

**PLAGIARISM CHECKER**

#### *Submitted by*

#### **MEENAKSHI R – 2116211701031**

**SENTHAMIZH VANI - 2116211201049**

#### CS19643 Foundations of Machine Learning

Department of Computer Science and Design

**Rajalakshmi Engineering College, Thandalam.**

BONAFIDE CERTIFICATE

#### This is to certify that the Mini project work titled “**PLAGIARISM CHECKER”** done by **Meenakshi R 2116211701031, Senthamizhvani 2116211701049** is a record of bonafide work carried out by them under my supervision as a part of MINI PROJECT for the subject titled CS19643 Foundations of Machine Learning by Department of Computer Science and Design.

|  |  |
| --- | --- |
| HEAD OF THE DEPARTMENTMr. S. Uma Maheshwara Rao,Professor and Head,Computer Science and Design,Rajalakshmi Engineering College, Thandalam,Chennai – 602 105. | FACULTY IN CHARGEMs. E. Preethi,Assistant Professor (SG),Computer Science and Design,Rajalakshmi Engineering College, Thandalam,Chennai – 602 105. |

This project report is submitted for practical examination for CS19643 Foundations of Machine Learning to be held on………………. at Rajalakshmi Engineering College, Thandalam.

|  |  |
| --- | --- |
| **EXTERNAL EXAMINER** | **INTERNAL EXAMINER** |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.No** | **CHAPTER** | **PAGE NUMBER** |
| 1. | ABSTRACT | 4 |
| 2. | INTRODUCTION | 5 |
| 3. | LITERATURE SURVEY | 9 |
| 4. | MODEL ARCHITECTURE | 14 |
| 5. | IMPLEMENTATION | 17 |
| 6. | RESULTS AND DISCUSSIONS | 33 |
| 7. | CONCLUSION | 36 |
| 8. | REFERENCES | 37 |
| 9. | APPENDIX I-CODING | 38 |
| 10. | APPENDIX II-OUTPUT SCREENSHOTS | 41 |

**ABSTRACT**

Being a growing problem, plagiarism is generally defined as “literary theft” and “academic dishonesty” in the literature, and it is really had to be well-informed on this topic prevent the problem and stick to the ethical principles. This paper presents a survey on plagiarism detection systems, a summary of several plagiarism types, techniques, and algorithms is provided. Common feature of deferent detection systems is described. end of this paper authors proposed a web enabled system to detect plagiarism in documents, code and images, also this system could be used in E-Learning, E-Journal, and E-Business. This tool is being used to detect out the similarities in the contents in many fields, such as, in students’ assignments, teachers’ study materials, bloggers’ content, echo avoid plagiarism. The availability of various documents on internet makes students to copy paste very easily, which leads to take grades without knowledge background. Plagiarism can be described as “taking someone else Idea or work as your own” which is obviously a bad practice. In this project we have also described about another concept which is paraphrasing. Paraphrasing is basically, reading the available facts or content of others and writing it in your own understanding

**CHAPTER 1**

**INTRODUCTION**

The term “plagiarize” is defined as to take (ideas, documents, code, image, etc.,) from another and pass them off as one's own without citation. So plagiarism is a global problem, which occurs in many different areas of our life. There are many different forms of plagiarism, Plagiarism at schools can be a highly de-motivating factor for teachers and also for students. If plagiarism is not addressed sufficiently, plagiarists could gain undeserved advantage, e.g. more marks for their assignments with less effort. There are various types of plagiarism involved: using sources without properly citing them, paraphrasing text, reusing ideas with/without citing references, and others. A plagiarized document detection plays important roles in many applications, such as file management, copyright protection, and plagiarism prevention. Existing protocols assume that the contents of files stored on a server are directly accessible. This assumption limits more practical applications, e.g., detecting plagiarized documents between two conferences, where submissions are confidential. Plagiarism can take one of the popular types such as copying of the whole or some parts of the document, rewording same content in different words, using others’ ideas or referencing the work to incorrect or non-existing sources. Other ways of plagiarism include translated plagiarism wherein the content is translated and used without referencing the original work, artistic plagiarism in which different media such as images and videos are used to present other’s work without proper citation, A plagiarized code which can be defined as the reuse of the source code without permission or citation. So the plagiarized program can be defined as a program which has been produced from another program with a small number of routine transformations, routine transformations, typically text substitutions, do not require a detailed understanding of the program. Unfortunately, plagiarism of programming assignments has been made easier by large class sizes. Plagiarism of computer programs can become quite common in large undergraduate classes. With a few simple editor operations, it is possible to produce a plagiarized program with a different visual appearance. This makes the manual detection of plagiarized program difficult in large classes. All these practices of plagiarism have negative impact on the learning process. Thus, how can we ensure dealing with plagiarism systems and how is plagiarism going to be detected and dealt with. It is a critical issue that needs solutions by computer scientists.

**Types of Plagiarism:**

Anyone who has written or graded a paper knows that plagiarism is not always a black-and white issue. The boundary between plagiarism and research is often unclear. Learning to recognize the various forms of plagiarism, especially the more ambiguous ones, is an important step in the fight to prevent it

1. SOURCES NOT CITED
2. “The Ghost Writer” The writer turns in another’s work, word-for-word, as his or her own.
3. “The Photocopy” The writer copies significant portions of text straight from a single source, without alteration.
4. “The Potluck Paper” The writer tries to disguise plagiarism by copying from several different sources, tweaking the sentences to make them fit together while retaining most of the original phrasing.
5. “The Poor Disguise” Although the writer has retained the essential content of the source, he or she has altered the paper’s appearance slightly by changing key words and phrases.
6. “The Labor of Laziness” The writer takes the time to paraphrase most of the paper from other sources and make it all fit together, instead of spending the same effort on original work.
7. “The Self-Stealer” The writer “borrows” generously from his or her previous work, violating policies concerning the expectation of originality adopted by most academic institutions.
8. SOURCES CITED (but still plagiarized!)
9. “The Forgotten Footnote” The writer mentions an author’s name for a source, but neglects to include specific information on the location of the material referenced. This often masks other forms of plagiarism by obscuring source locations.
10. “The Misinformer” The writer provides inaccurate information regarding the sources, making it impossible to find them.
11. “The Too-Perfect Paraphrase” The writer properly cites a source, but neglects to put in quotation marks text that has been copied word-for-word, or close to it. Although attributing the basic ideas to the source, the writer is falsely claiming original presentation and interpretation of the information.
12. “The Resourceful Citer” The writer properly cites all sources, paraphrasing and using quotations appropriately. The catch? The paper contains almost no original work! It is sometimes difficult to spot this form of plagiarism because it looks like any other well researched document.
13. “The Perfect Crime” Well, we all know it doesn’t exist. In this case, the writer properly quotes and cites sources in some places, but goes on to paraphrase other arguments from those sources without citation. This way, the writer tries to pass off the paraphrased material as his or her own analysis of the cited material.

