**ACKNOWLEDGEMENT**

We express our warm and profound sense of gratitude to all the eminent faculties who inspired, guided and supported us in accomplishing our project work.

We deeply indebted to **MR. C K SRINIVAS**, Department of CSE , our guide on this project & coordinator **MR. C K SRINIVAS, MR. VENKATESHWAR. A, MR. PANI RAMA PRASAD** on this project, for consistently providing us with the required guidance to help us in the timely and successful completion of this project. In spite of his extremely busy schedules in Department, he was always available to share with us his deep insights, wide knowledge and extensive experience.

We are much obliged to our honourable Head of the Department **Dr. R. N. Kulkarni** for understanding the capability in us to complete the project successfully and also, we are thankful to him for encouraging us and being motivation and inspiration for us.

We extend our warm thanks to our Principal **Dr. Yadavalli Basavaraj** for giving us the moral support and providing us all kinds of facilities.

We also thank all the faculty members of our department for being with us when needed.

Last but not the least we also thank our beloved friends who were there with us all the time supporting, providing assistance and giving us enough strength to successfully complete our project and also proving the popular saying, “A Friend in need is a Friend indeed” to be true.

**D PRIYANKA(3BR19CS041)**

**MUNDERGI SANDEEP(3BR19CS091)**

**THIMMALAPURAM SRUJANA(3BR19CS167)**

**TUSHAR SANTOSH TELKAR(3BR19CS168)**

**ABSTRACT**

Plagiarism is an expansive obstacle in various fields like education sector, business technology, fashion world, etc. The writers copy the readily available material and data and present it as their own work without providing required citations. Due to which, industries like the education system, fashion world newspaper publishers and lot more industries fall under immense intimidating remark. Although there are several tools available in the market that solve the problem of plagiarism using different approaches and features. The existing plagiarism detection tools available in the market compare plagiarism only when the input document contains the text format. But when the document contains both text and images it’s very difficult to compare images. In this paper we are proposing an exhaustive searching technique which searches the text, images and even text which is embedded in the image. The proposed method uses .docx, .pdf and different image formats. The tool proposed here in this project verifies the input document with the content available on the websites and gives the results with respect to similarity.

**I**

**ACKNOWLEDGEMENT**

We express our warm and profound sense of gratitude to all the eminent faculties who inspired, guided, and supported us in accomplishing our project work.

We deeply indebted to **Dr. R. N. Kulkarni**, Department of CSE, our guide on this project & coordinator **Mr. C. K. SRINIVAS, Dr. VENKATESHWAR. A, MR. PANI RAMA PRASAD** on this project, for consistently providing us with the required guidance to help us in the timely and successful completion of this project. Despite his extremely busy schedules in Department, They were always available to share with us his deep insights, wide knowledge, and extensive experience.

We are much obliged to our honourable Head of Department **Dr. R. N. Kulkarni** for understanding the capability in us to complete the project successfully and we are thankful to him for encouraging us and being motivation and inspiration for us.

We extend our warm thanks to our Principal **Dr. Yadavalli Basavaraj** for giving us moral support and providing us with all kinds of facilities.

We also thank all the faculty members of our department for being with us when needed.

Last but not the least we also thank our beloved friends who were there with us all the time supporting, helping, and giving us enough strength to successfully complete our project and proving the popular saying, “A Friend in need is a Friend indeed” to be true.

|  |  |
| --- | --- |
| CHITRIKI GANESH | 3BR19CS040 |
| DILIP KUMAR B K | 3BR19CS044 |
| HARSHITHA B | 3BR19CS019 |
| A PRATHAP REDDY | 3BR19CS001 |

