Plagiarism Detection System

Abhay Nitin Pai
Department of Computer Science and Engineering
Datta Meghe Institute of Engineering Technology and Research, Wardha, Maharashtra, India

Chinmay Neelmadhav Bhusari
Department of Computer Science and Engineering
Datta Meghe Institute of Engineering Technology and Research, Wardha, Maharashtra, India

Abstract- Plagiarism is an act of copying an idea from any possible source and presenting it without any citations to the origin. Plagiarism has become a serious issue in the Education System today, as it defiles ethical work and degrade the quality of the Education and Research in any university across the globe. The Plagiarism detection System Standard we have developed is a mixture of several well known techniques and algorithms. Also this system can be used as a standard for implementing any plagiarism detection system. The important techniques for this system are Merge Sort, Binary Search and Regular Expressions. Also the essence of ordering these techniques to achieve desirable goal in a short span of time is explained. Use of networking and databases helps in creating a server-client network which is necessary for such systems because the client cannot itself handle huge amount of registered papers in its system. Each and every techniques mentioned above when combined together can make a better and efficient Plagiarism Detection System.

Keywords – Merge Sort, Binary Search, Sockets, Regular Expressions, Text Analysis

I. INTRODUCTION

The act of plagiarism can be done in many ways. Several research has been done in the past decade which determine several techniques commonly practiced. Also the taxonomy for plagiarism has been provided in [1]. Prof. Derborah A. Frincke in his paper "Plagiarism, Education and Information Security" has mentioned few of them [2]. He made a survey according to which 18% of the students in his university under graduate level courses were found guilty for plagiarism. As of now we know few techniques with the help of which we can try to avoid plagiarism. Manual detection of plagiarism can take humongous amount of time. To reduce the load on the professors in universities, we need a software system that can detect plagiarism .The difficulty that arises while detecting plagiarism is that we are dealing with text on a huge scale. The job of comparing a document thesis with thousands of registered documents to detect plagiarism can outrun the computational power of any system. It may take months to detect plagiarism when we have extensive data at our hand. So what difference can we make? We need a fast and efficient system to detect plagiarism and to reduce the job of computation. Currently a plagiarism detection service website *www.turnitin.com* is one of the most widely used systems in the market.

"The need of computerized systems for plagiarism detection is feasible due to human inability to process large documents and to retrieve all suspected parts and original sources."[1]

Plagiarism Detection System mentioned below is a mixture of several efficient algorithms in order to reduce the complexity. Laying down basic algorithms in an orderly manner can help a lot to reduce overall complexity of the system. The system mentioned below can be implemented in any of the well known language and database. Modules that are being added in this system are a follows

- **Module 1**: To cross check the title of the submitted document files with the database and identify the author of the paper.
- **Module 2**: To find the percentage and a logarithmic value to number of words matched in the submitted document.
- **Module 3**: To detect the number of sentences matched in the submitted document.
- **Module 4**: To detect paraphrasing.
- **Module 5**: Checking the citation of the submitted document.

II. TEXT SEARCHING METHODS

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 $O(n^n)$ 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 $O(\log_2 n)$ 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 $O(n\log_2 n)$ [3]. Some of the sorting techniques with a complexity of $O(n\log_2 n)$ 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 $O(n\log_2 n)$
- Merge Sort is stable

In Text Searching methods 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 document.

III. MODULES

The Plagiarism Detection System to be implemented is divided into several modules. These modules just focus on Detecting Plagiarism rather than the other aspects of the System like server-client networking and security. Several words make a sentence, sentences makes a paragraph and paragraphs make a document. English language is very vast and rich in vocabulary, grammar and words. Using them wisely and appropriately can make a good thesis. But plagiarism can take place using vivid methods as mentioned above. Thus we have divided the modules with respect to the units in the English language such as words, sentences and paragraph. The processing of alphabets in this case is redundant. The system does include other smaller units as mentioned above but detecting plagiarism is the important task that we have to implement. The detailed explanation of each of these modules is given below:-

A. **Module 1:** To cross check the title of the submitted document files with the database and identify the author of the paper

As per IEEE standards no two papers can have the same title unless they have the same author , thus if the title matches then the author of the paper submitted is cross checked with the author of the paper in the database. If the second procedure on inception also gives positive result then the paper is accepted with a new entry with the same title as an update in the author's account. But if the title matches and the author does not match then the paper is rejected.

