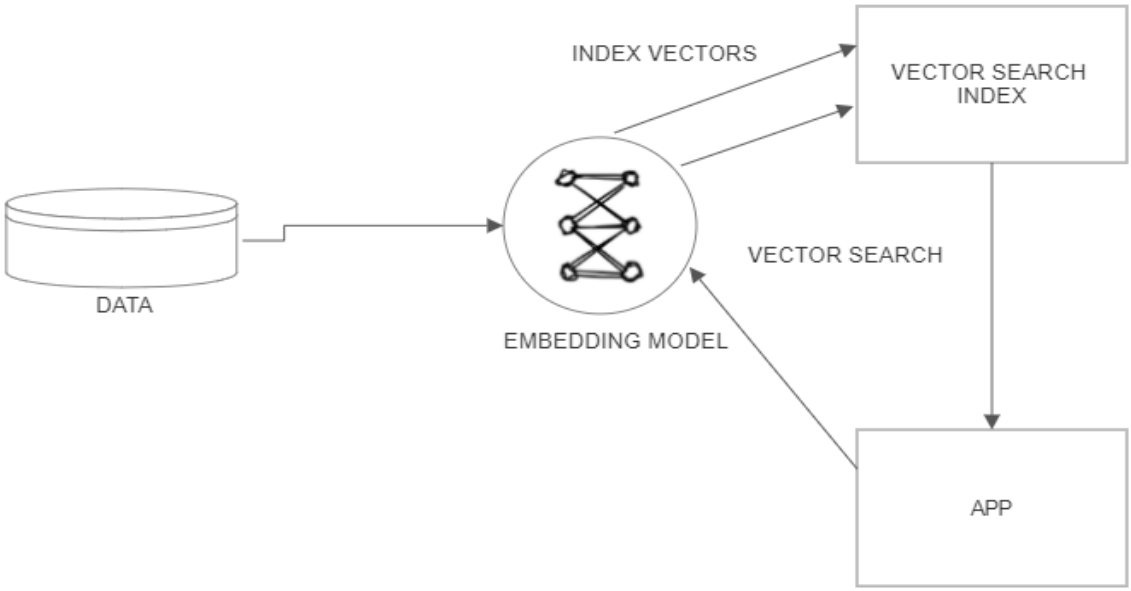


Figure 3.3: Flow of the vectors

Vector embeddings have revolutionized NLP and ML applications.[19] These embeddings represent words dense numerical vectors, capturing their semantic and contextual relationships. Here, we will see the significance of vector embeddings and their impact on different tasks.

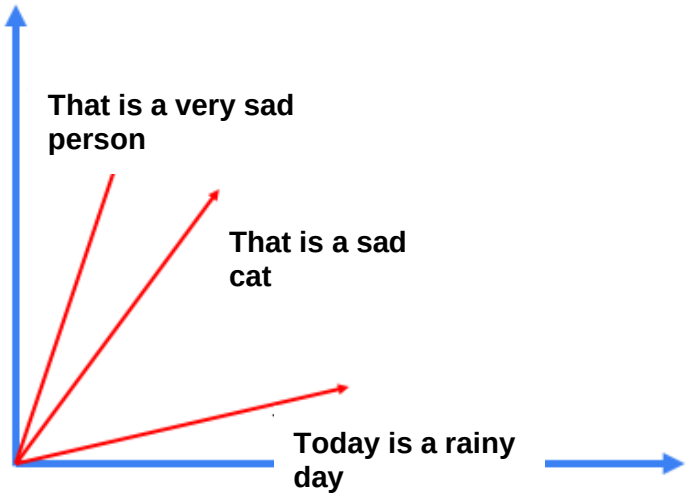
Vector embeddings encode semantic meaning in a continuous space, enabling machines to understand language in a more nuanced way.[20] Traditional approaches, such as one-hot encoding, represent words as sparse binary vectors, lacking the ability to capture semantic relationships. In contrast, vector embeddings employ techniques like Word2Vec, GloVe, or fastText to learn dense vector representations from large text corpora.

Semantics Similarity Search is the process of comparing pieces of text to see which ones have the most comparable meaning. While this may appear simple to the average person, languages are

highly complex. Many Natural Language Processing researchers have been interested in distilling unstructured text data into a format that a Machine Learning model can understand.[21]

Let’s take the following sentences:

- “That is a sad cat”
- “That is a very sad person”
- “Today is a rainy day”



Graph 3.1: Simplified plot of vector embeddings made into 2 dimensions

Now suppose that we can want to compare sentences to "That is a sad perosn". Initially, we create vetor embeddings for query sentance.

After that, we got to compare the distance between oour quey vector embedding and dataset's vector embedding. There are so many ways to calculate the distance between these vectors. Each has their own pros and cons when it comes to semantic search.