**II**

**TABLE OF CONTENTS**

**III**

|  |  |  |
| --- | --- | --- |
| **CHAPTER No** | **CHAPTER NAME** | **PAGE No** |
|  | ABSTRACT | I |
|  | ACKNOWLEDGEMENT | II |
|  | TABLE OF CONTENTS | III |
| 1 | INTRODUCTION  1.1 OVERVIEW  1.2 EXISTING PLAGIARISM TOOLS  1.3 PROPOSED SYSTEM  1.4 PROBLEM STATEMENT AND SCOPE OF THE PROJECT  1.5 VISION, MISSION, AND OBJECTIVES | 1 |
| 2 | LITERATURE SURVEY | 2 |
| 3 | REQUIREMENTS  3.1 FUNCTIONAL REQUIREMENTS  3.2 NON-FUNCTIONAL REQUIREMENTS  3.3 HARDWARE AND SOFTWARE REQUIREMENTS | 4 |
| 4 | SYSTEM DESIGN  4.1 FLOW CHART  4.2 SEQUENCE DIAGRAM  4.3 DATA FLOW DIAGRAM | 4 |
| 5 | IMPLEMENTATION  5.1 SOFTWARE REQUIREMENTS  5.2 MODULES DESCRIPTION | 4 |
| 6 | IMPLEMENTATION  6.1 UNIT TESTING  6.2 MODULE LEVEL TESTING  6.3 INTEGRATION AND SYSTEM TESTING | 5 |
| 7 | RESULTS AND DISCUSSIONS | 5 |
| 8 | CONCLUSION AND FUTURE SCOPE  8.1 CONCLUSION  8.2 FUTURE SCOPE | 6 |
|  | REFERENCES | 7 |

**IV**

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Table Name** | **Page No.** |
| 4.1 | INTERNAL EVALUATION OF TF-IDF MATRIX IN PYTHON. | 12 |
| 6.1 | UNIT TESTING | 20 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Figure Name** | **Page No.** |
| 4.1 | FLOW CHART | 9 |
| 4.2 | SEQUENECE DIAGRAM | 13 |
| 4.3 | DATA FLOW DIAGRAM | 14 |
| 7.1 | HOME PAGE | 21 |
| 7.2 | UI FOR TEXT INPUT | 21 |
| 7.3 | RESULTS OBTAINED FOR TEXT INPUT | 22 |
| 7.4 | UI FOR PDF FILE INPUT | 22 |
| 7.5 | RESULTS OBTAINED FOR PDF FILE INPUT | 23 |
| 7.6 | UI FOR IMAGE FILE INPUT | 23 |
| 7.7 | RESULTS OBTAINED FOR IMAGE FILE INPUT | 24 |

**V**

**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW**

Plagiarism is one of the growing global problems experienced by publishers, research, and educational institutions which is generally defined to be literary theft. i.e., holding the models, documents, canons, pictures, etc. of another person and presenting them as their own work. This is expressed as plagiarism. This proves an act of dishonesty in academics and literature and hence it must be prevented. Plagiarism can harm a person's reputation, result in legal action, and even result in the cancellation of credentials in both academic and professional settings. Therefore, it is now more crucial than ever to identify and avoid plagiarism, especially in the age of readily available online content.

**1.2 EXISTING PLAGIARISM TOOLS**

Plagiarism detection systems offer a range of combined features to identify instances of plagiarism and promote originality in writing. These systems, such as Turnitin, Grammarly, Copyscape, Plagscan, Urkund, and iThenticate, compare submitted documents against extensive databases of academic content, web pages, and published works. They generate detailed reports with similarity scores, highlighting potential instances of plagiarism and providing suggestions for proper citation and paraphrasing. Some systems also offer additional features like citation checking, source code plagiarism detection, and integration with learning management systems.

**Limitation of Existing System:**

Existing System does not allow image inputs. The existing systems are capable of detecting plagiarism in pdf and docx formats only.

**1.3 PROPOSED SYSTEM**

In this paper, a plagiarism detection tool is proposed that utilizes a combination of Natural Language Processing (NLP) text preprocessing, Term Frequency Inverse Document Frequency (TF-IDF), Vector Space Model (VSM), and Cosine Similarity techniques to accurately detect instances of plagiarism within a document by comparing it against online sources.

The tool first preprocesses the document's text by removing stop words, lemmatization, and tokenizing the text using an NLP model.

The tool extracts relevant text from online sources and then generates a TF-IDF matrix for the document, which represents the frequency of each word in the document relative to the frequency of that word across all the results obtained from online. processes to generate a TF-IDF matrix.

The tool then uses VSM to convert the document's TF - IDF matrix into a vector in a high-dimensional space, where each dimension represents a different word in the document. The tool also converts the online source's TF-IDF matrix into vectors for comparison purposes. The tool then computes the cosine similarity between the two sets of vectors, which measures the degree of similarity between the document and online sources based on the angle between their vectors.

The tool marks the source link with a red color whose cosine similarity score is above a certain threshold. The proposed tool offers an effective and reliable solution for detecting plagiarism within a document against online sources, Therefore the detection of plagiarism inside a document is becoming increasingly important in many fields and industries where originality is a critical factor.

