

Extrinsic Plagiarism Detection Using Fingerprinting

¹Siddharth Tata, ²Suguri Charan Kumar, ³Varampati Reddy Kumar

^{1,2,3}Dept. of ECE, GITAM University, Hyderabad, India

Abstract

Plagiarism, in recent times, is one of the serious problems due to which the uniqueness in the writings of different authors is being impaired. Plagiarism detection systems have to be developed to help preserve an author's identity, as there is a lot of effort put in by him to write a document. The aim is to study and develop an automated tool to detect cases of plagiarism in a document through our project.

Extrinsic plagiarism is where a suspicious document is compared with a given set of source documents and phrases, sentences, etc. which appear in both the documents. It may not simply mean copying text from a document, but copying it and using it in a modified form with a misconception of not being detected as plagiarism. This system will be able to detect extrinsic plagiarism. The text in the document is first to split into k-grams. The hash values i.e. fingerprints of all the k-grams are then generated using the Karp-Rabin string matching algorithm. A subset of all the fingerprints is calculated using the Winoing algorithm. A similar process is carried out for the suspicious and source documents and then the fingerprints of both documents are compared using Jaccard similarity to get the percentage of plagiarism.

Keywords

Fingerprints, Karp-Rabin String Matching Algorithm, Winoing algorithm, Jaccard Similarity, K-gram.

I. Introduction

The widespread use of computers and the advent of the Internet has made it easier to plagiarize the work of others. Plagiarism is the "wrongful appropriation" and "stealing and publication" of another author's "language, thoughts, ideas, or expressions" and the representation of them as one's original work. Most cases of plagiarism are found in academia, where documents are typically essays or reports. However, plagiarism can be found in virtually any field, including novels, scientific papers, art designs, and source code. Plagiarism is considered academic dishonesty and a breach of journalistic ethics. It is subject to sanctions like penalties, suspension, and even expulsion. Recently, cases of 'extreme plagiarism' have been identified in academia. Plagiarism detection is the process of locating instances of plagiarism within a work or document.

Detection of plagiarism can be either manual or software-assisted. Manual detection requires substantial effort and excellent memory and is impractical in cases where too many documents must be compared, or original documents are not available for comparison. Software-assisted detection allows vast collections of documents to be compared to each other, making successful detection much more likely.

A. Problem Statement

The aim is to develop software which will detect cases of plagiarism given a suspicious document. Plagiarism is an act of presenting another person's work or idea as your own. There are two different types of plagiarisms i.e. intrinsic plagiarism, where the whole document is written by one person and a small part of it is written by another and extrinsic plagiarism, where a suspicious

document has sentences and ideas which are copied from different source documents.

Extrinsic plagiarism is the most widely found plagiarism in academics, wherein students copy assignments and projects given by professors, either from the Internet or from fellow students. The project provides a convenient way for university professors to upload assignments of students and compare them for chances of plagiarism with the assignments of other students and display the result in the form of the percentage of plagiarism the suspicious file has when compared to all other files in the local database.

B. Contributions

The contributions of this work are as follows:

- An assessment method that evaluates student's assignments automatically for plagiarism which replaces the tedious job of teacher's manual evaluation.
- Satisfying student-teacher marks expectations through this system by detecting plagiarism in assignments.

II. System Design

A. Architecture

To execute a web application, we need a centralized server. In this application, the program gets executed in the localhost only. This was made possible with the help of Tomcat server. The database used in this was MySQL. All the files pertaining to the user are stored here. This application comprises a two-tier architecture. When we run the program it first takes the input from the user, the files he wants to check for plagiarism. Corresponding files are retrieved from database and compared with the file uploaded by the user and results are displayed.