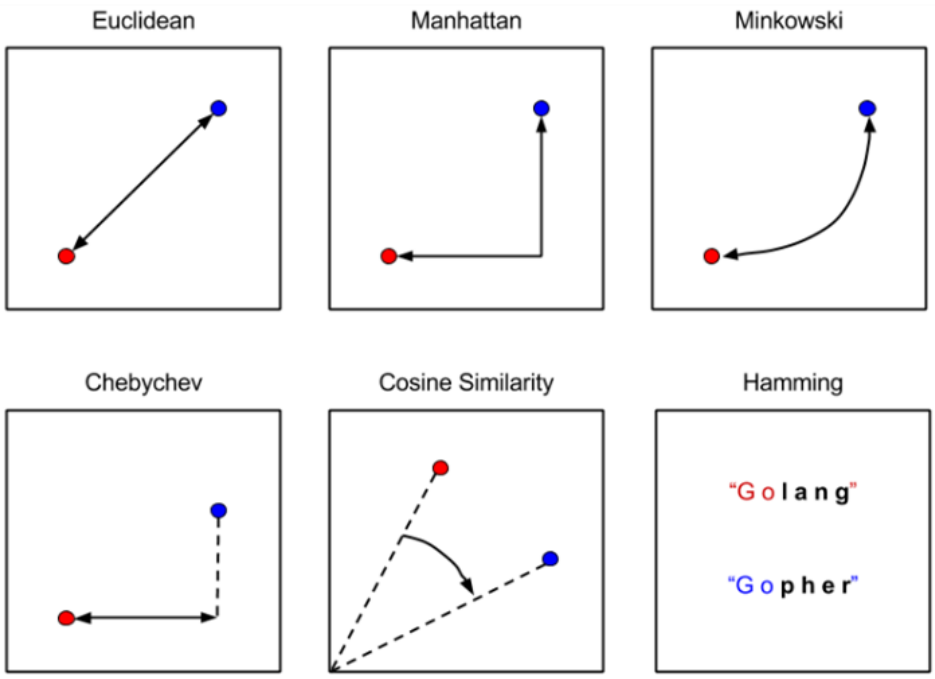


Figure 3.4: Distance metrics that are used in calculating a vector similarity

For example, now we will use the cos similarity which will measure the distance between inner product space of these two vectors

$$\cos(\mathbf{t}, \mathbf{e}) = \frac{\mathbf{te}}{\|\mathbf{t}\|\|\mathbf{e}\|} = \frac{\sum_{i=1}^{n} \mathbf{t}_i \mathbf{e}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{t}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{e}_i)^2}}$$

Figure 3.5: Cosine similarity formula

t = Vector 1

e = Vector 2

Now, by running this calculation between query vector and other three vectors in this plot above and then we can determine how same the sentences are to each other.

$$S_{\cos}(A,B) = \frac{\sum_{i=1}^{n}[\alpha_i^2 \times F_i(A) \times F_i(B)]}{\sqrt{\sum_{i=1}^{n}[\alpha_i^2 \times F_i^2(A)] \times \sum_{i=1}^{n}[\alpha_i^2 \times F_i^2(B)]}}$$

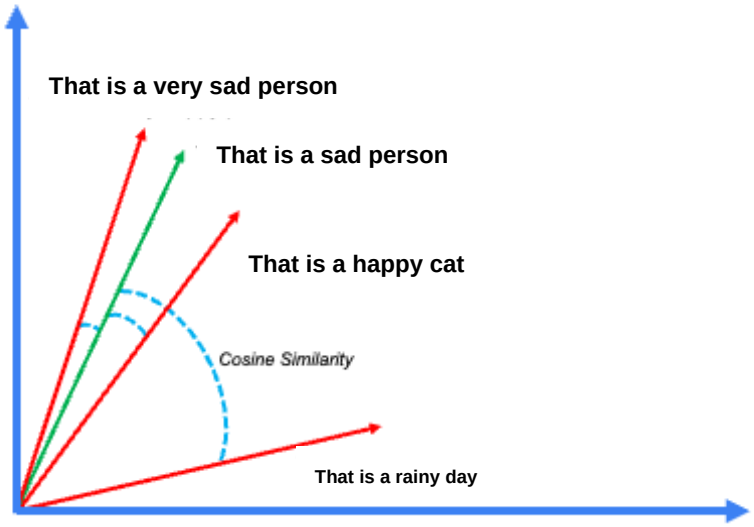
Figure 3.6: Weighted Cosine similarity formula

t = Vector 1

αi – word weight vector;

Fi(A) frequency of the ith word in document A

Fi(B) frequency of the ith word in document B



Graph 3.2: 2-D vector plot showing cosine similarity between vector embeddings that are created from our earlier formed senatnces

These embeddings have proven invaluable across a wide range of applications. In natural language processing, they enhance tasks such as sentiment analysis, named entity recognition, and text classification. By leveraging vector embeddings, models can capture subtle contextual cues and generalize better to unseen data.

Moreover, vector embeddings excel in capturing semantic similarity and analogy relationships between words. For instance, with word embeddings, it becomes possible to perform operations like "king - man + woman = queen," showcasing their ability to capture gender relationships. This makes them useful in tasks such as information retrieval, question answering, and machine translation.[22] How Vector embeddings also facilitate transfer learning. Pretrained embeddings trained on large-scale corpora can be used as a starting point for downstream tasks. This transfer of knowledge empowers models to perform better even with limited labeled data, saving computational resources and time.

Furthermore, vector embeddings enable the creation of semantic search systems and recommendation engines. By representing text and user preferences as embeddings, it becomes feasible to measure similarity and make relevant recommendations. Whether it's finding similar documents, suggesting related products, or personalizing content, vector embeddings play a vital role in enhancing user experiences.[23]

In conclusion, vector embeddings have emerged as a powerful tool in natural language processing and machine learning. Their ability to capture semantic relationships, facilitate transfer learning, and enhance various tasks has propelled advancements in the field.

3.2.2 Sentence Transformers

Sentence transformers are a powerful tool in the field of NLP which offer efficient and effective ways to generate high-quality sentence embedding. Embeddings enable us to encode the semantic meaning of sentences into dense numerical vectors, facilitating various downstream tasks. They leverage pre-trained models like BERT, RoBERTa, or DistilBERT, which are fine-tuned on large-scale text corpora.[24]

By utilizing sentence transformers, we can perform tasks such as sentence similarity, text classification, clustering, and information retrieval. These embeddings capture the contextual information and semantic relationships within sentences, allowing us to measure their similarity or classify them based on their content. The ability to generate sentence embeddings opens up opportunities for building recommendation systems, chatbots, and question-answering systems that rely on understanding the meaning of text.