The further implementation of this module deals with the condition that if the suspected paper is new i.e. in cross checking the title we do not get a positive result. If such is the case then the paper is first checked for plagiarism then after the authenticity of the paper has been ensured the paper is added in the database and when other papers are cross checked for plagiarism then the paper recently added is also cross referenced with for plagiarism.

B. **Module 2:** To find the percentage and a logarithmic value to number of words matched in the submitted document.

Manipulation of words from a registered paper is one of the plagiarism techniques. Changing the voice (active to passive and vice versa), manipulating the words in the sentences, adding words to the sentences to make it look different than the original one, etc are the techniques used by the culprit to make the paper look like an original work

which is not acceptable. In order to remove these anomalies we have build this module. Text searching technique mentioned above comes handy in this module as well as in the upcoming modules.
Let us assume that we have 'X' number of registered documents say $\{D_1,D_2,D_3,\ldots..,D_x\}$, in our database which are free from plagiarism and each document contains 'n' number of words. Thus total number of words can be given as 'Xn'. This shows us that we need to sort 'Xn' number of words for every document that will be submitted to the server. By using the Merge sort we can get the least complexity of

$$\sum \Theta(n.\log(n)) \approx \Theta\ (xn.\log_2(xn))$$

This complexity can be further reduced by selective approach. The registered papers can be divided into finite number of groups and accordingly the submitted paper can be checked against the papers of the group which it belongs to.
This module has to return the amount of words matched from a registered paper. These words can either be deliberately added in the submitted document (which can be a case of paraphrasing) or unintentionally. But a certain upper bound must be set in order to maintain the proficiency of the respective topic of research. In this module a logarithmic value and a percentage of the number of words that have been matched are taken into consideration. This is because the number of words in the submitted document can be huge and taking a direct value cannot be appropriate. Thus percentage to the words copied or a logarithmic value of the words that are matched comes handy and also it is easy to understand.

C. **Module 3:** To detect the number of sentences matched in the submitted document.
A case of plagiarism can take place when the author completely copies certain number of sentences from the registered document and place these sentences directly to the document which he will submit. To detect this case of plagiarism, we have built this module. This module is mainly dependent on Regular Expressions.
A regular expression is a set of pattern matching rules encoded in a string according to certain syntax rules [5] [6]. For every sentence in the submitted document, we need to build a regular expression dynamically in order to match it with the registered documents. The algorithm for building dynamic regular expression string is given as follows

```
algo:(String) PatternCreate(String x) // Returns a String and takes String x as an argument
{
        string string_to_return; //initialize a string which has to be return
        while(next word in x does not ends with '.') //Parse the string from left to right until
        {                                             // a period is encountered
                read next word --> z;
                char firstChar = z[0]; //take the first alphabet from the next word
                append "[firstChar]\\w+" to string_to_return; //add first character to the string
                if(z does not ends with '.') //if not the last word
                {
                        append "\\s" to string_to_return; //append space
                }
        }
        return string_to_return;
}
```

Let us assume that submitted document contains 'n' sentences and every sentence contains 'm' number of words. Thus we need to create 'n' number of regular expressions and these regular expressions will be matched with each of the registered document in the database. The complexity of the algorithm mentioned above will be O(m). Overall there will be 'nm' number of words in the submitted document. Thus to create a set of regular expression for the submitted document will be O(nm). Finally we need to match this set of regular expression with each document in the database.