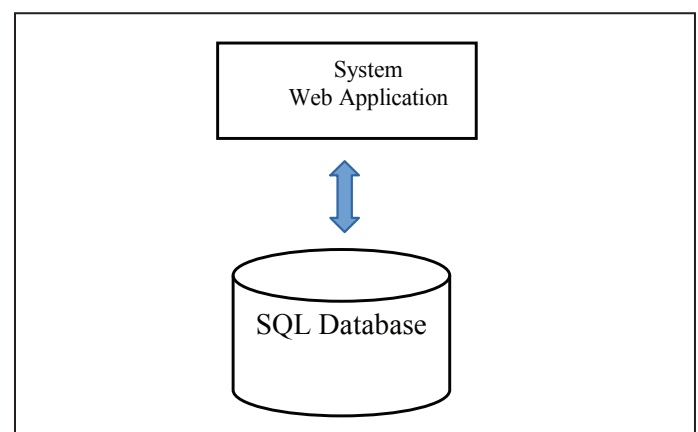


Fig. 1: Represents the System Architecture Design of the Web application

III. Implementation

A. User modules

The following are the steps for the user module:

- Selecting category
- Uploading files
- Getting the percentage of plagiarism

1. Selecting Category

When the user opens the index page, he can choose any of the two options i.e. File to File or Local Database categories. File to file category allows the user to compare any two files at a time whereas the local database category allows the user to compare one file with all other files present in the database pertaining to the same topic

2. Uploading Files

If the user chooses the file to file category, he will be directed to a screen where he will be provided with an option to upload two different files for comparison. If he chooses a local database, he will be provided a screen to upload a single file to be compared with all other similar files in the database.

3. Getting Percentage of Plagiarism

After uploading the files, the files go through the evaluation process i.e., generating n-grams, calculating hash values using the Karp-Rabin algorithm and selecting specific hash values using a winnowing algorithm and calculating similarity percentage between files using Jaccard similarity. Finally result is displayed to the user in the form of percentage of plagiarism present in the source file.

(i). Implementation Modules

The implementation module consists of the following:

- Generating k-grams of the file.
- Generating hash values for each of the k-grams.
- Selecting a subset of all the hash values.
- Comparing hash values of the two files.

(ii). Generating k-grams of the file

First, a pdf file from the database is retrieved and the special characters from the file are removed.

Ex: "This is a sample example". After removing special characters (here spaces) the string/file becomes: "This is a sample example".

After doing so, k-grams are generated by using the substring method in Java.

Let $k=2$ for a sample string. Consider the same string as before. Now, 2-gram strings will be: "Th" "hi" "is"....."pl" "le".

(iii). Generating hash values for each of the k-grams

Karp and Rabin's algorithm for fast substring matching is apparently the earliest version of fingerprinting based on k-grams. Their problem, which was motivated by string matching problems in genetics, is to find occurrences of a particular string s of length k within a much longer string. The idea is to compare hashes of all k-grams in the long string with a hash of s . However, hashing strings of length k is expensive for large k , so Karp and Rabin propose a "rolling" hash function that allows the hash for the $i+1^{\text{st}}$ k-gram to be computed quickly from the hash of the i^{th} k-gram. Treat a k-gram $c_1 \dots c_k$ as a k-digit number in some base b . The hash $H(c_1 \dots c_k)$ of $c_1 \dots c_k$ is this number:

$$c_1 * b^{k-1} + c_2 * b^{k-2} + \dots + c_{k-1} * b + c_k$$

To compute the hash of the k-gram $c_2 \dots c_{k+1}$, we need only subtract out the high-order digit, multiply by b , and add in the new low

order digit. Thus, we have the identity:

$$H(c_2 \dots c_{k+1}) = (H(c_1 \dots c_k) - c_1 * b^{k-1}) * b + c_{k+1}$$

Since b^{k-1} is a constant, this allows each subsequent hash to be computed from the previous one with only two additions and two multiplications.

Now, for the generated k-grams hash values will be generated using the following code:

```
int prime=101;
char[] a= s.toCharArray();
int hash=0;
for(int i=0;i<s.length();i++)
{
    hash+=a[i]*Math.pow(prime,i);
}
return hash;
```

Here, s is the k-gram string and for each letter in the string, the hash value is calculated and is summed up and stored in the variable named hash. This hash will be returned.

(iv). Selecting a subset of all the hash values

We use the Winnowing algorithm to generate a subset of all the hash values generated in the previous step. Following are the steps involved in the algorithm:

The hashes are divided into windows of a particular size.