The Average Word Embeddings Models are used to generate the embeddings stored in the feature store. We need to develop a comparable vector because we want to query new questions and locate the most similar match among questions in the feature store. This means that when we specify a new question, we will use the same model to translate it into a vector embedding.

```
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('average_word_embeddings_komninos')
```

One of the advantages of using sentence transformers is their flexibility and ease of integration. With Python, we can leverage popular libraries like Hugging Face's Transformers or SentenceTransformers, which provide pre-trained models and convenient interfaces for generating embeddings.[25] These tools allow us to process sentences individually or in batches, making it efficient to work with large amounts of text data.

Python's sentence transformers provide a potent means of encoding the context and meaning of phrases into dense numerical vectors. They show the ability to be fine-tuned on specific data, enabling better performance for applications that are task- or domain-specific. This versatility makes a variety of use cases possible, including sentiment analysis, the classification of documents according to a given domain, and the creation of text representations for search queries.

So, sentence transformers empower us to efficiently capture semantic relationships and perform diverse tasks in NLP. They serve as a valuable resource pre-trained models and user-friendly libraries to generate high-quality sentence embeddings.

So, this facilitates advancements in areas like text similarity, classification, clustering, and more. By harnessing the potential of sentence transformers, we can enhance our ability to handle complex language tasks and unlock new possibilities in the field of NLP.[26]

3.2.3 PineCone

Pinecone helps in the process of retrieving similar items based on their numerical representations. It offers a robust infrastructure designed to handle large-scale vector indexes, facilitating rapid and efficient search operations. Developers can use the power of Pinecone to build applications that require advanced similarity search, recommendation systems, personalization, and more.

By this technology, dense vector embeddings can be efficiently stored and retrieved, enabling the retrieval of similar items based on their semantic and contextual relationships[27]. The platform supports various distance metrics, including cosine similarity, providing flexibility and accuracy in similarity calculations. With its scalable and cloud-native architecture, Pinecone simplifies the management of large-scale vector indexes, ensuring high availability and low latency for real-time applications.

Pinecone enables developers to produce reliable and high-performance solutions that harness the power of vector similarity search, whether it's for locating related photos, looking for text documents, or empowering recommendation engines. Developers can unlock the potential for cutting-edge similarity-based functionality and offer top-notch user experiences by incorporating Pinecone into their applications.

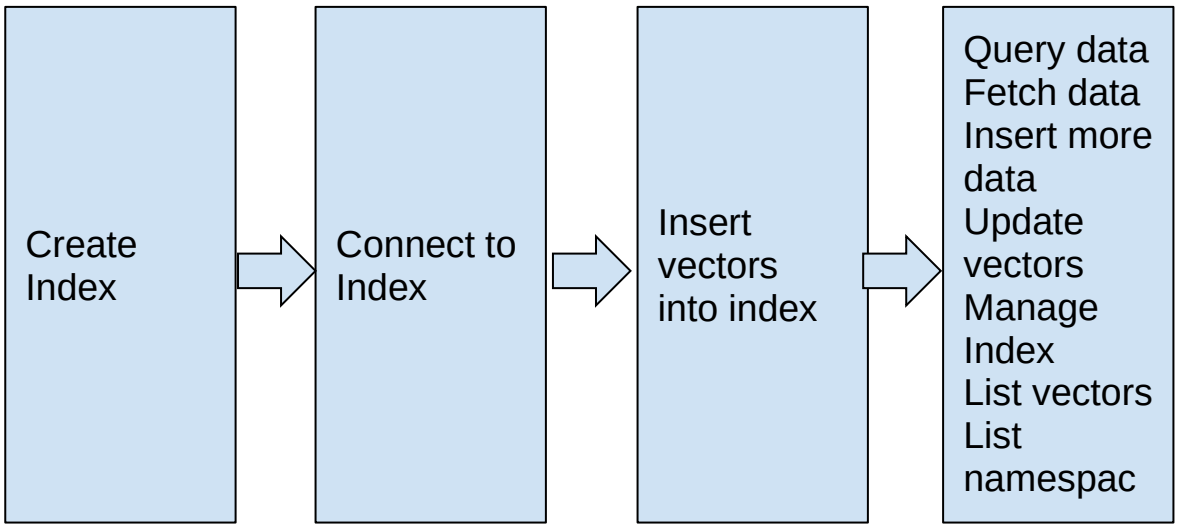


Figure 3.7: Workflow of Pinecone

Here's how Pinecone is used in making a my project:

1. Data Preparation: Collection of articles was collected according to my project. This was then fed into the code and manipulated.

2. Embedding Generation: Transforming each text segment into a numerical representation is important step in this project. This is done by using NLP techniques like word embeddings. And in this project, I have used average_word_embeddings_komninos which is used to convert text into numerical vectors[28]. By representing it in numerical form, effectively capture its semantic meaningand enable easy analysis in plagiarism checker.

3. Indexing with Pinecone: Above generated vector embeddings are indexed and then stored in the Pineone’s vector index.

4. User Input Processing: After the user submits the text for the check, this input text goes under preprocessing. It is cleaned, standardised and then transformed into vector embeddings using same technique,

5. Searching for Similarity: Input’s embedding is then used to search for the pinecone index for similar type of vector embeddings. It performs similarity search and gets the most similar embeddings.

6. Result Evaluation: The fetched embeddings are then mapped back to their corresponding segments and the checker calculates the similarity scores, which helps to identify the potential instances.

7. Displaying Results: The plagiarism checker presents the results to the user, indicating the matched text segments and providing relevant information such as similarity scores.