D. **Module 4:** To detect paraphrasing
A paraphrase is a restatement of the meaning of a text or passage using other words. Paraphrasing can be done in several ways to surpass the basic plagiarism test. Detection of paraphrasing is the most difficult task in Plagiarism Detection. Techniques for paraphrasing need some kind of intelligence to use the ideas and the concepts from a registered paper and presenting it without a citation. Thus this technique can also be considered as plagiarism.
The intelligence behind paraphrasing can cost huge amount of computational power and resources for detecting a plagiarism in the paraphrased sentences. The techniques for plagiarism can be implemented in any of the following cases

1. Most widely used technique [2] is making the use of a simple text editor, an internet and a thesaurus. First of all, the author copy down his required contents and paste it in the text editor. With the help of the thesaurus, he can massage the contents in the notepad by replacing few words.
2. Changing the voice of the sentence.
3. Changing the order of the words from the source.
4. Understanding the idea or the concept from the registered paper and writing it down in his own words.
5. For local paper work if language is not a barrier, then the author can translate a registered paper in his own language.
6. Other unknown techniques.

It is almost impossible to detect paraphrasing from the last technique. The author can apply his own intelligence and paraphrase any portion of the registered paper. The use of internet has also increased the rate of plagiarism. By using internet, paraphrasing becomes easy and fast. It becomes difficult even for a human evaluator to detect paraphrasing of such type.