- For example: If the hash values generated using the Karp-Rabin string matching algorithm are 12, 34, 45, 29, 12, 9, 45. The window size is 3, then the windows generated for these hashes will be (12, 34, 45), (34, 45, 29), (45, 29, 12), (29, 12, 9), (12, 9, 45).
- From each window, the hash value which is least from the right side is selected. If the same minimum hash is encountered as in the previous window, then nothing is selected in that window. This way the least number of hashes will be generated for efficient comparison.

In the previous example, the hashes generated will be 12, 29, 12, 9.

(v). Comparing hash values of two files using Jaccard similarity

After generating fingerprints for passages, we need a way to compare the fingerprints to determine whether or not plagiarism might be present. Jaccard similarity is a very common set similarity measure that is used in a wide variety of applications.

It is defined as

$$\text{Jaccard}(A, B) = (A \cap B) / (A \cup B)$$

Where A is the suspect fingerprint and B is the source fingerprint. Then the percentage of plagiarism between the two files will be given by the following formula:

$$\text{Percentage} = ((C * 100) / (A + B - C));$$

Where C is the number of common hashes between the two files.

IV. Results and Discussions

A. Final Output Screenshots

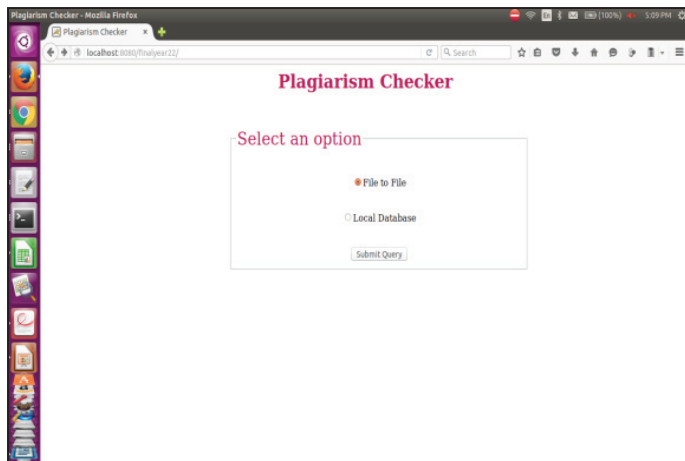


Fig. 2: Shows the Category Selection Screen

This is the starting page where users can choose either of the two options i.e. file to file or local database.

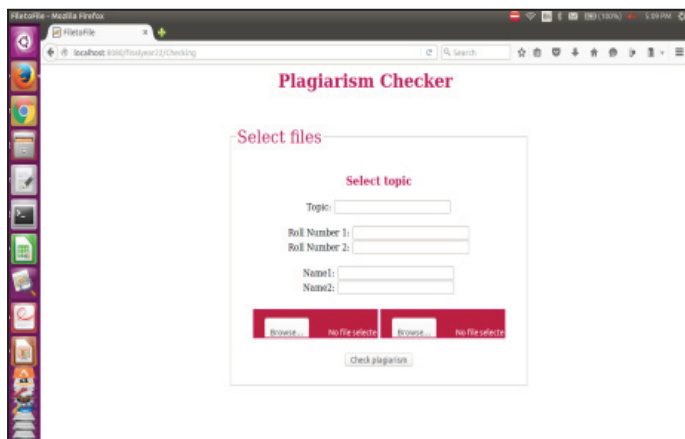


Fig. 3: Shows the file to File the Checker Screen

If the user selects the file to file category, he will be directed to this screen. On this page, the user will be entering details about the two files like names and roll numbers of the two students and the topic to which the two files belong to and then he will be uploading two files.

