FIRAT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
TÜRKİYE

COMBINING APPROXIMATE STRING MATCHING ALGORITHMS AND TERM FREQUENCY IN THE DETECTION OF PLAGIARISM

Zina BALANI

Master's Thesis

DEPARTMENT OF SOFTWARE ENGINEERING

JANUARY 2022

FIRAT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
TÜRKİYE

Department of Software Engineering

Master's Thesis

COMBINING APPROXIMATE STRING MATCHING ALGORITHMS AND TERM FREQUENCY IN THE DETECTION OF PLAGIARISM

Author
Zina BALANI

Supervisor
Prof. Dr. Cihan VAROL

JANUARY 2022
ELAZIG

FIRAT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

T Ü R K İ Y E

Department of Software Engineering

Master's Thesis

Title:	Combining Approximate String Matching Algorithms and Term Frequency in the Detection of Plagiarism
Author:	Zina BALANI
Submission Date:	10 November 2021
Defense Date:	14 January 2022

THESIS APPROVAL

This thesis, which was prepared according to the thesis writing rules of the Graduate School of Natural and Applied Sciences, Fırat University, was evaluated by the committee members who have signed the following signatures and was unanimously approved after the defense exam made open to the academic audience.

		Signature
Supervisor:	Prof. Dr. Cihan VAROL Sam Houston State University, Faculty of Science & Engineering	Approved
Chair:	Doc. Dr. Murat KARABATAK Firat University, Faculty of Technology	Approved
Member:	Dr. Ogr. Uyesi. Ali ARI Inonu University, Faculty of Engineering	Approved

This thesis was approved by the Administrative Board of the Graduate School on

..... / ..... / 20.....

Signature

Prof. Dr. Kürşat Esat ALYAMAÇ
Director of the Graduate School

DECLARATION

I hereby declare that I wrote this Master's Thesis titled “Combining Approximate String Matching Algorithms and Term Frequency in the Detection of Plagiarism” in consistent with the thesis writing guide of the Graduate School of Technology, Firat University. I also declare that all information in it is correct, that I acted according to scientific ethics in producing and presenting the findings, cited all the references I used, express all institutions or organizations or persons who supported the thesis financially. I have never used the data and information I provide here in order to get a degree in any way.

14 January 2022

Zina BALANI

PREFACE

Various techniques and algorithms have been developed to prevent literature from being plagiarized on the Internet. However, in today's developed technology, most of the techniques are discovered by plagiarized text developers, and plagiarism is still a real challenge. In this study, a hybrid algorithm consists of edit distance based approximate string matching algorithm and term frequency inverse document frequency was developed to increase the plagiarism detection rate. The powerful hybrid algorithm can successfully recognize the exact and disguised words in both everyday language and source code.

I have the highest gratitude to God for blessing me with skills that enables me to work in my project. I could not have achieved my current level of success without a strong support group. First of all, I have a special thanks for my dear supervisor Prof. Dr. Cihan Varol that provided advice and guidance throughout the research process and special thanks for my family and friends, who supported me with love and understanding. Thank you all for your unwavering support.

Zina BALANI
ELAZIG, 2022

TABLE OF CONTENTS

Page

ABSTRACT

Combining Approximate String Matching Algorithms and Term Frequency in the Detection of Plagiarism

Zina BALANI

Master's Thesis

FIRAT UNIVERSITY
Graduate School of Natural and Applied Sciences

Department of Software Engineering

January 2022, Page: xi + 37

One of the key factors behind plagiarism is the availability of a large amount of data and information on the internet that can be accessed rapidly. This increases the risk of academic fraud and intellectual property theft. As increasing anxiety over plagiarism grow, more observation was drawn towards automatic plagiarism detection. Hybrid algorithms are regarded as one of the most prospective ways to detect similarity of everyday language or source code written by a student.