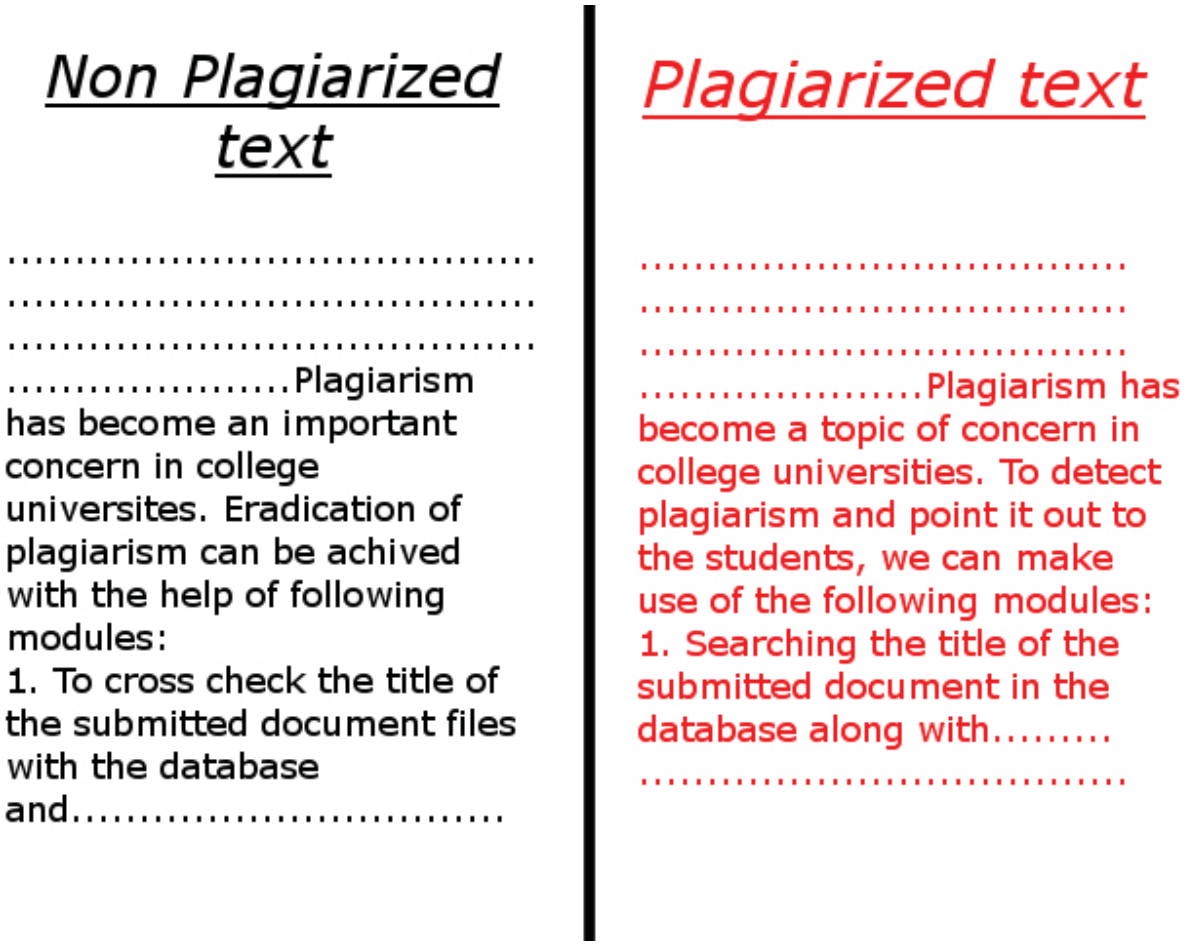

Figure 1: A case of Paraphrasing

Module 2 can come handy in detection of paraphrasing from point number two and three mentioned above. Another technique for detection of paraphrasing can be making a use database and storing all the words from English in it along with their synonyms. For each word from the source and the destination, a mapping can be done to the database and if the word and the synonym matches from the source and the destination, then that particular word can be considered plagiarized. Thus a limit on synonyms matched has to be set for detection of paraphrasing with the help of this technique.

E. **Module 5:** Checking the citation of the submitted document.

A paper needs to have research work in it in order to get accepted in an international level organization. But there may be a case where the paper may contain significant amount of citation and limited amount of research work. Use of citations up to an extend is acceptable but a paper with too many cited sources can be reckoned as plagiarized as the author's original work has been reduced to a minimum in comparison to his research.

This module gives information about the percentage of the citation when compared with the paper and then it is the job of the examining body of the organization to which the paper is submitted to pass the paper as being in order or to reject it.

Another extension of this module is that if the author has used the work of another author has not cited the other author. The percentage of data detected to be plagiarized in this test will be added to the overall plagiarism percentage and may become the last nail in the coffin for the author and his paper may be rejected.

Various Journals and Conferences accept different types of citation rules. The most widely used citation rule is a number in a box bracket and the number which is a reference to the paper mentioned below. (Example: [x] where x is the reference number). Once such a reference is found in the submitted document, we need a crosscheck with the referenced document and also we need to keep a count on the number of citation.

A simple regular expression for the format given above is "[\[\\d\]]". With this regular expression we can retrieve the number of occurrences of the citation.

IV. Grouping of papers

When a paper is submitted to the system, cross checking the paper with all the registered papers can become redundant. Each paper belongs to a field of research work. The registered papers can be divided into a finite number of groups so that the processing load on the server can be reduced.