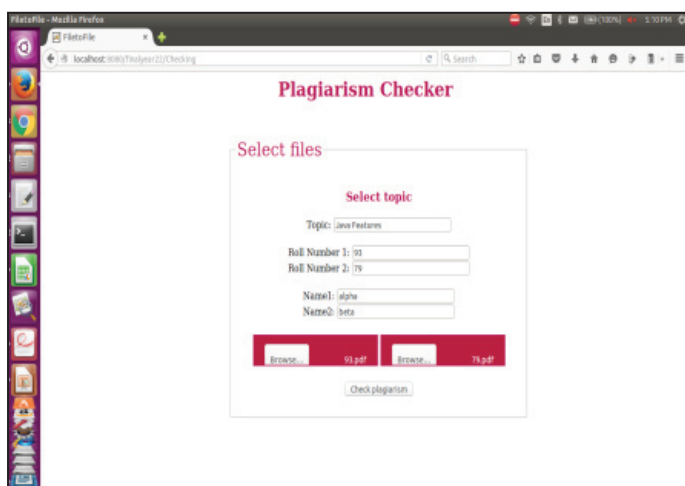


Fig. 4: Shows the File to File the Screen After the User Fills all the Details

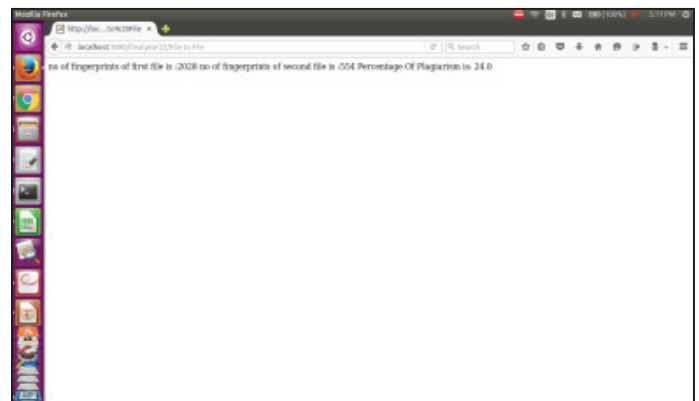


Fig. 5: Shows the Results of the File to file a Plagiarism Check

Finally, output, i.e. the percentage of plagiarism between the two files will be displayed on a new screen.

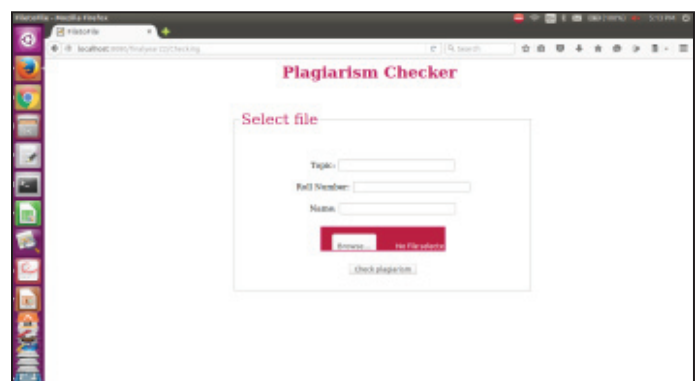


Fig. 6: Shows the Screen to Fill Details for the Local Database Category

The user needs to fill the topic name, roll number, and name of the student and has to upload a single file.

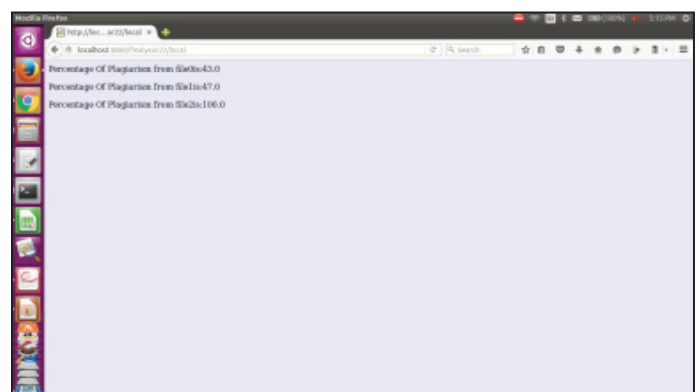


Fig. 7: Show the Results After Comparison

Finally given file will be compared with all the related files in the database and the percentage of plagiarism with each file will be displayed.

B. Discussion on Results

Table 1: Shows the variation in plagiarism percentage with constant window size and change in n-gram size.