**Benefits in the proposed system:**

* Checks plagiarism in images containing text in them.
* The proposed system gives the exact source location (Uri)

**1.4 PROBLEM STATEMENT AND SCOPE OF THE PROJECT**

**Problem Statement:**

To design, develop and implement a software system to detect plagiarism in an input document.

**Scope of the Project:**

The proposed system can detect plagiarism for digital text images, however detection of plagiarism with respect to handwritten documents is inaccurate.

**1.5 VISION, MISSION, AND OBJECTIVES**

**Vision**

To develop a system that detects plagiarized activity and thus upvote the process of learning and Genuity of an individual.

**Mission**

* To read the input from the user and validate whether input is appropriate.
* To feed the content of the source document to the internet and obtain matching results.
* Populate similarity index between the source and the results obtained online.

**Objectives**

1. To check the input document for the grammar, spelling, and appropriate sentences.
2. To read and store the text extracted from the input document into a repository.
3. To generate similarity index report for the input document.
4. Display or highlight the location in the document about plagiarism.

**CHAPTER 2**

**LITERATURE SURVEY**

In this paper the author has used a vector space model called Cosine similarity. As is a known thing, evaluating the similarity by just considering the raw text is not accurate when the suspicious document contains text that is rephrased. A vector space is just the representation of raw text as vectors. Cosine similarity is good when compared to other methods. Because it gives more importance to the weight of the terms that are infrequent in the dataset [1].

This research compares the unigram, bigram, and trigram of the vector space model with cosine similarity measure to identify an intra-corpal plagiarism detection tool that is ideal for text-based assignments. The use of the unigram, bigram, and trigram vectors was tested using a manually evaluated, labeled dataset. Trigram vector produces better results with the tagged data even though it takes relatively more effort. Additionally, the trigram sequence matching method using the Jaccard measure is contrasted with the chosen trigram vector space model using the cosine similarity metric. The cosine similarity score has slightly higher values than the others in the results. Due to its emphasis on giving terms that do not appear frequently in the dataset more weight, the trigram technique's cosine similarity metric is preferred over the alternative [1].

This paper discusses a Java source code plagiarism detection tool named CopyPoppy, programming plagiarism issues are addressed in this research. It's significant to remember that some pupils may comprehend code solutions and utilize them as a basis for creating their own code. Since they are just using another person's source code as inspiration for their own finished code, these types of situations may not always be seen as instances of plagiarism. Therefore, numerous factors must be considered when determining whether a source code is plagiarized. To discern between what is deemed plagiarism and what is not, CopyPoppy was created [2].

In this paper, the authors have discussed that Plagiarism is the act of using someone else's words or ideas without giving the original author due credit. The tool proposed in this paper checks plagiarism in NCERT textbooks. NCERT textbooks are bound by copyrights and usage guidelines. Although it is possible to download copies of these textbooks and use them as references or as textbooks, publishing these copies is undoubtedly prohibited. To avoid being accused of plagiarism, you must correctly cite these sources throughout your piece of writing [3].

This paper describes a text similarity detection approach for electricity customer service work orders based on the TFIDF algorithm and cosine similarity. This method entails gathering and preprocessing text data, extracting and weighting relevant keywords with TFIDF, and estimating text similarity with cosine distance. This method has a greater accuracy value, requires less time, and is less difficult than other detecting methods. Overall, the TFIDF and cosine similarity methods have the potential to improve customer service quality by discovering similarities across client complaints [4].

In this paper, the authors have addressed that copying the entire work and representing that as your own work is global plagiarism. In this advanced generation of internet technology and easy availability of various kinds of data, from various sources has become much easier to access the data and share and copy various kinds of data which lead to plagiarism. Many students and educators frequently copy and paste the information which is available on the internet which is a notable problem of plagiarism [5].

In this paper, the existing methodologies for plagiarism detection are categorized into complete or global plagiarism which is copying entire work and representing that as your own work. Paraphrasing plagiarism means rephrasing a piece of text, sentence, or paragraph in your own words. Other types are intrinsic (internal) and extrinsic (external) plagiarism [6].

In this paper, the authors have given a brief idea about Intrinsic plagiarism. Sometimes when a plagiarism source doesn’t have any digital content then the intrinsic plagiarism detection plays as a substitute for instances like plagiarized content that is directly written by another author without copy-pasting it from any other source, another instance is when a student asks someone to write his or her essay or thesis [7].