This study investigates the applicability and success of combining both the Levenshtein edit distance approximate string matching algorithm and the term frequency inverse document frequency (TF-IDF), thereby boosting the rate of similarity measured using cosine similarity. The proposed hybrid algorithm is also able to detect plagiarism occurred on natural language, source codes, exact, and disguised words. The developed algorithm can detect rearranged words, inter-textual similarity of insertion or deletion and grammatical changes. In this research three various dataset are used for testing: automated machine paragraphs, mistyped words, and java source codes. Overall, the system proved to be detecting plagiarism better than the yet alone TF-IDF approach.

Keywords: Approximate, Hybrid, Plagiarism, Similarity, TFIDF

ÖZET

İntihal Tespitinde Yaklaşık Dizi Eşleştirme Algoritmaları ile Terim Sıklığının Birleştirilmesi

Zina BALANI

Yüksek Lisans Tezi

FIRAT ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü

Yazılım Mühendisliği Anabilim Dalı
Ocak 2022, Sayfa: xi + 37

İntihalin arkasındaki en önemli faktörlerden biri, internet üzerinde hızla erişilebilen büyük miktarda veri ve bilginin bulunmasıdır. Bu, akademik dolandırıcılık ve fikri mülkiyet hırsızlığı riskini artırır. İntihal üzerine endişe arttıkça, otomatik intihal tespitine yönelik daha fazla araştırmalar yapıldı. Hibrit algoritmalar, bir öğrenci tarafından yazılan günlük dil veya kaynak kodun benzerliğini tespit etmenin en olası yollarından biri olarak kabul edilir.

Bu çalışma, hem Levenshtein düzenleme mesafesi yaklaşık dizi eşleştirme algoritmasını hem de frekans ters belge frekansı (TF-IDF) terimini birleştirmenin uygulanabilirliğini ve başarısını araştırıp, böylece kosinüs benzerliği kullanılarak ölçülen benzerlik oranını saptamada daha etkili bir yöntem sunmaktadır. Önerilen hibrit algoritma aynı zamanda doğal dilde, kaynak kodlarında, tam ve kılık değiştirmiş kelimelerde meydana gelen intihalleri de tespit edebilmektedir. Geliştirilen algoritma, yeniden düzenlenmiş sözcükleri, metinler arası ekleme veya silme benzerliğini ve dilbilgisi değişikliklerini algılayabilmektedir. Bu araştırmada test için üç farklı veri seti kullanılmıştır: otomatik makine paragrafları, yanlış yazılmış kelimeler ve java kaynak kodları. Genel olarak, sistem intihalleri tek başına kullanılan TF-IDF yaklaşımından daha iyi tespit ettiğini kanıtlamıştır.

Anahtar Kelimeler: Benzerlik, Hibrit, İntihal, TFIDF, Yaklaşık

LIST OF FIGURES

Page

LIST OF TABLES

Page

ABBREVIATIONS

LCS : Longest Common Substring
LED : Levenshtein Edit Distance
PBDE : Passage Boundary Detection and Evaluation
PDLK : Plagiarism Detection Linguistic Knowledge
SVD : Singular Vector Document
TFIDF : Term Frequency Inverse Document Frequency

1. INTRODUCTION

Onward entering the age of digital transmission, simplicity of sharing data across the internet has facilitated to search for online sources for finding information. This carries an increased chance of scientific abuse and educated attribute steal as anxiety about infringement of copyright increase [1]. Content taken directly from someone else's previously published sources is known as plagiarism [2]. Plagiarism is regarded as a serious problem in modern research manuscripts [3].

This work will focus on two plagiarism areas: source code (known as "code clone") and everyday language. Copying a general talk is comparatively hard to detect because of plenty of morphological, syntactic, and semantic characteristics of everyday language. Moreover, copiers wield various paths to overcome plagiarism checkers through replacing actual words with implementing different types of hiding and using smart plagiarism techniques, as well as rewriting, synonym replacement, rephrasing, concept processing, string interpretation, and thought acquiring [4].

To mitigate the negative effects of plagiarism, systems aim to identify cases of theft in paper or documents via comparing a large collection of documents with suspicious documents. Finally, systems detect if the suspicious document has been stolen or similar to each other [5].