By integrating Pinecone's vector similarity search capabilities, the plagiarism checker can efficiently compare the user's input against a large corpus of documents to identify potential instances of plagiarism. Pinecone's fast and scalable search engine enhances the speed and accuracy of the plagiarism detection process, enabling efficient handling of large amounts of text data.

Chapter 04: RESULTS

4.1 DISCUSSION

The outcomes of a plagiarism checker are extremely important to the tool's overall performance and value. To comprehend and interpret the findings correctly, a full explanation of the findings is required. Here are some important things to remember when debating a plagiarism checker's findings.

- Plagiarism Detection Accuracy: It is important to show the accuracy of the model by showcasing TP and TN.
- Similarity Scores: Plagiarism checkers usually provide similarity scores or to show the level of similarity between the analyzed content and potential sources of plagiarism.
- Limitations and False Positives: FP(content that was falsely highlighted as plagiarised content) need to be reduced down to minimum. This can occur due to the occurrence of common phrases.
- Source Identification: We need to identify the original source of the plagiarised content as this would give us the clear picture from where the content was copied

4.2 EVALUATION

Evaluation is one of the crucial steps in any project that involves assessing the performance of a trained model. The purpose of this step is to see how well a model is making predictions on our required data and self us achieve the objective.

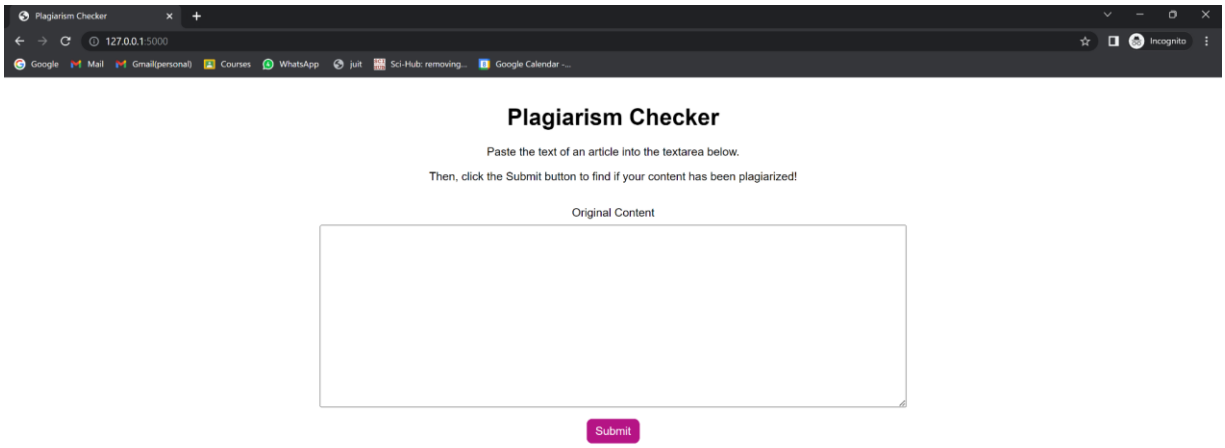


Figure 4.1: UI of the project

Given above is the UI of the plagiarism checker. All the relevant information and instructions are provided on the webpage only, so it makes the user understand the directions to perform the task.

The text box is placed where one can put their required text and click on the the submit button. This would result in the output of articles from where the content might have been copied.

Rest is given below.