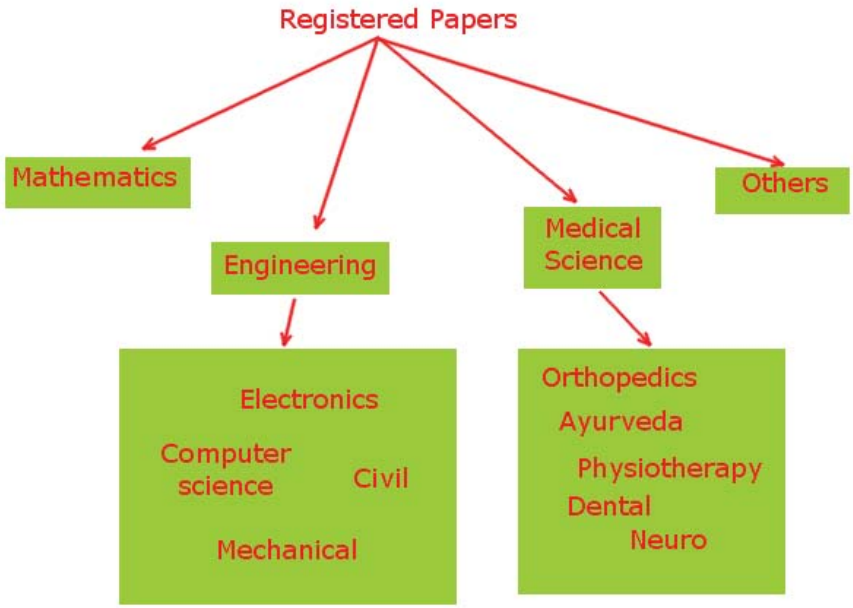

Figure 2: Grouping of Registered Paper

V. CLIENT-SERVER PROGRAMMING

A client system itself cannot afford the memory and computational power required for detecting plagiarism. Storing the huge collection of registered papers in a database for a client system is redundant and meaningless. A server with high computation power and capacity to store large database, is needed in order to remove the entire resource anomaly on the client side.

For creating a server-client network, we need to create a socket connection. The java built in repository has this feature. The Java classes “java.net.Socket” and “java.net.ServerSocket” are needed in order to create a server-client network in Java. A reliable connection is preferred over a non-reliable connection because file transfer from the client to the server should be done reliably, without any loss in the data. The 5 modules mentioned in the figure below runs on the server. Also the server needs to maintain huge database to store the upcoming non plagiarized paper and also to cross check it with the other papers that are already stored in the database.
Two different enhanced models can be built up to achieve faster performance. Implementing a server-client network for Plagiarism Detection should be efficient enough to serve several clients at a time. Thus the implementation of Multi-Threading becomes necessary. Also maintain a single server can cost network delay to the clients. Thus a many-to-many client-server network can help to remove the network delay anomaly.

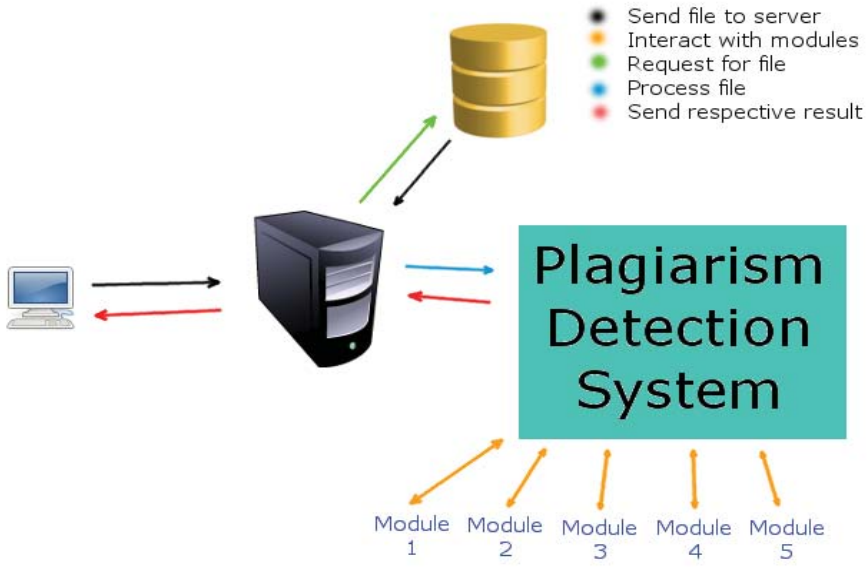

Figure 3 : An overview of PDS at a Server-Client level

VI. MLTITHREADING FOR FASTER PERFORMANCE

With the help of a single socket connection, the server cannot serve multiple clients at a time. The server needs to accept all the authenticated users and serve them at a time. To achieve this, a concept of Multi-threading comes handy. For each connection a thread will be created by the server and the complete process of Plagiarism Detection will be served for each incoming client.
The server has its own database which helps in detecting plagiarism. Multi-Threading cannot affect the server’s database because most of the operations will be reading operations that will be performed on the database. Multi-Threading can increase the performance of the system, serve several clients at a time and also use the resources on the server to an optimum level.