In this paper, the author gives some insights into Extrinsic plagiarism. A suspicious document is checked against the available reference documents i.e., local corpus or collection. This local corpus is used as a reference and can either be online or offline content. The goal of any system is to find plagiarized sections in the available source document. Extrinsic plagiarism means the given document should be examined for plagiarism by comparing it with the reference document. Various technologies used for extrinsic plagiarism detection are the use of string matching, the vector space model (VSM), syntax, semantics, structural analysis, etc. [8].

This paper is a survey of the available tools. The authors have discussed the tools and the approach and techniques that are involved in them. Most of the tools that are discussed in this paper output a list of document pairs and their similarities (in percentages) [9].

In this paper the author has discussed about the Plagiarism detection systems and had given a brief about working of such systems in general. Software that detects plagiarism takes documents or text as input, process them, and output a detailed analytical report, consisting of the measure of similarity with primary sources of the plagiarized documents [10].

The system proposed in this paper was comprised of four main stages:

1. Tokenization and pre-processing.
2. Creating a corpus of sentences by mixing source and suspect documents, which are represented as a tf-idf vector.
3. Creating a similarity index between the source and the suspect document.
4. Documents with similarities more than the threshold similarity are marked as plagiarized [11].

In this study, the author created an AI chatbot that uses AIML methods such as n-gram, TFIDF, and cosine similarity to determine ranking and sentence similarity. Each sentence in the given input sentence is scored (cosine similarity score), and sentences with a high cosine similarity score are discovered for the query [12].

Digital photographs are becoming increasingly common. Every day, countless pictures are created by various organizations such as students, engineers, and physicians to meet their various demands. They may access photos based on their rudimentary properties or the text connected with them. Text in such photos can give useful information. The author in this paper intends to automatically retrieve and summarize visual information from photos. For this aim, an optical character recognition system with several algorithms is necessary. Tesseract, which was developed by HP Labs and is now owned by Google, is presently the most accurate optical character recognition engine [13].

**CHAPTER 3**

**REQUIREMENTS**

**3.1 FUNCTIONAL REQUIREMENTS**

**Input:** The system must accept input in the form of a text or as different formats such as text, PDF, Word documents and other image formats like (.png, .jpeg, .jpg and .bmp).

**Preprocess:** The system must have functionality for preprocessing the data input by the user.

**Search Engine:** The system must be equipped with a search engine that fetches related links present on the Internet.

**Scrapping:** To scrape the content from acquired links, the system must contain a scrapper.

**Evaluating Similarity:** The system must determine how closely the suspect input data resembles the content present in the matching sources.

**Report Generation:** The system must produce thorough reports on all instances of plagiarism discovered, including the proportion of similarity and the content's source.

**Accuracy:** Minimum errors must be produced by the system to deliver reliable findings.

**3.2 NON-FUNCTIONAL REQUIREMENTS**

**Performance:** The Application is built in such a way by taking all the necessary precautions and this has gone through rigorous testing to ensure the user a swift feel.

**Availability:** It will be available only when there is an internet connection because it is an online application.

**Usability:** The user interface is made so responsive which would make the user to easily work with the software.

**Robustness:** The System can withstand or overcome adverse conditions or rigorous testing.

**Maintainability:** The Software is built so simple such that any further updates if required can be implemented easily.

**3.3 HARDWARE AND SOFTWARE REQUIREMENTS**

**Hardware Requirements:**

* Monitor/Display: 14” LCD monitor, resolution of 1600 x 900 or better.
* Motherboard: Intel G41 Chipset, Intel H81 Chipset, Socket AM4 Chipset or better chipset.
* Processor (CPU): i3 and above, ryzen3 and above.
* Input Devices: Mouse, keyboard.
* Ports: Serial, parallel, USB ports.
* Ram: 2GB and above.
* Hard Disk: 150GB and above.

**Software Requirements:**

* Operating System: Windows 7 and above.
* Software: Visual Studio Code, PyCharm.
* Programming Language: Python.
* Front End: Python Web Framework FastAPI, HTML, CSS.

**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 FLOW CHART**

Extracting Text from the input document

Fetching links from the internet that matches the extracted text from the suspicious document.