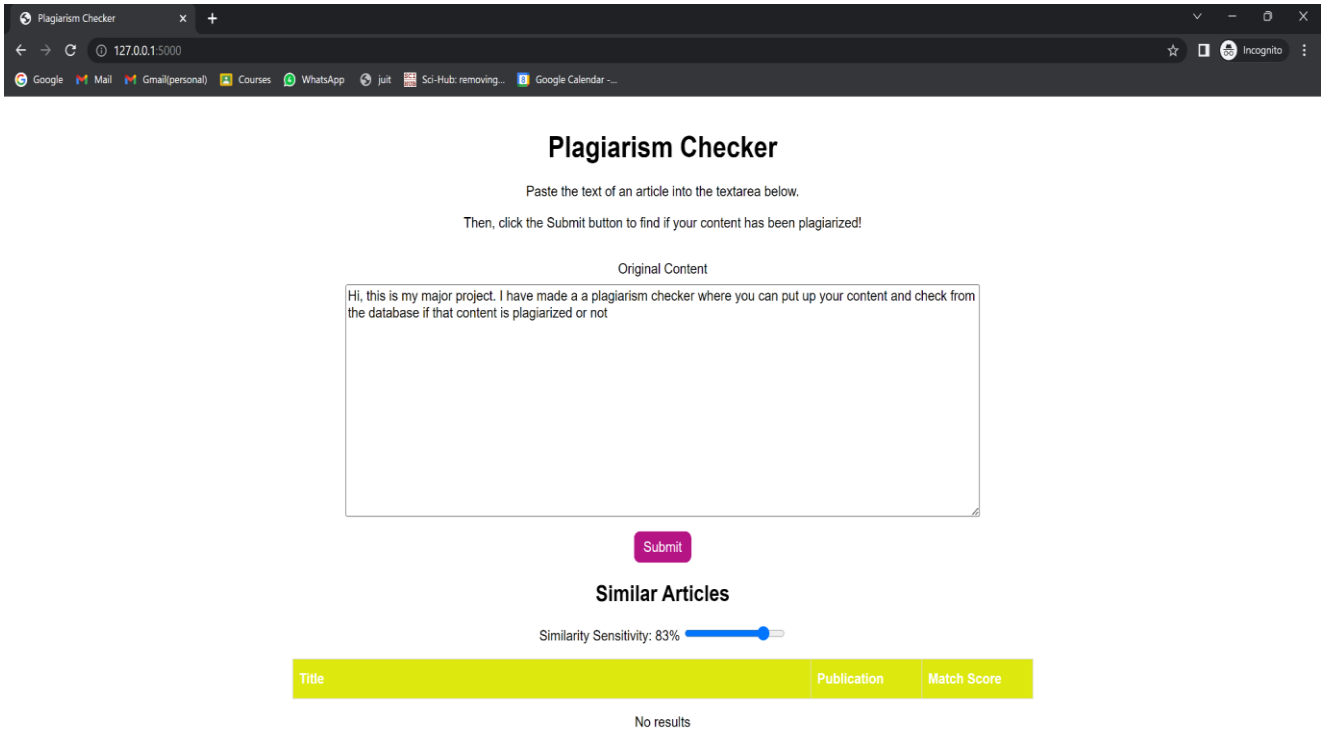


Figure 4.2: Text placed in the textbox

I have written down some random text which has no relation to the given database. So it gives me a No Results output because the content have not been plagiarised from anywhere.

Now after this, we will take a headline from ‘sr no. 15’ and paste it down in the textbox and click the submit button.

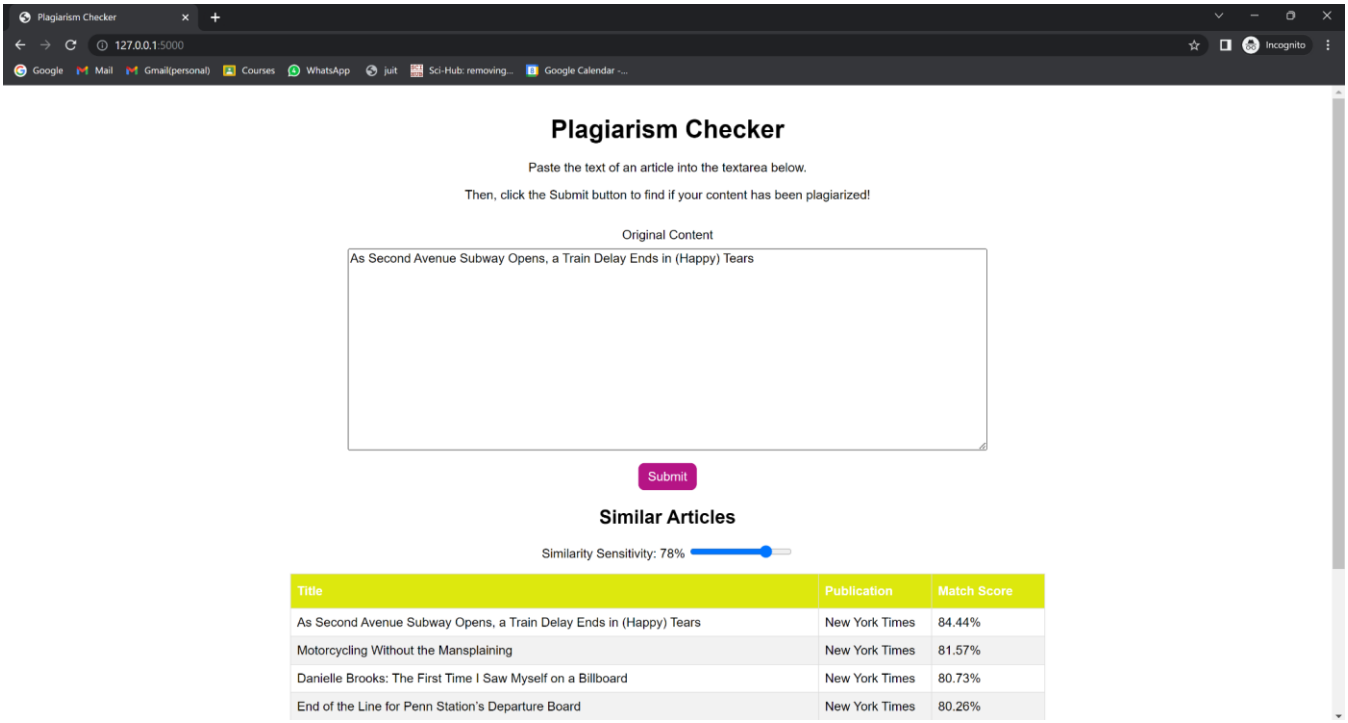


Figure 4.3: Title placed in the textbox

As we took article no 15's title and pasted it down the textbox, we can see the required headline is shown in the result which gives us the clear information about the successful working of the project.

Also the Match Score is also shown in the column, that gives us the idea of how similar the two texts are.