N-gram size	Size of window	Percent of Plagiarism
2	3	45.6
3	3	42.7
4	3	40.3

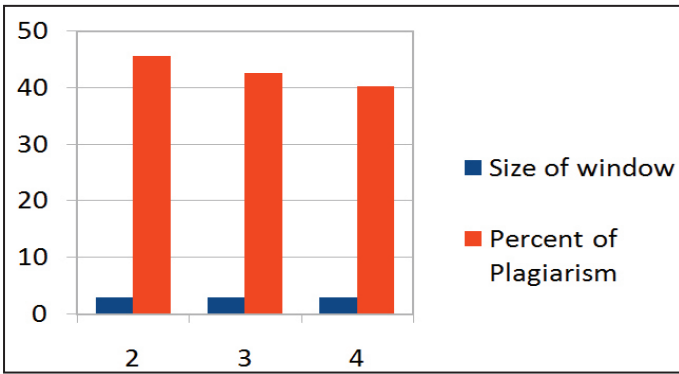


Fig. 8: Shows the Variation in Plagiarism Percentage when the Window Size Remains Fixed but the Size of k-gram Varies.

Table 2: Shows the Variation in the Percentage of Plagiarism with a Change in Window Size and Constant n-gram Size.

Window size	N-gram size	Percent of Plagiarism
4	3	35
5	3	34.8
6	3	34.2

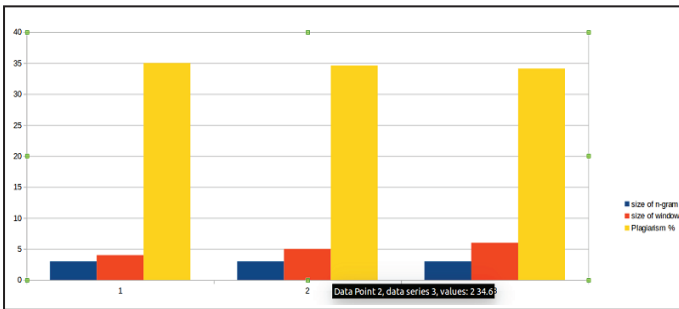


Fig. 9: Shows the Variation in Plagiarism Percentage when the Size of k-gram Remains Constant but the Window Size Varies.

Roll No. 1	Roll No. 2	Percentage of plagiarism
81	108	56
81	91	63
81	87	54
81	68	22
81	64	60
81	66	55
81	76	57
81	72	58
81	65	49
81	112	51
81	317	43
81	93	33
81	97	32
81	83	57
81	79	44
81	74	58
81	111	74

- A student with the least percentage of plagiarism
- A student with the highest percentage of plagiarism

Table 3: Shows the Percentage of Plagiarism from the Assignments of Students

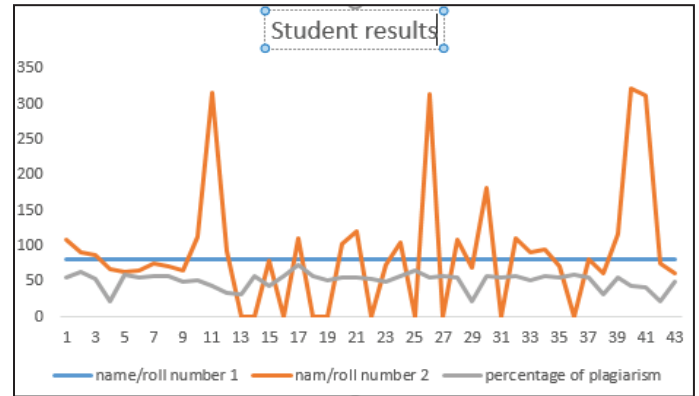


Fig. 10: Shows the Variation in Plagiarism in the Assignments of Students.

V. Conclusion and Future Scope

The research proposes the combination of different techniques to produce an approach for detecting plagiarism in any field. In recent years, a number of automated online tools for plagiarism are available, but there are very few open-source tools to help professors at the graduate level to check the assignments of students of plagiarism. Plagiarism detection provides a very useful means of testing the knowledge of the students and helps him improve his originality by thinking himself. The salient feature of the project (from the pedagogical perspective) is that it can help the professor achieve immediate results regarding the extent of plagiarism in assignments of students regardless of the size of the class. The professor’s burden of going through each assignment is reduced in this way.

The future scope of this project can be to check for content on the Internet i.e. content related to the topic of the assignment can be retrieved from the Internet and that can also be used as a reference to check if the student has copied from there. Another extension would be to check not only text files, but also programs are written by students for plagiarism, but this would need different approaches.