There are two essential methods for automatic plagiarism detection. Extrinsic / external plagiarism detection: extrinsic internal plagiarism detection. Intrinsic plagiarism detection only analyzes the input document and finds some parts that have not been created by the same author without performing a comparison with an external corpus. External plagiarism detection requires a reference collection of documents that are considered original. Suspicious documents are compared to all documents in this collection to detect duplicate or approximately matching components in the source document [2, 6]. This study deals with external plagiarism detection methods.

Strings can be identical in two manners. There are cases where they use syntactically the same string, and there are cases where they have just as a semantic definition. The classification of identity measurements is character-based, q-gram, token-based, and mixed measures. Character-based and q-gram compute identity depends on a set of characters in two strings. Token-based measures use white space, line breaks, or punctuation characters to break a string into terms and attributes (known as tokens) and calculate the likeness among two token sequences. Mixed measures are an integration of character-situated measures and token-situated measures [7].

At the beginning of the 20th century, in the early 1970s, scientists have studied a large-scale copy of the program to prevent technology and software. There are several classic systems available for detecting plagiarism in program source code. Plagiarized program code is different than the plain language as different codes can perform the same function. Some smart plagiarists use certain

methods to alter the code. For example, a for loop becomes a while loop or adding a large number of randomly generated intermediate variables [8].

In today's developed technology where techniques of plagiarism detection are available, previous methods and techniques that were used to detect and find measures of plagiarism are no longer in use. Most investigators are now working on the success of hybrid algorithms to find the similarity.

Therefore, in this study the applicability and success of two specific hypotheses are investigated:

- Approximate string matching algorithms can be employed to detect plagiarism.
- Combining approximate string matching with TF-IDF will increase the plagiarism detection rate.

The proposed system can define various shapes of plagiarism such as reorganizing text, source code, inter-textual likeness, and approximately similar. Edit distance is used to find the similarity for mistyped textual information if the resulting degree of identity transcends a predetermined threshold that is greater than or equal to 0.50. Once this threshold is reached, then TF-IDF counts the word as its original, which increases the frequency of the common words. As a result, the rate of plagiarism will be boosted which is measured by Cosine Similarity. The success rate of a hybrid version of approximate string matching and term frequency is investigated and is the results reflected that a robust combined algorithm is capable to find both plagiarisms on everyday language and code clone.

2. REVIEW OF LITERATURE

"Plagiarizing" is stealing and transmitting the thoughts and words of others when done without proper citation of the source. Information theft in the shape of computer data is increasing significantly. This also happens in academia and educational institutions. In particular, the part called plagiarism, specified as a shape of abuse of investigation. Abuse means a structure, deformity, copying, or any other implementation that drifts significantly by widely accepted practices in the discipline or academic field and the investigation community often proposes, conducts, reviews, or reports on research and creative activities [6]. Plagiarism Detection System is the operation of using software to search for cases of plagiarism in an article or document, comparing a large collection of documents with the suspicious article, and eventually detecting whether the suspicious article is copied or unique [5].

2.1. Background History

A wide range of problems can emerge when many students copy someone else's work and submit it as their own. To detect these cheats, the online learning management systems started to add copying determination tools, and when exactly two or sufficiently identical assignments are found a flag is raised for the issue. However, plagiarists use a variety of methods to alter the submitted work to avoid detection by the system. In recent years, the problem of detecting plagiarism in ordinary language has received a lot of investigators' attention. Several plagiarism determination techniques have been released specifically for the English language.

According to [1], one of the oldest methods of detecting plagiarism was introduced by Bird in 1927, who investigated the application of statistical methods to detect plagiarism in multiple-choice answers.

Afterward, during the 1960s techniques advanced and pointed to finding plagiarism in multiple-choice exams. In 1965 after The Russian scientist Vladimir Levenshtein, Levenshtein Distance is advanced, also called edit distance. It's a measure of the similarity between two strings called source string (s) and target string (t). Distance is the number of deletions, additions, or replacements needs to convert s to t [9]. This technique was used for near-duplicate detection for several years.