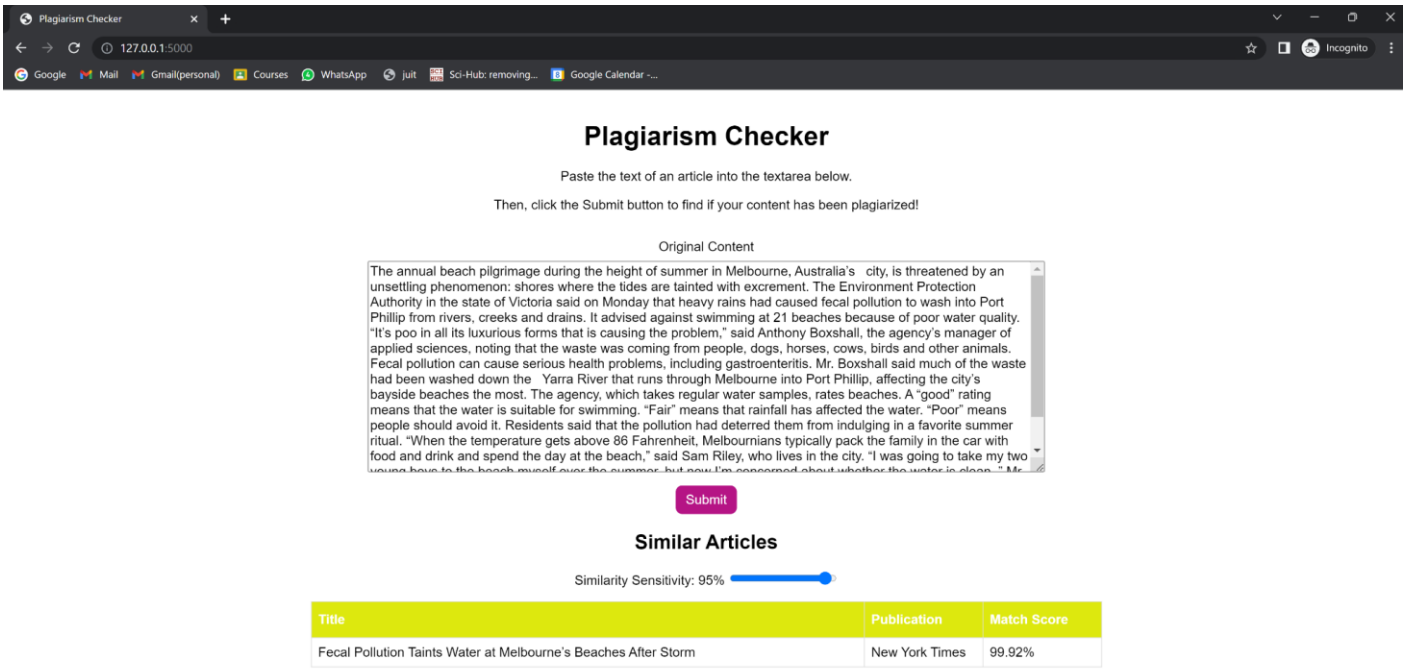


Figure 4.4: Full text placed

Now, we take on article number 19's full text and paste it in the textbox. As we can see in the above output, It can correctly identified the article from where the content was copied and given us the output.

It also gave us a match score of ~100% as the whole content was plagiarised.

All of this information resonates with the proper and successful working of my major project.

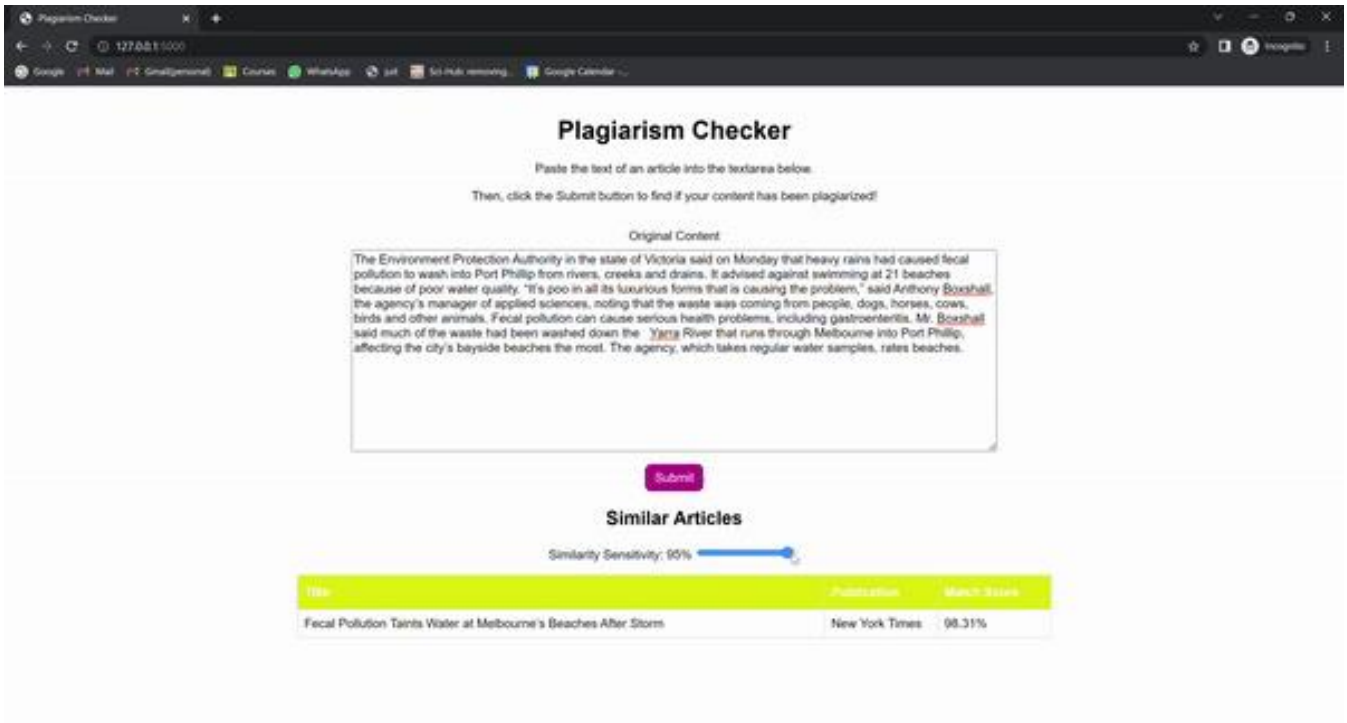


Figure 4.5: Working of the Similarity Senstivity bar

As we can see in the above figure, by adjusting the dynamic similarity sensitivity bar, we are able to get filtered results based on the match score.