Extracting Content from the links obtained (by parsing the HTML documents)

**Feature Extraction**

**Text Pre-processing**

Lower casing

Removing stop-words

Removing punctuations

lemmatization

Trigram generation

**Tokenizing**

Trigrams

**Cleaning**

Tf – Idf vectorizer

Features (Frequency Vectors)

**Generating Similarity Index**

Not Plagiarised

Plagiarised

Input Document (formats: .jpg, .jpeg, .png, .docx, .pdf,.bmp).

> 15%

< 15%



Fig. 4.1 Flow Chart.

Fig. 4.1 proposes the methodology used for the detection of plagiarism.

The proposed system detects plagiarism in different ways.

i. Detection of plagiarism in local documents.

ii. Detection of plagiarism through the web portal.

**Step 1: Input Document.**

The system lets the user input documents (suspicious documents) in different formats.

**Step 2: Text extraction.**

The system detects the type of document that is being uploaded and applies suitable conversion techniques to extract text from the documents.

**Step 3: Fetching Results online.**

The text obtained in the previous step is now used as query text to fetch results online (i.e., links to web pages).

**Step 4: Scraping Content.**

In this step, the system scrapes the content from the Internet (by parsing the html web pages).

**Step 5: Text Pre-Processing.**

**Cleaning:**

In this, the text extracted in the previous step undergoes various cleaning processes such as case conversion (to lowercase), removing stop-words, removing punctuation, and lemmatization.

**Tokenizing:**

The cleaned text is converted into trigrams. These trigrams are used for feature extraction.

**Step 6: Feature extraction.**

The trigrams are vectorized using the tfidfvectorizer which are known as features.

Tf (t) = (t € d) / (no. of words in d) (1)

idf (t, D) = log(N / count(d € D:t € d)) (2)

Where,

Tf – Term frequency.

t – term

d – document

D – Set of Documents

Idf – Inverse Document Frequency

N – Total no of Documents

**Evaluating tf-idf:**

tf-idf = tf (t, d) \* idf (t, D) (3)

Equation (1) represents the formula to evaluate tf (term frequency). Term frequency is a measure of how often a word “t” appears inside a document “d”.

Equation (2) represents the formula to evaluate idf (inverse document frequency). Inverse document frequency is a weight that indicates how frequently a word t is used across a set of document D (corpus). The lower its score, the more frequently it is used across documents. The word's importance decreases with the increase in score.

Equation (3) is the combination of tf and idf to evaluate tf-idf for a particular word.

**Example:**

A = “The car is driven on the road”; B = “The truck is driven on the highway”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Word** | **TF** | | **IDF** | **TF-IDF** | |
| ***A*** | ***B*** | ***A*** | ***B*** |
| The | 1/7 | 1/7 | log (2/2) = 0 | 0 | 0 |
| Car | 1/7 | 0 | log (2/1) = 0.3 | 0.043 | 0 |
| Truck | 0 | 1/7 | log (2/1) = 0.3 | 0 | 0.043 |
| Is | 1/7 | 1/7 | log (2/2) = 0 | 0 | 0 |
| Driven | 1/7 | 1/7 | log (2/2) = 0 | 0 | 0 |
| On | 1/7 | 1/7 | log (2/2) = 0 | 0 | 0 |
| The | 1/7 | 1/7 | log (2/2) = 0 | 0 | 0 |
| Road | 1/7 | 0 | log (2/1) = 0.3 | 0.043 | 0 |
| Highway | 0 | 1/7 | log (2/1) = 0.3 | 0 | 0.043 |

Table. 4.1 Internal evaluation of TF-IDF matrix in python.

**Step 7: Generating Similarity Index**

Using Equation (4) cosine similarity, the system evaluates the similarity index between the documents and thus concludes that the input document is plagiarized or not.

Similarity = cos(θ) = = (4)

**4.2 SEQUENCE DIAGRAM**

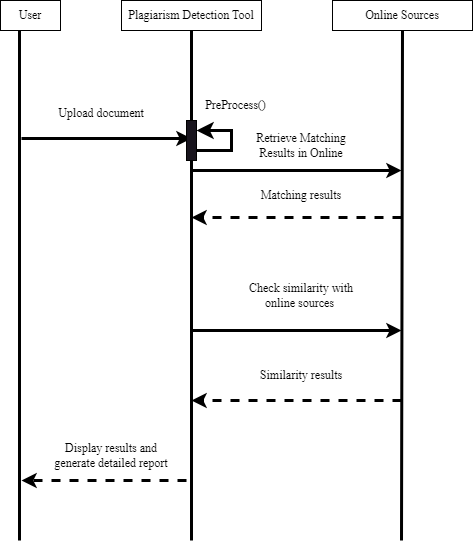


Fig. 4.2 Sequence Diagram.