Why Plagiarism Detection is Important?

In some of the academic enterprises like universities, schools and institutions, plagiarism detection and prevention became one of the educational challenges, because most of the students or researchers are cheating when they do the as-signed tasks and projects. This is because a lot of resources can be found on the internet. It is so easy to them to use one of the search engines to search for any topic and to cheat from it without citing the owner of the document. So it is better and must all academic fields they should have to use plagiarism detection soft-wares to stop or to eliminate students cheating, copying and modifying documents when they know that they will be found

#### What are copyright laws?

Copyright laws exist to protect our intellectual property. They make it illegal to reproduce someone else’s expression of ideas or information without permission. This can include music, images, written words, video, and a variety of other media. At one time, a work was only protected by copyright if it included a copyright trademark (the symbol). According to laws established in 1989, however, works are now copyright protected with or without the inclusion of this symbol. Anyone who reproduces copyrighted material improperly can be prosecuted in a court of law. It does not matter if the form or content of the original has been altered – as long as any material can be shown to be substantially similar to the original, it may be considered a violation of the Copyright Act Scope This document will address two key themes in terms of the Plagiarism Detection System - A submission system of authenticated users and plagiarism detection of submitted documents. This system is only for academic purpose. Any academic activity which is not related to document submission and viewing documents is considered as out of scope, and will not be covered in this document.

**CHAPTER 2**

## LITERATURE SURVEY

We classified the survey into four categories:

* + 1. Plagiarism in documents.
    2. Plagiarism in code.
    3. Plagiarism techniques.
    4. Plagiarism algorithms.

These categories explained as follows:

1. Plagiarism in Documents

Most of the work in document plagiarism has been done for academic purpose. Detecting plagiarism is important to judge and mark students’ work especially for postgraduates who are strictly prohibited from cheating, rewording, rephrasing, or restating without referencing. In this regard, numerous plagiarism detection systems have been developed. These systems can be classified into two main categories, web enabled systems and stand-alone systems.

1.Web-enabled systems: Developing web systems for plagiarism detection overcomes machine capability problems, facilitate the availability of the system to many users and extend the search of plagiarized resources to the World Wide Web easily. Here is discussion of two: First Turnitin [5, 6] is the most well-known commercial plagiarism detection system to which many universities from UK and USA subscribe. It uses an enormous database from the Internet and previous student works to be compared with the query document.

2.SafeAssign checks all submitted papers against the following databases:

1. the Internet.
2. ProQuest database.
3. Institutional document archives containing all documents submitted toSafeAssign.
4. Global Reference Database containing documents that were volunteeredbstudents to help prevent cross-institutional plagiarism.

A)Stand-alone systems:

Stand-alone software is developed to be installed on computers. Two systems will be explored here, EVE and WCopyFind. First EVE (The Essay Verification Engine) is a desktop application but it has the capability to make large number of searches on the Internet to locate matches between sentences in the query document and suspected websites. Thus, in order for EVE to work, the machine should be connected to the Internet. Second WCopyFind developed by University of Virginia, finds plagiarism between two or more assignments. The user can set or change some of the parameters that may influence the detection process such as the number of words used for detecting similarity among statements. Several other tools have been developed for plagiarism detection. They use variety of document characteristics that need different plagiarism detection approaches such as fingerprinting and fuzzy information retrieval.

1. Plagiarism in Code:

Various plagiarism approaches have been proposed for detecting source code written with PYTHON Each of these approaches focuses on certain characteristics of code plagiarism. For example, there are approaches which are designed mainly to compare source codes written in different programming languages. There are also approaches which are designed to handle complicated code modification but require longer detection time compared to common approaches. One of the approaches that we considered suitable for detecting plagiarism in programming course is the structure based method, which mostly use tokenization and string matching algorithm to measure similarity. Some of existing plagiarism detectors that employ such structure

based methods are Plague, YAP and JPlag.

1. Plague is one of the earliest structure-based detectors. Plague works in several steps. First, structure profiles of each source code are created. Then, those structure profiles are compared using K Means algorithm. Suggested by James MacQueen in 1967, the algorithm is designed to handle text files. Plague’s detection results are returned in the form of lists. By using a corresponding interpreter, the results can be processed further to make it easier to comprehend for common users. Plague is able

to detect plagiarism for source code written in PYTHON.

1. YAP was developed based on Plague with some enhancements. The first version was created by Michael Wise. Then it was optimized into YAP2. The final version YAP3, which can also be used to detect text plagiarism . All three versions of YAP have two phases in their processes. The first phase is the generation phase, where a token file is created for each source code. The second phase is comparison of every token files. The result of each comparison is a value called percent match, a value between 0 as minimum and 100. If the percent match of a pair of token files is larger

than this minimum value, then the corresponding pair will be judged as a case of suspected plagiarism. YAP’s detection result is presented in the form of a text file. JPlag is a system that can be used to detect plagiarism for source code written in PYTHON. It is available as a free web service. Its input is a directory containing programs that will be detected. Every source code in the directory are parsed and

transformed to token strings.

1. These token strings will be compared to each other using Running Karp-Rabin Greedy String Tiling algorithm.JPlag’s detection result is displayed as a group of HTML files that can be opened using a standard browser. Detection statistics, similarity distribution, and pairs of programs suspected as plagiarism instances are shown on the main page. The user can also choose a certain pair of program to be shown side-
2. Plagiarism Techniques

Plagiarism techniques known as similarity detection techniques. A good example is found in the formerly popular attribute counting techniques. Attribute counting techniques create special “fingerprints” for collection files, including metrics, such as average line length, file size, average number of commas per line. The files with close fingerprints are treated as similar. Clearly, small fingerprint records can be compared rapidly,but this technique is now considered unreliable, and rarely used nowadays . Modern plagiarism detection systems usually implemented using certain content comparison techniques. The most popular techniques include string tiling, finding the joint coverage for a pair of file and parse trees comparison. Usually these techniques work for file pairs, so the comparison routine should be called for each possible file pair found in the input collection. Also Fast Plagiarism Detection technique (FPDS) tries to improve the algorithmic performance of plagiarism detection by utilizing a special indexed data structure to store input collection files. And Tokenization is a commonly used technique that fights against renaming variables and changing loop types in computer programs. Simple tokenization algorithms substitute the elements of program code with single tokens.