Chapter 05: CONCLUSIONS

5.1 CONCLUSION

In conclusion, this project provided an opportunity to get hands-on experience in a working plagiarism checker. The project was made possible by the use of publicly accessible datasets using articles. This further helped to create a plagiarism checker which outputs the similarity index along with 3 columns mainly Title, Publication and Match score.

Therefore, the development of the plagiarism checker project has proven to be a significant endeavor in addressing the growing concerns surrounding academic integrity and originality of written content. Through the utilization of advanced technologies such as pandas for data handling, Flask for web framework, average_word_embeddings_komninos for text matching, and Pinecone for efficient vector storage and retrieval, the plagiarism checker offers a powerful and comprehensive solution.

By implementing a multi-step process that involves reading and preprocessing data from a CSV file, generating word embeddings for both the source text and user input, and searching for similarities in Pinecone's index, the plagiarism checker efficiently identifies potential instances of plagiarism. The system calculates similarity scores and presents them to the user, providing valuable insights into the level of originality and potential content overlap.

Furthermore, the integration of Flask as the web framework ensures a user-friendly interface, allowing users to conveniently input their text for analysis. The combination of sophisticated text matching algorithms and the use of vector embeddings enables the plagiarism checker to deliver accurate and reliable results.

Overall, the plagiarism checker project demonstrates the effectiveness and importance of leveraging advanced technologies in combating plagiarism and the output results demonstrate the effectiveness of sentence transformers when it comes to plagiarism checker and its working. By offering a robust and efficient solution for plagiarism detection, this project serves as a valuable tool for educators, researchers, and content creators alike, promoting academic integrity and fostering a culture of originality in written works.

5.2 APPLICATION OF MAJOR PROJECT

There are several practical applications that can be used, some of these are as mentioned below:

1. Academic Institutions: Plagiarism checkers are extensively used in educational institutions to ensure academic integrity by detecting instances of plagiarism in student assignments, research papers, and theses.
2. Industry of Publishing: Before publishing, plagiarism checkers assist publishers in ensuring the originality of manuscripts submitted and locating any potential instances of plagiarism.
3. Content Creation: Content creators, such as bloggers, journalists, and writers, can utilize plagiarism checkers to verify the originality of their work and avoid unintentional plagiarism.
4. Research Institutions: Plagiarism detection tools are crucial in research institutions to uphold ethical research practices and identify any instances of plagiarism in scholarly articles and research papers.
5. Website Content: Website owners and administrators can use plagiarism checkers to validate the originality of website content, ensuring it is not copied from other sources without proper attribution.
6. Legal Document Verification: Plagiarism checkers assist legal professionals in verifying the originality of legal documents, contracts, and patent applications.

7. Online Learning Platforms: Plagiarism checkers are integrated into online learning platforms to prevent cheating and ensure the originality of student submissions.
8. Grant Applications: Plagiarism checkers are used during the evaluation of grant applications to identify any instances of copied content and maintain the fairness and integrity of the selection process.

5.3 LIMITATIONS

- The biggest limitation of the project was the lack of vector embedding platform like pinecone.
- Pinecone allows only upto 1000 nonzero vector uploads.
- Another limitation of pinecone is that only one session can run in the free trail plan, so one needs to purchase a paid plan to run sessions,
- Language Dependency: The plagiarism checker's effectiveness may vary depending on the language of the text. Some algorithms or models used for text matching and similarity calculation may perform differently across different languages, potentially impacting the accuracy of plagiarism detection.
- Limited Source Database: The plagiarism checker's performance heavily relies on the comprehensiveness and quality of the source database it uses for comparison. If the database is limited in terms of coverage or contains outdated or incomplete sources, it may result in false negatives or limited detection capability.
- Complex Plagiarism Techniques: Plagiarism can take various forms, including paraphrasing, translation, or rewording. While the checker may be effective in detecting verbatim copying, it may face challenges in identifying more sophisticated forms of plagiarism that involve substantial modifications to the original text.
- With the introduction of AI text based models like ChatGPT, one needs to improvise and introduce a more complex AI-based plagiarism detector, which can further detect content that has been generated through AI.

5.4 FUTURE WORK

The future of the plagiarism checker project holds various possibilities for further development and enhancement. Here are some potential aspects to consider:

- Improved Accuracy: The research can concentrate on improving the methods and algorithms used to detect plagiarism. To improve the precision of plagiarism detection and lower false positives or false negatives, this may need using advanced natural language processing (NLP) models, machine learning algorithms, or deep learning approaches.
- AI detector: With the advancement in technology, AI tools have been introduced and we could improvise and develop new algorithms which can detect the AI pattern, thus checking AI plagiarism.
- Multilingual Support:As plagiarism occurs in a variety of linguistic contexts, expanding the checker's language support capabilities would be beneficial. Plagiarism can be found in a wider range of content by utilising language-specific models and resources.
- Enhanced Text Preprocessing: To deal with noise, variances, and inconsistencies in the incoming text, the project can investigate advanced text preprocessing approaches. Improved handling of punctuation, capitalization, spelling mistakes, and other linguistic idiosyncrasies that may have an impact on the efficacy of plagiarism detection may be required.
- Integration with Online Platforms: Users can enjoy a seamless experience by integrating the plagiarism detector with well-known online platforms like learning management systems (LMS), content management systems (CMS), or document collaboration tools. Real-time checks for plagiarism and automatic content duplication detection within the platform might be made possible by this integration.

- Integration with External APIs:Integrating the plagiarism checker with external APIs and services can expand its capabilities. For example, integrating with academic databases, digital libraries, or online sources can provide a more comprehensive comparison for detecting plagiarism.