Back in the 1970s, researchers began working on similarity detection techniques for source code [10]. Since the 1970s, numerous tools have been introduced to measure the similarity of code. They are used to address issues like code duplication detection, software license violations, and software plagiarism [11].

In 1975, Halstead suggested the first algorithm called the attribute enumerating process. The technique computed the statistics of the operators and operands seeming in the original project and operated them as the primary basis for determining the result. [10].

Between the 1970s till mid-1990s was a golden period of developing different methods and algorithms of exact and approximate string matching algorithms [12].

Since the mid-1990s, the focus of research has changed to the study of ordinary language scripts. In 1994 Manber suggested the theory of a resemble fingerprint. The standard command is to measure resemblance among legal papers by string matching. This rule has been accepted by nearly all investigators. Therefore, based on this, investigators raised term frequent occurrence counting, keyword extraction techniques, and accomplished matching by computing hash values for script [10].

Experimental outcomes indicate that a hybrid technique that considers word frequency and character-based word similarity increases matching. The first venture in this direction was found in 2003 by Cohen et al. A measurement format named Soft-TFIDF, which extends the Jaro-Winkler method to combine the frequency weight of a word in a measure of cosine similarity and a measure of CLOSE at the character level [13].

In 2007, according to [1], Dreher proposed using a normalized word vector algorithm to calculate similarity based on VSM (Vector Space Model), which performs synonym generalization for each word. Despite the advent of more advanced approaches, paraphrasing remained difficult.

In 2012, Ekbal, Saha, and Choudhary suggested ways to detect plagiarism. This includes three main steps. First, the primary functions of everyday language handling are performed in preprocessing steps. The next step selects a set of original papers that are identical to the suspect paper. A VSM is used to determine the original document for each suspect document. Eventually, the resemblance of both papers is computed based on the cosine similarity. If the resulting resemblance measurement passes over a predetermined threshold, the document is regarded to be the original document of the suspect document. In the third step, a similar text is found in both source and suspect documents using the n-gram method [14].

In 2018, a system was recommended for Urdu text, using a distance measurement technique, vector space model, and structural alignment algorithm. System Implementation is measured based on machine learning classifiers (Support Vector Machine and Naive Bayes). Preliminary outcomes demonstrate that the output of the suggested technique is ahead of other available versions (i.e. cosine method, simple Jaccard measure) [15].

2.2. Plagiarism Types

Types of plagiarism includes textual plagiarism and source code plagiarism as follows:

2.2.1. Textual Plagiarism Types

Copy/paste is aforethought as genuine plagiarism. Moreover, paraphrasing, plagiarism by translation, and plagiarism of idea are aforethought as cases of intellectual plagiarism. Textual plagiarism occurs in a number of ways that can be divided into five categories [5, 6]:

1. "Copy-Paste": The plagiarist identifies a useful resource and copies part of it into the text that will probably change the author's name with a few edits.
2. Disguised Plagiarism: using our own words to reconstruct others' ideas by simply approving grammar changes, word synonyms, and word additions, deletions, and substitutions.
3. Translation with plagiarism: The ingredient is translated and used without referring to the original property.
4. Use of incorrect references: Adding incorrect or non-existent references.
5. Plagiarism of ideas: This is a kind of plagiarism that is hard to determine as it is not just cheating of the text, but a reconstruction that allows any text changes as long as the meaning of the sentence is preserved.
6. Structural plagiarism: A person's thoughts, the order of discussion, the choice of citations from others, or even footnotes that are used in the same order without giving credit are considered structural plagiarism. This kind of plagiarism is quite hard to check because it requires an observant reading of both texts to detect what occurs.

2.2.2. Source Code Plagiarism Types

To avoid plagiarism detection, plagiarists typically make non-essential changes to the plagiarized code clone. These disguises usually cover the implied tools [10]:

1. Include spaces, and brackets in source code expressions, and comments.
2. Add a declaration of unused variables.
3. Switch the order of the function descriptions in the source code, for example, the order of "A, B, C" in "C, B, A".
4. Add function descriptions that are not needed in the program without a useful function.
5. Rename the custom identifier in the program. Those consumer-specified descriptive contain inconstant names and task names.
6. Add a series of instructions that have no active meaning on the program, like "i = i" or "i ++, i--" etc.