In figure 4.2 the user uploads the suspicious document to by choosing the type of input to the plagiarism tool. The tool internally pre-processes the text in the input document and fetches the matching results online. The tool checks the similarity with the results obtained from the online to generate the similarity index with respect to each result. The results obtained can be viewed by the user and the tool also generates a detailed plagiarism report which the user can download.

**4.3 DATA FLOW DIAGRAM**

**Diagram

Description automatically generated**Fig. 4.3 Data Flow Diagram.

**Level 0:** It shows the complete system as a single process and does not clue as to its internal organization on the context diagram also known as level 0 DFD the systems interaction with the outside world are modelled purely in term of data flows across the system boundary.

**Level 1:** It indicates how the system is divided into subsection, each of which deals with one or more of the data to or from an external agent, each of which together provide all of the functionality of the system as a whole, it also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.

**CHAPTER 5**

**IMPLEMENTATION**

**5.1 SOFTWARE REQUIREMENTS**

Following are the Software requirements used:

1. Python.
2. FastAPI (Python Web Framework).
3. JavaScript.

**1. Python:**

Python is a popular programming language known for its simplicity and readability. It supports multiple programming paradigms and has a vast standard library and third-party ecosystem. Python code is easy to read and write due to its clean syntax and indentation-based block structure. It is an interpreted language, meaning it does not require explicit compilation. Python is widely used in various domains such as web development, scientific computing, data analysis, and machine learning.

**2. FastAPI:**

FastAPI is a modern, high-performance web framework for building APIs with Python. It is known for its speed and efficiency, making it suitable for developing high-performance applications. FastAPI leverages Python type hints and the asynchronous programming paradigm to provide automatic request and response validation, resulting in robust and reliable APIs. It integrates seamlessly with other Python frameworks and libraries, making it easy to incorporate existing code and tools. FastAPI also offers built-in support for features like OAuth authentication, dependency injection, and automatic API documentation generation. Its simplicity, performance, and comprehensive features have made FastAPI a popular choice for building web APIs in Python. The computational logic of the system has been completely coded in Python. Hence, we opted for FastAPI a Python web framework, HTML5, CSS3, and Bootstrap5 to build a web interface.

**3. JavaScript:**

JavaScript is a versatile programming language commonly used for front-end web development. It allows developers to create interactive and dynamic web pages by manipulating the Document Object Model (DOM) of a webpage. JavaScript is a client-side scripting language, meaning it runs in the user's web browser. It is also increasingly being used on the server-side with frameworks like Node.js. JavaScript supports both procedural and object-oriented programming paradigms and has a vast ecosystem of libraries and frameworks that enable developers to build complex applications. It has become an essential component of modern web development, enabling features like form validation, interactivity, animations, and AJAX-based communication with servers.

**5.2 MODULES DESCRIPTION**

1. **main:** This acts as the entry point, which contains GET and POST methods that let the user interact with the system.
2. **Driver:** This module has all the function calls required for the further computations. (i.e., it has the function calls to different functions required in the process).
3. **ScrapContent:** This Module has two following methods.
   * + - 1. **get\_urls:** Fetches the related links using Google's *custom search engine API* with respect to the input data.
         2. **get\_content:** scrape data from obtained links using *scrapper API*.
4. **Stripper:** In this module, the method remove\_tags will remove HTML tags like style, script, and head from scraped data using *bs4* and return the string.
5. **Convert:** This Module has four following methods
   * + 1. **word\_to\_txt:** If the input file format is a .docx (a Word document), then this method will be executed to extract text from a Word document using *docx2txt*.
       2. **pdf\_to\_txt:** If the input file format is a .pdf (pdf document), then this method will be executed to extract text from a pdf document using *pdfminer.high\_level*.
       3. **img\_to\_txt:** If the input file format is an image format, then this method will be executed to extract text from an image file using *Pytesseract* and *PIL*.
       4. **download\_pdf:** if copied content found in .pdf file available online then this method extracts text from .pdf file.
6. **textPreProcessing:** This Module has two following methods.
7. **Preprocess:** In this method, the text extracted in the previous steps undergoes various cleaning processes such as case conversion (to lowercase), removing stop-words, removing punctuation, and lemmatization using *spacy*.
8. **string\_matching:** This method is used to compare strings (the suspicious and the source string) by three words using the trigram approach.
9. **plagcheck:** This Module has three following methods.
   * + 1. **vectorize:** This method converts the text data into tf-idf vectors using *TfidfVectorizer.*
       2. **similarity:** This method Generates the similarity Index using *cosine\_similarity*.
       3. **check\_plagiarize:** This method detects plagiarism with respect to the input using the above two methods.