By focusing on these aspects, the plagiarism checker project can continue to evolve, improve its accuracy, and address the ever-changing challenges associated with plagiarism detection.

REFERENCES

[1] A. Barrón-Cedeño et al., "A Comparison of Plagiarism Detection Tools and Techniques," in IEEE Transactions on Knowledge and Data Engineering, vol. 31, no. 8, pp. 1500-1513, Aug. 2019.

[2] R. Kumar et al., "Detecting Plagiarism in Natural Language Texts," in IEEE Transactions on Computers, vol. 66, no. 5, pp. 869-882, May 2017.

[3] P. Singh and S. Singh, "An Integrated Plagiarism Detection Framework Using N-Gram Technique," in Proceedings of the IEEE International Conference on Advances in Computing, Communications and Informatics, 2018, pp. 120-125.

[4] S. M. Rehman et al., "Plagiarism Detection Techniques: A Comparative Study," in IEEE Access, vol. 8, pp. 84056-84067, 2020.

[5] M. Barrios and J. L. Ramírez, "Effective Plagiarism Detection Using Latent Semantic Analysis," in IEEE Latin America Transactions, vol. 16, no. 6, pp. 1587-1594, June 2018.

[6] A. Hussain et al., "An Enhanced Plagiarism Detection Algorithm Using Fuzzy Logic for Language," in IEEE Access, vol. 9, pp. 27189-27198, 2021.

[7] N. Selvi and S. Abirami, "A Novel Approach for Plagiarism Detection Using Hybrid Optimization Algorithm," in Proceedings of the IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials, 2019, pp. 286-291.

[8] Y. Wang et al., "Plagiarism Detection Based on Feature Set Reduction and Random Forest Classifier," in Proceedings of the IEEE International Conference on Cyber Security and Cloud Computing, 2020, pp. 116-121.

[9] S. Mishra et al., "Efficient Plagiarism Detection for Source Code in Open-Source Projects," in IEEE Transactions on Software Engineering, vol. 47, no. 7, pp. 1432-1448, July 2021.

[10] M. P. Bressan et al., "Cross-Language Plagiarism Detection Using Multilingual Document Similarity Measures," in IEEE Transactions on Knowledge and Data Engineering, vol. 33, no. 7, pp. 2784-2797, July 2021.

[11] Y. P. Ma et al., "Plagiarism Detection Based on Lexical and Syntactic Features Using Convolutional Neural Network," in Proceedings of the IEEE International Conference on Computational Science and Computational Intelligence, 2019, pp. 698-703.

[12] S. Shukla et al., "Plagiarism Detection in Scientific Literature: A Deep Learning Approach," in Proceedings of the IEEE International Conference on Computational Intelligence in Data Science, 2020, pp. 1-6.

[13] N. Tahir et al., "A Deep Learning Model for Plagiarism Detection Using Convolutional Neural Network," in Proceedings of the IEEE International Conference on Innovations in Engineering and Technology, 2021, pp. 1-6.

[14] M. Al-Bared et al., "Plagiarism Detection Model Based on Text Embedding Technique Using Recurrent Neural Networks," in IEEE Access, vol. 9, pp.

[15] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, 2019, pp. 3973-3983.

[16] M. Cer et al., "Efficient Sentence Embedding Generation for Unsupervised Semantic Textual Similarity," in Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 683-693.

[17] X. Zhang et al., "SBERT-WK: A Sentence Embedding Method by Dissecting BERT-based Word Models," in Proceedings of the 2020 IEEE International Conference on Big Data, 2020, pp. 2806-2813.

[18] D. Cer et al., "Universal Sentence Encoder," in Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 2569-2579.

[19] N. Reimers et al., "Sentence Transformers: Multilingual Sentence Embeddings using BERT," in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2020, pp. 1-6.

[20] S. Ruder et al., "Sentence Embeddings in NLP: Methods, Models, and Applications," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 9, pp. 3016-3030, Sept. 2021.

[21] A. S. Kolhatkar and A. Garg, "BERTRank: Leveraging BERT for Sentiment Analysis," in Proceedings of the 2020 IEEE International Conference on Big Data, 2020, pp. 3733-3738.

[22] J. Wang et al., "An Improved Text Classification Method Based on Sentence Transformers," in Proceedings of the 2021 IEEE International Conference on Artificial Intelligence and Big Data, 2021, pp. 273-278.

[23] Z. Yang et al., "Sentence Transformers for Multi-Document Summarization," in Proceedings of the 2021 IEEE International Conference on Information and Computer Technologies, 2021, pp. 243-248.

[24] K. Jindal and S. Baghel, "Deep Learning based Feature Extraction for Sentiment Classification," in Proceedings of the 2020 IEEE International Conference on Computing, Electronics & Communications Engineering, 2020, pp. 1-5.

[25] H. Li et al., "Hierarchical Attention Based Sentence Embedding for Sentiment Classification," in Proceedings of the 2020 IEEE International Conference on Big Data, 2020, pp. 3807-3812.

[26] W. Wu et al., "A Novel Method for Sentence Classification based on BERT and Capsule Networks," in Proceedings of the 2021 IEEE International Conference on Big Data, 2021, pp. 3232-3239.

[27] C. Wu et al., "Graph-Embedded BERT for Aspect-based Sentiment Analysis," in Proceedings of the 2020 IEEE International Conference on Data Mining, 2020, pp. 1135-1140.

[28] R. Zhang et al., "Sentence Classification based on Hierarchical Attention with BERT," in Proceedings of the 2020 IEEE International Conference on Data Mining, 2020, pp. 994-999.