1. Plagiarism Algorithms

A number of algorithms to detect plagiarism are discussed. The simple algorithm based on string comparisons will explain as shown below:

1. Remove all comments.
2. Ignore all blanks and extra lines, except when needed asdelimiters.
3. Perform a character string
4. Maintain a count of percentages of character correlation.

This algorithm is run for all possible program pairs. This simple algorithm will detect many cases of plagiarism. For code plagiarism detection, Faidhi and Robinson characterize six levels of program modification in a plagiarism spectrum. Level 0 is the original program without modifications. In level 1, only comments are changed. Level 2 changes the identifier names. Level 3 changes position of variables. Level 4 changes constants and procedures. In level 5 program loops are changed. In level 6 control structures are changed to an equivalent form using a different control structure (i.e. “for” changed to “if”). Several algorithms for plagiarism detection are based on software metrics. Theses algorithms extract several software metrics features from a program and use this set of features to compare programs for plagiarism.

**CHAPTER 3**

**MODEL ARCHITECTURE**

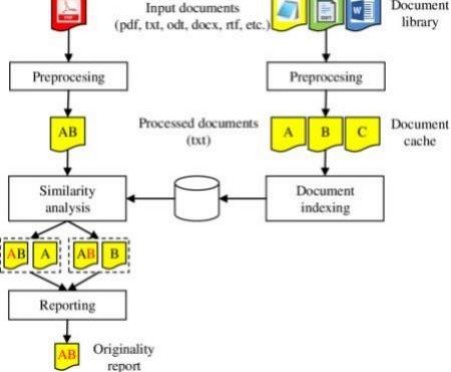


Fig 1 Model Architecture of Plagiarism Checker

After the search for plagiarized text is completed, a reporting module receives the output of the similarity analysis module (i.e. the list of non-original fragments and information about their sources) and summarizes the results, providing statistics about the level of originality, top plagiarized sources and more. The similarity analysis task is the one that requires the most time and computing power to complete. Searching directly within the original documents available in the document library is very inefficient due to the preprocessing stage required for each document, for each fragment text search. The first step towards improving the search speed is to cache the common format of the preprocessed version of each document. In this way, the preprocessing of documents happens only once per document. If a document is added to, changed or removed from the document library, then the document cache should be updated also. If the preprocessing algorithm or its parameters change, then the document cache has to be completely rebuilt. This is one of the reasons why most commercial plagiarism detection services request in their terms of service a permanent unlimited license on the uploaded.

State Transition Diagram:

State Transition Diagram represents active states for each class and the events (triggers) that cause changes between these active states. Here I have provided diagram for each of the actors.

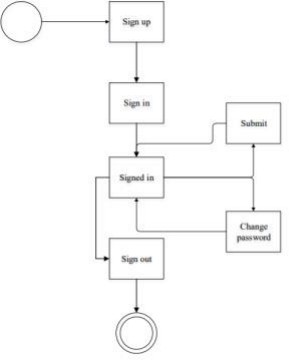


Fig 2. State Transition Diagram-Student

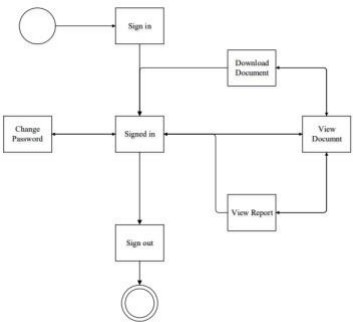


Fig3. State Transition Diagram-Teacher

**CHAPTER 4**

**IMPLEMENTATION**

A plagiarism checker uses advanced database software to scan for matches between your text and existing texts. They are used by universities to scan student assignments. There are also commercial plagiarism checkers you can use to check your own work before submitting. Behind the scenes, plagiarism checkers crawl web content and index it, scanning your text for similarities against a database of existing content on the internet. Exact matches are highlighted using keyword analysis. Some checkers can also identify non-exact matches (paraphrasing plagiarism).On the user end, the checker typically provides you with a plagiarism percentage, highlights the plagiarism, and lists the sources.

##### Semantic search

Searching text and documents can generally be done in two ways. Lexical search looks for patterns and exact word or string matches, while semantic search uses the meaning of your search query or question and puts it into context. Vector databases store and index vector embeddings from Natural Language Processing models to understand the meaning and context of strings of text, sentences, and whole documents for more accurate and relevant search results.Using natural language queries to find relevant results is a better experience and allows users to find what they need more quickly without having to know specifics about how the data is

classified.

##### Similarity search for images, audio, video, JSON, and other forms of unstructured data

Images, audio, video, and other unstructured datasets can be very challenging to classify and store in a traditional database. This often requires keywords, descriptions, and metadata to be manually applied to each object. The way one individual classifies one of the complex data objects may not be obvious to another. As a result, searching for complex data can be very hit and miss. This approach requires the searcher to understand something about how the data is structured and construct queries that match the original data model.

##### Ranking and recommendation engines

Vector databases are a great solution for powering ranking and recommendation engines. For online retailers, they can be used to suggest items similar to past purchases or a current item the customer is researching. Streaming media services can apply a user’s song ratings to create perfectly matched recommendations tailored to the individual rather than relying on collaborative filtering or popularity lists. The ability to find similar items based on nearest matches makes vector databases ideal for offering relevant suggestions, and can easily rank items based on similarity scores

##### Deduplication and record matching

Another use case for vector similarity search is record matching and deduplication. Using the similarity service to find near-duplicate records can be used in a wide range of applications. Consider an application that removes duplicate items from a catalog to make it far more usable and relevant.

##### Anomaly detection

As good as vector databases are in finding similar objects, they can also find objects that are distant or dissimilar from an expected result. These anomalies are valuable in applications used for threat assessment, fraud detection, and IT Operations. It’s possible to identify the most relevant anomalies for further analysis without overwhelming resources with a high rate of false alarms.