**CHAPTER 6**

**SYSTEM TESTING**

System testing is a critical aspect of Software Quality Assurance and represents the ultimate review of specification, design and coding. Testing is a process of executing a program with the intent of finding an error. A good test is one that has a probability of finding a yet undiscovered error. The purpose of testing is to identify and correct bugs in the developed system. Nothing is complete without testing. Testing is vital to the success of the system.

In the code testing the logic of the developed system is tested. For this every module of the program is executed to find an error. To perform specification test, the examination of the specifications stating what the program should do and how it should perform under various conditions.

Unit testing focuses first on the modules in the proposed system to locate errors. This enables to detect error in the coding and logic that are contained within the module alone. Those resulting from the interaction between the modules are initially avoided. In unit testing step each has to be tested separately.

Testing and validation are the most important steps after the implementation of the developed system. The system testing is performed to ensure that there are no errors in the implementation system. The software must be executed several times in order to find out the errors in the different modules of the system.

Validation refers to the process of using the new software for the developed system in a live environment i.e., new software inside the organization, in order to find out the errors. The validation phase reveals the failures and the bugs in the developed system. It will come to know about the practical difficulties the system faces when operated in the true environment.

Testing may be done at different levels:

* Unit level
* Module level
* Integration and system level

**6.1 UNIT TESTING**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST ID** | **USER** | **TESTCASE NAME** | **INPUT** | **EXPECTED OUTPUT** |
| T1 | End-User | File type Selection | Select the file type from the file selector.  Choices: Files( PNG, JPG, JPEG, BMP, PDF, DOCX) | Successful input type allowance based on the type of file selection. |
| File input other than the above formats. | File type not allowed |
| T2 | End-User | Minimum words in the input document | Document with word length > 3 (without stop words). | Valid input document. |
| Document with word length < 3. | Invalid input document |

**Table 6.1 Unit Testing**

**6.2 MODULE LEVEL TESTING**

Module testing is done at each module using test cases as prepared above. Module level testing examines the output of every module involved. Modules are designed during the time of design.

**6.3 INTEGRATION AND SYSTEM TESTING**

Integration testing is used to verify the combining of the software modules. Integration testing addresses the issues associated with the dual problems of verification and program construction. System testing is used to verify whether the developed system meets the requirements.

**CHAPTER 7**

**RESULTS AND DISCUSSIONS**

A screenshot of a computer

Description automatically generated

Fig. 7.1 Home Page.

A screenshot of a computer

Description automatically generated

Fig. 7.2 UI for Text Input.

A screenshot of a computer

Description automatically generated with low confidence

Fig. 7.3 Results obtained for text input.

A screenshot of a computer

Description automatically generated with medium confidence

Fig. 7.4 UI for pdf file input.

A screenshot of a computer

Description automatically generated with low confidence

Fig. 7.5 Results obtained for pdf file input.

A screenshot of a computer

Description automatically generated with medium confidence

Fig. 7.6 UI for Image file input.

Application

Description automatically generated with low confidence

Fig. 7.7 Results obtained for image file input.

**CHAPTER 8**

**CONCLUSION AND FUTURE SCOPE**

**8.1 CONCLUSION**

A novel approach is proposed, and a tool is developed that detects the contents of plagiarism in the input document. The proposed tool takes the input, which is either in the form of a pdf, a document (.docx) format, or any kind of image format such as .jpeg, .png, or .bmp, and then compares it with the contents available online. It generates the report separately by specifying the matching source present on the website. So that the tool is tested for its correctness and completeness by comparing it with various documents available online.