References

- [1] Saul Schleimer, Daniel S. Wilkerson, Alex Aiken-“Winnowing: Local Algorithms for Document Fingerprinting.” pp. 3-5.
- [2] Brenda S. Baker, "On finding duplication and near-duplication in large software systems. In L. Wills, P. Newcomb, E. Chikofsky, editors, “Second Working Conference on Reverse Engineering”, pp. 86–95, Los Alamitos, California, 1995. IEEE Computer Society Press.
- [3] Brenda S. Baker, Udi Manber, “Deducing similarities in java sources from bytecodes”, In Proc. of Usenix Annual Technical Conf., pp. 179–190, 1998.
- [4] Sergey Brin, James Davis, Hector Garcia-Molina, “Copy detection mechanisms for digital documents”, In Proceedings of the ACM SIGMOD Conference, pp. 398–409, 1995.
- [5] Andrei Broder, “On the resemblance and containment of documents”. In SEQS: Sequences ’91, 1998.
- [6] Andrei Broder, Steve Glassman, Mark Manasse, Geoffrey Zweig, “Syntactic clustering of the web”. In Proceedings of the Sixth International World Wide Web Conference, pp. 391–404, April 1997.

- [7] Ramesh R. Naik, Maheshkumar B. Landge, C. Namrata Mahender, "A Review on Plagiarism Detection Tools". pp. 16-21, 2015.
- [8] Barnbaum, C., "Plagiarism: A Student's Guide to Recognizing It and Avoiding It", Valdosta State University, [Online] Available http://www.valdosta.edu/cbarnbau/personal/teaching_MISC/plagiarism.htm (Accessed 23 January 2006).
- [9] Liles, Jeffrey A., Michael E. Rozalski, "It's a Matter of Style: A Style Manual Workshops for Preventing Plagiarism", College & Undergraduate Libraries, 11 (2), pp. 91-101, 2004.
- [10] Maurer H., Kappe F., Zaka B., "Plagiarism- A survey. Journal of Universal Computer Science", 12,8, pp. 1050-1084, 2006.
- [11] Mohamed Elkhidir, Mohannad M. Ibrahim, Tarig A. Khalid, Shawgi Ibrahim, Mohamed Awadalla, "Plagiarism Detection using Free-text Fingerprint Analysis", 2015.
- [12] Michael Tschuggnall, Guntther Specht, "Detecting Plagiarism in Text Documents through, s.l. Technikerstraße Innsbruck".
- [13] Roger S. Pressman, "Software Engineering –A Practitioners Approach", 7th Edition, Pearson Education, India, 2010.
- [14] Subramanyam Allamraju, "Professional Java Server programming", J2EE 1.3 Edition, CeditBuest, Apress Publications.
- [15] Deitel, Deitel, Goldberg, "Internet & World Wide Web How To Program", Third Edition, Pearson Education, 2010.



Mr. Tata Siddharth has pursued his undergraduate studies in Electronics and Communications Engineering at GITAM University, Hyderabad in the year 2017. He has been working in Cognizant Solutions on UI Development. His penchant towards the Machine Learning and Computer Science course has helped him in deep research and execution of several projects successfully. He had researched and built a Weather

Prediction Model, which has improved the accuracy of predicting the weather changes by 15 %. He has built an efficient model that predicts a student's performance in a course, using Logistic regression and Feature Engineering. He is currently working on the development of a complete website with user interactions with a dynamic, suggestive response by the developer contingent of the company to its client.



Mr. Suguri Charan Kumar has pursued his undergraduate in Electronics and Communications Engineering at GITAM University, Hyderabad. He is currently working in Cognizant Technology Solutions as a Programmer Analyst. With a predilection towards the Data analysis and Data Management, he is relentlessly researching in the field of Data Science. He had successfully implemented a Low Leakage 32 Bit Parallel Prefix Adder during his

Sophomore years. He is currently working on the development and enhancement of Data Management models in his company.



Mr. Varampati Reddy Kumar Reddy has pursued his undergraduate in Electronics and Communication Engineering at GITAM University, Hyderabad. He is currently working as a Programmer Analyst in Cognizant Technology Solutions. His proclivity for hard work and interest to pursue new skills has helped him in several research works during his Engineering. He has successfully. He is currently working on the development of ETL

tool by using ABAP which increases the efficiency in maintaining Data.