Some Examples of Plagiarism tools:

Plagiarism Detection Tools In the past two decade, several plagiarism detection tools have been developed. Some of these tools are discussed in brief, next. Also, we have analyzed their pros and cons, and reported in a tabular form in Table 2We reported the classification of tools

* 1. Grammarly Online Plagiarism Checker: This is by far the best online Plagiarism checker tool which you can use. Grammarly is a well-known tool among writers and also who need to quickly check if article is original or copied from other places. If you are a professor and need to check if the research paper is original or copied from multiple sources online, this tool is perfect for you. See in the above screenshot, how easy it is to use the plagiarism feature of Grammarly. This is a multi-feature tool for webmasters to check the originality of the content, check Grammar, Check the spelling of the article. I have been using it for a while, and highly recommend it to you.
  2. Unicheck:- If you are looking for a solid paid option then Unicheck could be the right tool for you. The interface is sleek and it checks the pages really fast. This is a perfect tool corporates and professors who don’t mind paying a little for higher accuracy.
  3. 4.Safe Assignment**:** This anti-plagiarism checker claims to search an index of 8 billion documents available in the Web. It uses some major scholastic databases like ProQuest™, Find Articles™ and Paper Mills during searching and detection process. Safe Assignment maintains a database where user account is essential to keep fingerprints of the submitted documents in order to avoid any legal or copy right problem. This tool uses proprietary searching and ranking algorithms for match detection of fingerprints with its resources. The results of plagiarism detection is presented to the user within couple of minutes.
  4. Docol:This Web based service uses capabilities like searching and ranking of Google API. The submitted document is uploaded to a server and evaluation is done in the server side. The software provides a simple console to set fingerprint (search fragments) size,date constraints, filtering and other report related options. The evaluation result is sent to the user through email identifying plagiarized sections and sources of plagiarism. This is totally Google API dependent and so it may be unavailable at any point of time.

**CHAPTER 5**

**ALGORITHM**

## K-Means Algorithm:

Clustering is typically used to cluster different data samples into groups or in data mining field. In our case, we want to use clustering as a vector pre-sorting tool, which then allows us to simplify the search for similar vectors. By default, using clustering methods in the data mining field, the input data set is divided into two groups – the training and control set. The training set is used to create clusters and the data from the control set verifies the accuracy of this classification. Our goal was to design a clustering method that would allow new data to be added gradually and to maintain the clusters in an optimal configuration. The basic K-Means algorithm in its specification only describes the principal ideas, but not the ways of effective implementation. If we want to use this algorithm in our method, we need to design and implement several improvements. To improve the algorithm efficiency, we need to identify which parts of it are significant for us. In most cases, the improvement of the initial phase of the algorithm – finding the initial centres – is the good starting point for such attempts. The starting distribution is important for two reasons – a better start-up distribution speeds up the algorithm and can eliminate the problem of suboptimal solutions where the algorithm drops to a local minimum. Despite the importance of this initial distribution, we decided not to focus on optimizing the initial cluster distribution in our approach. This does not make sense to our algorithm because it is done only at the beginning and the centres are being recalculated during the algorithm run. This calculation is based on the previous state and there is no need to reinitialize the clusters again. The K-Means algorithm is relatively simple, so there are only a few aspects that can be improved. Our analysis showed that the most ineffective part is the assignment of vectors to clusters because this kind of operations has the complexity of O(n\*k) where n is the number of vectors and k is the number of clusters. For each vector cluster combination, their distance must be calculated, and the calculation of the Euclidean distance for vectors of length 162 also takes some time. In this section, we will focus on optimizing the second part of the K-Means algorithm, because this part runs very often.

We suggest the improvement of the algorithm in several points:

* Adding parallelization,
* improving the logical structure of the algorithm,
* improving implementation techniques,
* pre-processing of input data.

The following chapters we will detail the mentioned points. At the end, we will evaluate the suggested techniques and compare the speed of the algorithm before and after the application of these techniques.

One of the basic techniques of algorithm acceleration is their parallelization. In the literature, we can find several methods that attempt to parallelize the K-Means algorithm. In our work, we decided to explore and implement some basic parallelization methods. In general, a parallelization of algorithms is not a trivial task . The main goal is to use most of the available computing power of the CPU. In the basic implementation of the algorithm, we observed 25% CPU usage during the run because the test was run on a processor that had 2 hyperthreaded cores.Given the large dimension of the input data and the frequency of the distance calcu-lation operations, we decided first to parallelize the Euclidean distance calculation. We divided the input vectors into several groups and calculated the distances of these parts Increasing K-Means Clustering Algorithm Effectivity 125 in parallel. Finally, we combined the partial results. This principle is generally known as MapReduce. We used the same approach by new cluster centres calculating, where we calculated the individual elements of the resulting centre in parallel. As can be seen in this method has already been able to utilize almost all available CPU computing resources. On the other hand, this approach (as will be shown in the results) brings considerable overhead costs. Managing threads (through which parallelization has been implemented), synchronization, and other aspects of this approach do not allow full utilization of the parallelization capabilities and the amount of computing power is consumed by the operating system. Some approaches can partially reduce this overhead by using the thread workers pool that eliminates some resource management overhead. Although this approach utilized the available computing resources to the maximum, the benefit we gained was less than the added overhead of the operating system. Based on these results, we built our final parallelization proposal at a higher level ofthe algorithm. We will parallelize the operation of assigning vectors to clusters as the first step. The single assignments are independent of each other, so we can perform this operation in parallel. At the beginning, we divide a set of vectors that we can assign to n groups (n equals to the number of available CPU cores) and then we perform the vector assignment in parallel. When implementing, we need to be aware of how the assignment operation is implemented, as it may require some degree of synchronization. In the second step, we divide the clusters into n groups, and then we calculate the centres for the individual clusters in parallel as well. As in the previous case, the clusters are independent of each other, so such an approach should not cause any synchronization problem. this approach does not reach 100% utilization of available computing resources. This problem is caused by uneven cluster size, so some cluster groups are calculated faster than others. The solution could be to smarter inclusion of clusters into groups or to use an approach where the calculation of each cluster would represent a separate task, and these tasks would be planned based on the current availability of computing resources. On the other hand, our results show that performance improvements using this approach are minimal, but implementation complexity increases significantly.