**8.2 FUTURE SCOPE**

* 1. As the model does not have access to proprietary or licensed content, the future scope could include collaborating with content providers or databases to obtain necessary permissions and access to their proprietary or licensed content. This could enable the model to detect plagiarism more accurately by comparing it against a wider range of sources, including proprietary and licensed content.
  2. To address the limitation of not detecting text within images embedded inside the document, the future scope could be an optimized system that checks for images inside documents further text from those images will be extracted from OCR and then checks for plagiarism.
  3. To overcome the limitation of processing documents limited to 5 - 6 pages, the future scope could involve enhancing the processing capacity of the tool to handle longer documents. This could be achieved through the optimization of algorithms, improving computational resources, and utilizing distributed processing techniques. Increasing the processing capacity would enable the tool to handle larger documents and provide more comprehensive plagiarism detection for longer content pieces.

**PUBLICATION**

This paper is published in 2nd IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics organized by the Ballari Institute of Technology and Management, Ballari, India in association with IEEE Bangalore Section on 29th – 30th April 2023.

**REFERENCES**

[1] M. A. C. Jiffriya, M. A. C. A. Jahan and R. G. Ragel, "Plagiarism detection on electronic text-based assignments using vector space model," 7th International Conference on Information and Automation for Sustainability, 2014, pp. 1-5, Doi: 10.1109/ICIAFS.2014.7069593.

[2] N. F. A. N. Rahiman, M. G. M. Johar and R. A. A. Helmi, "CopyPoppy – A Source Code Plagiarism Detector," 2022 IEEE 10th Conference on Systems, Process & Control (ICSPC), Malacca, Malaysia, 2022, pp. 123-128, doi: 10.1109/ICSPC55597.2022.10001740.

[3] N. V. Ravindhar, G. Nagappan, G. Lokesh, P. Punith and P. Prabhu, "To Prevent Copy Right Infringement Piracy Plagiarism of NCERT Textbooks," 2023 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE), Bengaluru, India, 2023, pp. 606-611, doi: 10.1109/IITCEE57236.2023.10090901.

[4] A. D. Hartanto, Y. Pristyanto and A. Saputra, "Document Similarity Detection using Rabin-Karp and Cosine Similarity Algorithms," 2021 International Conference on Computer Science and Engineering (IC2SE), Padang, Indonesia, 2021, pp. 1-6, doi: 10.1109/IC2SE52832.2021.9791999.

[5] Vandana, "A Comparative Study of Plagiarism Detection Software," 2018 5th International Symposium on Emerging Trends and Technologies in Libraries and Information Services (ETTLIS), 2018, pp. 344-347, Doi: 10.1109/ETTLIS.2018.8485271.

[6] Prof.Sneha Autade, Anuja Suryawanshi, A Systematic Literature survey on Plagiarism detection Tools, International Journal of Application or Innovation in Engineering & Management (IJAIEM), Volume 7, Issue 2, February 2018, pp. 030-034, ISSN 2319 - 4847.

[7] Imene Bensalem, Paolo Rosso, Salim Chikhi, Intrinsic Plagiarism Detection using N-gram Classes, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1459–1464, October 25-29, 2014, Doha, Qatar. c 2014 Association for Computational Linguistics.

[8] Kanjirangat, Vani & Gupta, Deepa. (2016). A Study on Extrinsic Text Plagiarism Detection Techniques and Tools. JOURNAL OF ENGINEERING SCIENCE AND TECHNOLOGY REVIEW. 9. 150-164. 10.25103/jestr.095.02.

[9] V. Ljubovic and E. Pajic, "Plagiarism Detection in Computer Programming Using Feature Extraction from UltraFine-Grained Repositories," in IEEE Access, vol. 8, pp. 96505-96514, 2020, Doi: 10.1109/ACCESS.2020.2996146.

[10] Abdelhamid, Mehdi & Azouaou, Faical & Batata, Sofiane. (2022). A Survey of Plagiarism Detection Systems: Case of Use with English, French and Arabic Languages.

[11] Muharram Mansoorizadeh & Taher Rahgooy. Persian Plagiarism Detection Using Sentence Correlations.

[12] L. Athota, V. K. Shukla, N. Pandey and A. Rana, "Chatbot for Healthcare System Using Artificial Intelligence," 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2020, pp. 619-622, doi: 10.1109/ICRITO48877.2020.9197833.

[13] C. Kaundilya, D. Chawla and Y. Chopra, "Automated Text Extraction from Images using OCR System," 2019 6th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2019, pp. 145-150.

[14] S. Prasanth & R. Rajshree & B. Saravana Balaji. (2013). A Survey on Plagiarism Detection. International Journal of Computer Applications. 86. Doi: 10.5120/15104-3428.