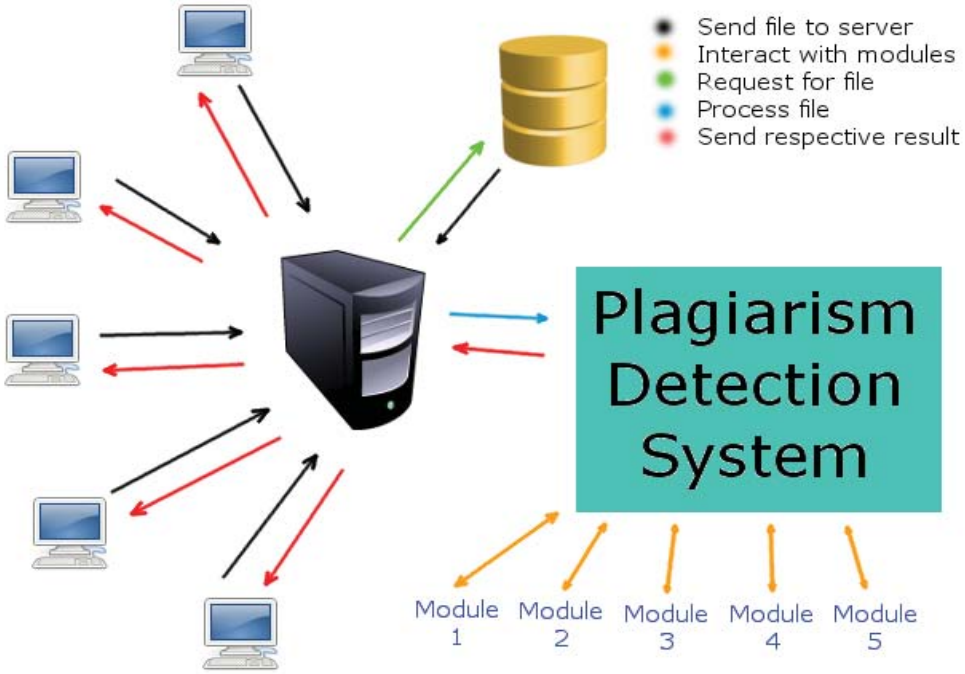

Figure 4: An overview of PDS with Multi-Threading

In Java, Multi-Threading can be achieved with the help of two classes in Java repository. They are "java.lang.Runnable" and "java.lang.Thread". The server can make use of these classes to implement Multi-Threading and serve the authenticated incoming users.

IV.CONCLUSION

This complete paper is has been built up using techniques from Design and Analysis of algorithms, Computer Networks, Database Management and also from Theory of Computation to a certain extent. The best available techniques from these topics have to jotted down and used in an acceptable efficient way to detect plagiarism. Considering this layout as a standard any user or author can create his or her own standard for detecting plagiarism. Plagiarism Detection can cost huge amount of computation power and also there can be several ways in which the author can surpass the plagiarism test. Manual detection of plagiarism is always reliable. But certain amount of load can be reduced by the Plagiarism Detection System Standard given in this paper.

REFERENCES

[1] Salha M. Alzahrani, Naomie Salim, and Ajith Abraham, "Understanding Plagiarism Linguistic Patterns, Textual Features, and Detection Methods" IEEE Transactions on Systems, Man, And Cybernetics—Part C: Applications And Reviews, Vol. 42, No. 2, 1094-6977, March 2012

[2] Deborah A. Frincke,"Plagiarism,Education and Information Security", IEEE Computer Society, 1540-7993,2007

[3] Donald E. Knuth, "The Art of Computer Programming", Volume 3, Chapter 5, Addison-Wesley publication, 1968

[4] Daniela Chuda, Pavol Navrat , Bianka Kovacova, and Pavel Humay ,"The Issue of (Software) Plagiarism: A Student View", IEEE Transaction on education, Vol. 55, No. 1, February 2012

[5] "Chapter 9: Regular Expressions". *The Open Group Base Specifications Issue 6, IEEE Std 1003.1, 2004 Edition*. The Open Group. 2004

[6] Sharon Biocca Zakhour, Scott Hommel, Jacob Royal, Issac Rabinovitch, Tom Risser and Mark Hoeber, "The Java Tutorial: A Short Course on the Basics", 4th Edition, Addison-Wesley Professional