## String Matching Algorithm:

#### Text Mining

Text mining refers to the process of analyzing and exploring large sets of unstructured data to identify and isolate topics, keywords, concepts, patterns and other elements evident in data sets. The process is becoming more practical due to the emergence of large data platforms and various deep learning algorithms with the ability to analyze big sets of unstructured data . Through text mining, it is possible to focus on text data instead of other more structured forms of data examined in data mining. However, both the processes involve organizing and structuring the data in a particular fashion before subjecting it to qualitative and quantitative analysis. Text mining comprises of a wide set of algorithms and topics for analyzing texts and the ability to analyze big sets of unstructured data . Through text mining, it is possible to focus on text data instead of other more structured forms of data examined in data mining. However, both the processes involve organizing and structuring the data in a particular fashion before subjecting it to qualitative and quantitative analysis. Text mining comprises of a wide set of algorithms and topics for analyzing texts and by using deep learning vocabulary network portray improved clustering performance. Furthermore, in addressing the diversity and complexity of natural language, the semantically responsive text clustering approach has been proposed in. The authors suggest an extension-oriented modeling approach along with a similarity aggregation method and a space construction method. This approach has the ability to improve semantic sensitivity during the text mining process by organizing the generated clusters into different granularities. The effectiveness of this approach is confirmed through spanning different communities. Some of the common text mining tools include text clustering, text visualization, and association rule extraction. Text clustering rests on the cluster hypothesis that argues that clustering is possible when the relevant documents possess more similarities with each other than with non-relevant documents. It is a trustworthy technique that is largely applied in analyzing and examining big sets of data akin to data mining . Moreover, it is proven that text clustering remains an effective tool for procedure of topic ranking in a dynamic set of data is gaining the attention of researchers using text clustering in the digital field . Even though text clustering remains vital in browsing and navigation processes, various text clustering strategies fail to address challenges such as space complexity and high time, less robustness, privacy risks, inability to comprehend contextual elements of words, etc. Authors in propose an efficient text-based clustering framework. They determined the similarities between words by using the cosine similarity. The data vector was designed from the component similarities and was later used to compute the clustering particles. Following mutation to maximize clustering, the framework is examined by using Mean Square Error (MSE), Processing Time (PT), and Peak Signal to Noise Ratio (PSNR) . The findings indicate that the proposed framework produces optimal PSNR, MSE and PT compared to the Pair-wise Random Swap (PRS) and Fuzzy CC Means (FCM) approaches. Authors in reported that another way of overcoming challenges to text mining such as data dimension reduction and extraction is by using a deep learning vocabulary network. The vocabulary network’s design is based on closely related word sets that,contain the concurrence relations of terms or words. Here, the frequency in feature vectors is replaced with the significance of words in terms of page rank and vocabulary network to create exact feature vectors that depict the meanings of text clustering. When the deep-learning vocabulary network was compared to other approaches such as representative algorithms, the results demonstrate that feature vectors determined associated with each other in the database, while confidence is the portion of data-sets that contains associated text data in the database. Here, the researcher employs a prior algorithm to find frequently occurring items over transaction data sets, and the technique is effective in matching similar and dissimilar text data to offer insight for decision-making. Some researchers argue that association rule extraction remains an effective approach of finding accurate and vital knowledge from an assortment of documents. Today, information technology and the internet are platforms that provide huge recognized examinations of clustering data-sets and algorithms. Authors in collected and textually theme analysis of texts, while it also facilitates topic analysis technique where named parties with concurrent occurrences are categorized together and the frequent entity is placed in various sets by using the graph-based approach. Currently, the analyzed various documents from six scientific databases. Text clustering was shown to be effective in matching texts from natural language documents. In cases where similarity was not detected, they attested it to ambiguity or interrelations between analyzed text documents. Cosine similarity is still the most common metric of similarity. The collection of sentences is carried out by selecting sentences from each cluster on of basis of term frequency and inverse term frequency rankings in this same cluster .

1. Association Rule Extraction:

Approaches associated with association rule extraction focus on identifying the relationships in a big set of variables in data-sets. Association-Rule Mining (ARM) strategies identify the variable-value arrangements that occur repeatedly. It is also called knowledge discovery because it is likened to correlations analysis that focuses on the associations between two variables. The association procedures for text mining focus on exploring the relationships among various factual notions or topics used to describe a body of data. The primary objective is to discover the vital association rules comparative to a body of data in such a manner that the existence of particular topics in articles may resemble the existence of other topics. Some researchers agree that association rule mining is a prominent research field that used to unearth frequent patterns and arrangements in repositories of synthetic or real-world data-sets. Association rule mining underscores that various associations are related or occur among a group of datasets in the database. In many ARM approaches, the rules are evaluated from two main measures, namely confidence and support. Support is the aggregate number of text data that are amounts of data and information to users. However, a major issue facing text matching is that searching and finding precise and crucial information may be time-consuming or lead to misunderstandings. Creating a method to generate knowledge discovery through the association rule extraction proves to be effective in both saving time and avoiding misunderstandings. Authors in support the use of a temporal approach in extracting negative and positive association rules from texts. Data extraction by using the temporal approach is not popular amongst the ARM researchers, especially for negative associations. However, the researchers insist that the rule allows the researchers to answer essential questions when applying association rules, which boosts the trustworthiness of the resulting association rules for use in matching texts. Traditionally, association analysis is an unsupervised strategy mostly evident in knowledge discovery assignments. Therefore, the majority of the relevant studies focus on simplifying association rules and improving the performance of algorithms. However, other problems arise when association rules are created drawback is that word cloud fails to contemplate the linguistic knowhow about text words and corresponding associations with a specific subject when offering a numerical summary to the isolated words . As a result, in the majority of the systems, the word clouds are usually used in a numerical manner to summarize texts and offer little or no chance to correlate the data. Authors in provide the findings of a set of controlled research experiments that demonstrate that layouts where text words are structured into visually and semantically distinctive categories are more efficient and effective for acknowledging underlying topics compared to ordinary world cloud designs. The white space separators and spatially grouped color coding lead to considerably powerful understanding of the fundamental topics as compared to an ordinary word design or layout. As a result, data mining experts are developing data-sets for visually and semantically distinct category identification tasks for use in replicating results for word cloud formats and designs in the future. Authors in agree that word cloud remains an effective text matching tool in writing and reading classes, especially English classes for ESL learners. During the decade, many word cloud applications were designed to provide additional visual appeal to slide shows, posters, and websites amongst others. For writing and reading classes, word clouds are vital for reducing reading time and helping with vocabulary and writing comprehension. Moreover, word clouds are also used for addressing rhetoric elements in instructors’ assignment

#### Word Cloud

Another text mining technique in common use is word cloud. As noted, a word cloud or tag cloud is the visual representation of a word for a particular written content arrangement according to its frequency. It is amongst the most common methods for graphically presenting text data, which makes it vital for examining different kinds of texts such as written opinions or short answers to a questionnaire or survey. It is a preliminary phase during the in-depth analysis of particular text material and the needed information. Nevertheless, this technique faces various challenges. A major descriptions, syllabuses, and students’ essays. The process locates a collection of vocabulary in texts for use in creating word clouds that assist students in understanding and summarizing texts, learning collocations and spellings, finding synonyms, and avoiding repetition, and using word maps for creating . Parsing the text in a document file is a crucial task. In order to compare every word or sentence, we should be able to find it in the source file easily. The result for the search can be either “found” or “not found”. But accessing the source file word by word and also frequently can take the complexity of such a parsing to O(nn ) which is an exponential growth. Such a parsing can take huge amount of time to give an output. A theoretical algorithm for searching an element which takes O(1) complexity, can get output in a single instruction, but it is not possible to implement in any of the language because it is an NP-HARD problem in Computer Science. Still research is carried in this field and the problem is stuck on whether P = NP or not. The minimum worst-case complexity in searching an element from a given set of elements is O(log2n) which can be applied in a programming language and that is achieved by Binary Search technique. Binary Search Technique follows Divide and Conquer policy to search an element in a given set of object. The constraint in this Binary Search Technique is that the set of objects that it has to be implemented on should be already sorted. Now here the problem arises on sorting the words in the source document. Like searching, sorting also has a theoretical algorithm which takes O(1) complexity. But the algorithm fails while implementing it in any language due to the class (NP-HARD) in which it has been classified. Practically to sort a set of elements, few algorithms in their worst-case takes a minimum complexity of O(nlog2n) [3]. Some of the sorting techniques with a complexity of O(nlog2n) are Merge Sort, Heap Sort, Intro Sort, Time Sort, Binary Tree sort, Patience Sort, Smooth Sort and Tournament Sort. Here in the Plagiarism Detection System we will make use of Merge Sort.

The reason for using Merge Sort over other sorting algorithm is: No matter what the input size is, the complexity of Merge Sort will remain O(nlog2n) Merge Sort is stable In Text Searching method we are also going to make use of Regular Expressions. Regular Expression is a great tool to search text in the registered document. An algorithm will be run over each and every sentence in the submitted document. This algorithm will dynamically generate a regular expression for every sentence in the submitted doc

# Rabin Karp Algorithm

At repair of method brute force can be classified to follow the sequence comparison of pattern character and character text for every attempt. At the comparison process there are four categories

1. From right to left
2. From left to right
3. In specific order4. In any order

Based on four above categories, algorithm Karp-Rabin included into category from left to right. Algorithm Karp-Rabin applies function of hash providing simple method to avoid time complexity O(m2). Than checked position every pattern which there is in text, would more efficiently if done only at pattern wanted. Equality checking between two words applies function of hash. Function of hash must have propertys as follows :

1. Ability of efficient computing
2. High discrimination to string
3. Function of hash (y [ j+1 .. j + m]) must easy to be computing from hash ( y[j ..j+m -1])

hash (y[j + m] ) Algorithm Karp-Rabin has marking as follows:

1. Applies function of hash
2. Phase preproses in time complexity O(m) and constant place.
3. Seeking phase in time complexity O(mn)
4. O(n+m) estimates active time

Function of hash also applied default value index to or key and applied then each time data relating to value or key is taken. In a seeking hence, first time name would be hashing with function of the same hash when save the data (index) causing yields a value which will be compared to at data is index with the value. Hence in general seeking with 10 possibilities (digit 0-9) would be quicker compared to based on 26 possibilities ( character a-z).

### Plagiarism Detection Methods:

Detection of plagiarism in text document with high accuracy is a challenging task. In the past two decades, a large number of methods have been reported by researchers to handle this task. These methods can be classified into eleven distinct categories. Some prominent methods undereach of these categories are discussed next. Also, we have analyse their pros and cons.

1. Character-Based Methods: Most plagiarism detection methods belong to this category. These methods exploit character-based, word-based, and syntax-based features. It utilizes these features to find similarity between a query document and existing documents. However, the similarity between a pair of documents may be estimated using both exact matching and approximate matching. In exact matching, every letter in both the strings must be matched in the same order. Our survey reveals that most detection techniques are developed based on n-gram or word n-gram based exact string similarity finding approach. For instance, Grozea et al. use character 16-gram matching, whereas the authors of use word 8-gram matching. Similarly, some researcher has made an effective use of approximate string matching approach. This string matching shows degree of similarity/dissimilarity between two strings. There are several proximity measures available to support the approximate string matching. One can use string similarity metric or vector similarity metric for the purpose.
2. Vector-Based Method: Here, lexical and syntax features are extracted and categorized as tokens rather than strings. The similarity can be computed using various vector similarity measures like Jaccard, Dice's, Overlap, Cosine, Euclidean and Manhattan coefficients. Our observation is Cosine coefficient and Jaccard coefficients are popular and effective in finding similarity between two vectors. Cosine coefficient in detecting partial plagiarism without sharing documents content. Hence it is useful to detect plagiarism in documents where submission is considered as confidential
3. Syntax-Based Methods: These methods exploit syntactical features like part of speech (POS) of phrase and words in different statements to detect plagiarism. The elements of basic POS tag are verbs, nouns, pronouns, adjectives, adverbs, prepositions, conjunctions and interjections. In the authors use POS tag features followed by string similarity metric to analyse and calculate similarity between texts. The authors of use syntactical POS tag to represent a text structure as a basis for further comparison and analysis i.e., documents containing same POS tag features are carried out for further analysis and for identification of source of a plagiarism
4. Semantic-Based Methods: A sentence may be defined as an ordered group of words. Two sentences may be same but the order of their words may be different. sentence is constructed by just transforming from active voice to passive voice but the semantics of the sentences are same. WordNet is used in this content to find the semantic similarity between words or sentences. The degree of similarity between two words used in knowledge-based measures by Gelbukh is calculated using information from a dictionary. This similarity between two words is used as semantic similarity between two words. In another approach, Resnik used WordNet to calculate the semantic similarity, whereas, Leacock's et al., determine semantic similarity by counting the number of nodes of shortest path between two concepts.

Fuzzy-Based Methods: In a fuzzy-based method, similarity of text such as sentences is represented by values ranging from zero (entirely different) to one (exactly matched). Here, the words in a documents are represented using a set of words of similar meaning and sets are considered as fuzzy since each word of the documents is associated with a degree of similarity. This method is attractive because it can detect similarity between documents with uncertainty. In, a

1. correlation matrix is constructed which consists of words and their corresponding correlation factors that measures the degree of similarity among different words. Then, it obtains the degree of similarity among sentences by computing the correlation factors between pair of words from two different sentences in their respective documents. In, the degree of similarity of two documents or any two Web documents are identified by using fuzzy IR approach. The authors introduce a tool for this purpose. There is another method discussed in which adapts fuzzy approach to find in what extent two Arabic statements are similar. For that they used a plagiarism corpus of 4477 sources statements and 303 query/suspicious statements.
2. Structure-Based Methods: Unlike those methods above, developed based on lexical, syntactic, and semantic features of the text in documents to find similarity between two documents, the structure based method uses contextual similarity such as how the words are used in entire documents. However, our survey can find a few methods of this category. Contextual information is generally handled using tree-structure feature representation as can be found in ML-SOM. In the author detects plagiarism in two steps. First step performs document clustering and candidate retrieval using tree-structure feature representation and second step detects by utilizing ML-SOM.
3. Stylometric-Based Methods: These methods aim to quantify the writing styles of the author to detect plagiarism. It computes, similarity score between two sections or paragraphs or sentences based on stylometric features of the authors. These methods are instances of intrinsic plagiarism. The style representation formula may be writer specific or reader specific. A writer specific style is mostly with author's vocabulary strength or complexity of presenting a document. On the other hand, a reader specific style deals with how a reader can easily understand the texts. One can find usefulness of outlier mining to detect plagiarism in a document under this approach. A detail discussion on Stylometric-Based methods is available in.
4. Methods for Cross-Lingual Plagiarism Detection: Cross-lingual plagiarism detection is a challenging task. It requires in depth knowledge of multiple languages. Finding appropriate similarity metric for such method is also an important issue. This type of methods work based on cross-lingual text features. Various types of these methods include (1) cross-lingual syntax based methods, (2) cross-lingual dictionary based method, and (3) cross-lingual dictionary based methods. A detail survey on Cross-Lingual methods is done in. In, a statistical model is used to evaluate the similarity between two documents regardless of the order in which the

terms appear in suspected and original documents.

1. Grammar Semantics Hybrid Plagiarism Detection Methods: These methods are effective method in plagiarism detection for their use of natural language processing. They are capable of detecting copy/paste and paraphrasing plagiarism accurately. Such methods eliminate the limitations of semantic-based method. A semantic-based method usually cannot detect and determine the location of plagiarised part of the document but such grammar-based method can address this issue efficiently.
2. Classification and Cluster-Based Methods: In information retrieval process, supervised and unsupervised grouping of documents plays an important role. In many research problem such as text summarization text classification, and plagiarism detection, classification and clustering are useful in reducing the search space during the information retrieval process. It helps in reducing the document comparison time significantly during plagiarism detection. Some

Methods use keywords or specific words to cluster the similar sections of documents.

1. Citation-Based Methods: In a novel method is proposed to detect plagiarism in citation basis. This method is a new approach towards detecting plagiarism and scientific documents that have been read but not cited. Citation-based methods belong to semantic plagiarism detection techniques because these techniques use semantics contained in the citation in a document. The similarity between two documents is computed based on the similar patterns in the citation sequences

**CHAPTER 5**

**RESULTS AND DISCUSSIONS**

For those who want to make a report with no plagiarism in it, they should use the plagiarism tool to detect the areas having plagiarism. Plagiarism detector is a windows appliance to which you can have access to the internet. With the invention of the internet and its easy access, it has now become so easy for students to steal other’s writing. But with the help of an online plagiarism tool, one can easily detect extreme resemblances. The program functions in a way that produces a report in which it makes the places of plagiarism apparent to you and provides you a connection with the website source so that it would be easy for you to check. You also check your own work because it may happen that you have unintentionally copied the stuff from any internet website without revealing its source in your work.

# Issues and Challenges:

Based on our survey we observe that in past two decades, a large number of methods and tools have been developed to support fast and accurate plagiarism detection. Most prominent methods have been able to address the major issues related to (i) salient syntactic and semantic feature extraction, (ii) handling of both monolingual and cross-lingual plagiarism detection, and (iii) detecting plagiarism in both text data and program source code with or without using references. However, with the rapid growth of digital technology to support its reproduction, storage and dissemination, some important issues and research challenges are still left unattended. In this section, we highlight some of such issues and challenges that need to be addressed by computer science and linguistic researchers.

1. A detection method for both text data and source code that ensures both proof of correctness and proof of completeness is still missing, and hence an important issue.
2. A proximity measure that guarantees detection of plagiarized text segment(s) in both intrinsic and extrinsic detection framework with high accuracy, is still not available.
3. Developing a cross-lingual plagiarism checking tool that can perform without external references but ensures high accuracy is a challenging task.
4. Developing a repository that maintains references based on author footprints, which is complete and accurate is another challenging task.
5. Developing a plagiarism checker that accepts an idea narrated by user and generates a detail plagiarism report (with similarity if detected from 1%-99%) with correct sources, is an important issue.

**OUTPUT SCREENSHOTS:**



**CHAPTER 6**

**CONCLUSION**

In this project we discuss about plagiarism detection tools and plagiarism checker software mostly useful in the information era. Information technology, computers are the most important discovery of the 20th century, and with the discovery of internet it became more powerful. Now many things we got online. They are easy to use, anybody can click the button and get the relevant information, but with the advantage of ICT there are some disadvantages also, anybody can use these online information to their documents. To check all this literature theft plagiarism detection software is must for every institution and Universities Plagiarism detection is essential for protecting the written work. It is concluded that all institute es and teachers should be aware of plagiarism and anti-plagiarism software. We have designed a simple method which assists us with the detection of instances of plagiarism in assignment of school and college students. Our scheme is easy to adapt for the large variety of programming languages in use, and is sufficiently robust to be highly effective in an educational environment. While having a detection rate as good as other more complex software, it presents its report as a simple graph, enabling large numbers of assignments to be checked quickly and efficiently. By using data mining algorithm and NLP it will provides straightforward documentation which can be used as clear and convincing evidence should a suspected instance of plagiarism be disputed.

**REFERENCE**

1. P. Kudi, A. Manekar, K. Daware, and T. Dhatrak, "Online Examination with short text matching," in IEEE Global Conference on Wireless Computing Networking, Lonavala, India, Dec. 2014, pp. 56–60 https://doi.org/10.1109/GCWCN.2014.7030847.
2. R. Munoz, A. Montoyo, and E. Metais, Natural Language Processing and InformationSystems. Alicante, Spain: Springer, 2011.
3. M. Allahyari et al., "A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques," Jul. 2017, Accessed: Dec. 26, 2020. [Online]. Available: [http://arxiv.org/abs/1707.02919.](http://arxiv.org/abs/1707.02919)
4. K. B. Cohen and D. Demner-Fushman, Biomedical natural language processing.Amsterdam, Netherlands: John Benjamins Publishing Company, 2014.
5. L. Xinwu, "A new text clustering algorithm based on improved k means," Journal ofSoftware, vol. 7, no. 1, pp. 95–101, 2012.
6. P. D. Asanka, "Finding similar files using text mining," in 8th International Conference on Computer Science Education, Colombo, Sri Lanka, Apr. 2013, pp. 431–435, https://doi.org/10.1109/ICCSE 2013.6553950.
7. T. Svadas and J. Jha, "Document Cluster Mining on Text Documents," InternationalJournal of Computer Science and Mobile Computing, vol. 4, no. 6, pp. 778– 782, Jun. 2015.
8. M. J. Basha and K. P. Kaliyamurthie, "An Improved Similarity Matching based Clustering Framework for Short and Sentence Level Text," International Journal of Electrical & Computer Engineering, vol. 7, no. 1, pp. 551–558, 2017.

**APPENDIX I :**

**CODING**

import numpy as np

import glob import os

def levenshtein(seq1, seq2):

size\_x = len(seq1) + 1

size\_y = len(seq2) + 1

matrix = np.zeros((size\_x, size\_y))

for x in range(size\_x):

matrix[x, 0] = x

# row aray with elements of x

for y in range(size\_y):

matrix[0, y] = y

# column array with elements of y

for x in range(1, size\_x):

for y in range(1, size\_y):

if seq1[x - 1] == seq2[y - 1]:

# if the alphabets at the postion is same matrix

[x, y] = min( matrix[x - 1, y] + 1, matrix[x - 1, y - 1], matrix[x, y - 1] + 1 )

else:

# if the alphabets at the position are different matrix

[x, y] = min( matrix[x - 1, y]+ 1, matrix[x - 1, y - 1] + 1, matrix[x, y - 1] + 1)

# returning the levenshtein distance return (matrix [size x - 1, size\_y - 1])

# one for entire folder with masterfile, one for two separate files, one for all files within the folder choice = int (input( "Enter 1 to check for plagiarism in two files"))

k = 0 # to count the number of plagiarised files if (choice == 15):

plag = int(input("Enter the percentage of plagiarism allowed\n"))

path1 = input("Enter the path of the folder to scan:\n") os.chdir(path1)

# opening all text files within the folder and stores them in an array myFiles

= glob.glob('\*.txt') print("\nThe text files available are :\n") print(myFiles)

path = input("\nEnter the masterfile path (not in the same folder):\n") with open(path, 'r') as file:

data = file.read().replace('\n', '')

str1 = data.replace(' ', '')

print("\nPlagiarised files are :")

for i in range(0, len(myFiles)):

with open(myFiles[i], 'r') as file:

data = file.read().replace('\n', '')

str2 = data.replace(' ', '')

if (len(str1) > len(str2)):

length = len(str1)

else:

length = len(str2)

n = 100 - round((levenshtein(str1, str2) / length) \* 100, 2)

if (n > plag):

print(path, "and", myFiles[i], n, "% plagiarised")

k = k +1

if (k == 0):

print("No plagiarised files") elif (choice == 1):

plag = int(input("Enter the percentage of plagiarism allowed\n"))

path2 = input("Enter the path of the first file to scan:\n")

path3 = input("Enter the path of the second file to scan:\n") with open(path2, 'r') as file:

data= file.read().replace('\n', '')

str1 = data.replace(' ', '') with open(path3, 'r') as file:

data = file.read().replace('\n', '')

str2 = data.replace('', '')

if (len(str1) > len(str2)):

length = len(str1)

else: length = len(str2)

n = 100 - round((levenshtein(str1, str2) / length) \* 100, 2)

if (n > plag):

print("\nFor the files", path2, "and", path3, n, "% similarity")

else:

print("\nSimilarities are below the given level")

elseif (choice == 45):

plag = int(input("Enter the percentage of plagiarism allowed\n"))

path1 = input("Enter the path of the folder to scan:\n")

os.chdir(path1)

# opening all text files within the folder and stores them in an array myFiles = glob.glob('\*.txt')

print("\nThe text files available are :\n")

print(myFiles)

print("\n") for i in range(0, len(myFiles)):

for j in range(i, len(myFiles)): with open(myFiles[i], 'r') as file: data = 40

file.read().replace('\n', '')

str1 = data.replace(' ', '')

with open(myFiles[j], 'r') as file: data = file.read().replace('\n', '')

str2 = data.replace(' ', '')

if (len(str1) > len(str2)):

length = len(str1)

else:

length = len(str2)

if (myFiles[i] != myFiles[j]):

n = 100 - round((levenshtein(str1, str2) / length) \* 100, 2)

if (n > plag):

print("For the files", myFiles[i], "and", myFiles[j], n, "% plagiarised\n")

k = k + 1

if k == 0:

print("No documents areplagiarised") else: print("Invalid Input")

##### APPENDIX II

**OUTPUT SCREENSHOT**