2.3. Types of Plagiarism Detection Approaches

The plagiarism identification is categorized as multilingual and monolingual depending on the homogeneity or heterogeneity of the language of the contrasted text documents [5]. Classification of plagiarism detection methods is shown in Figure 2.1.

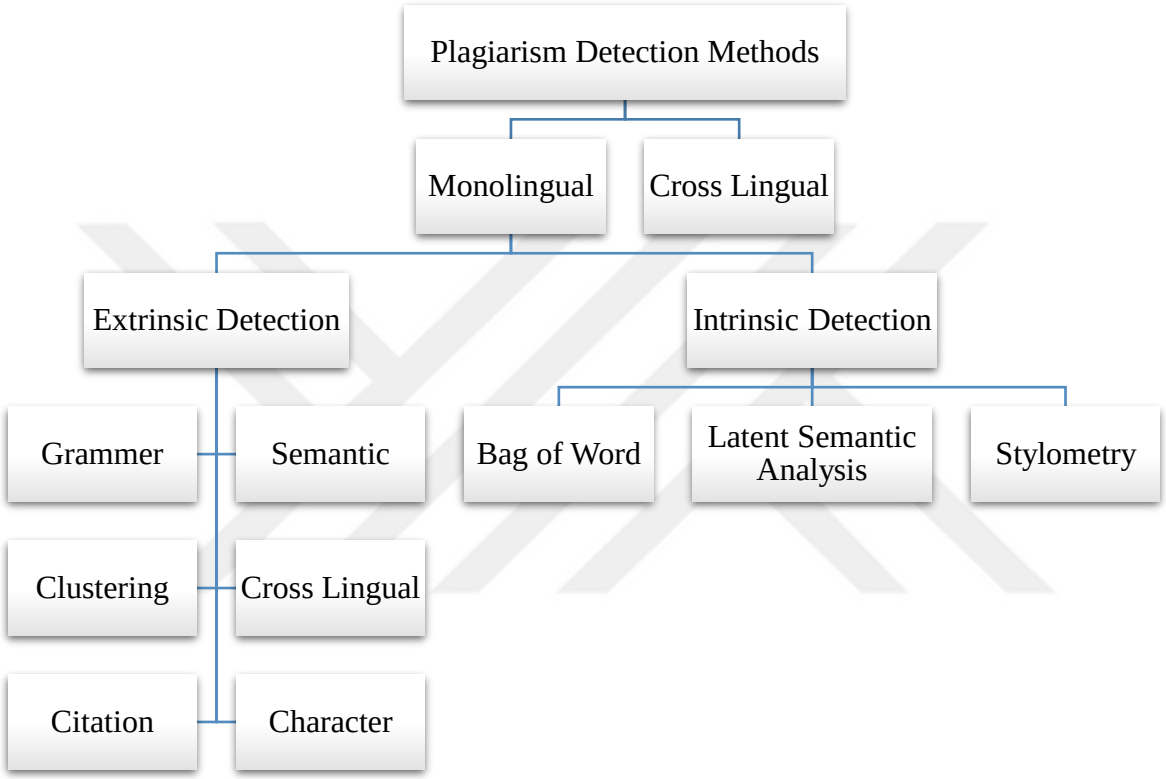

Figure 2.1 Categorization of plagiarism identification techniques

2.3.1. Monolingual

Monolingual plagiarism detection automatically detects and extracts plagiarism in the same language environment (e.g. Turkish-Turkish plagiarism). Almost all plagiarism identification systems are designed for monolingual detection that can be classified according to two earlier functions: extrinsic and intrinsic [16].

Extrinsic Detection

Plagiarism is identified by checking against the substance of submitted investigative articles with the substance of investigative articles formerly issued in different corpora. For this, it is necessary to mention the dataset [6]. The framework of the external plagiarism detection system can be seen in Figure 2.2.

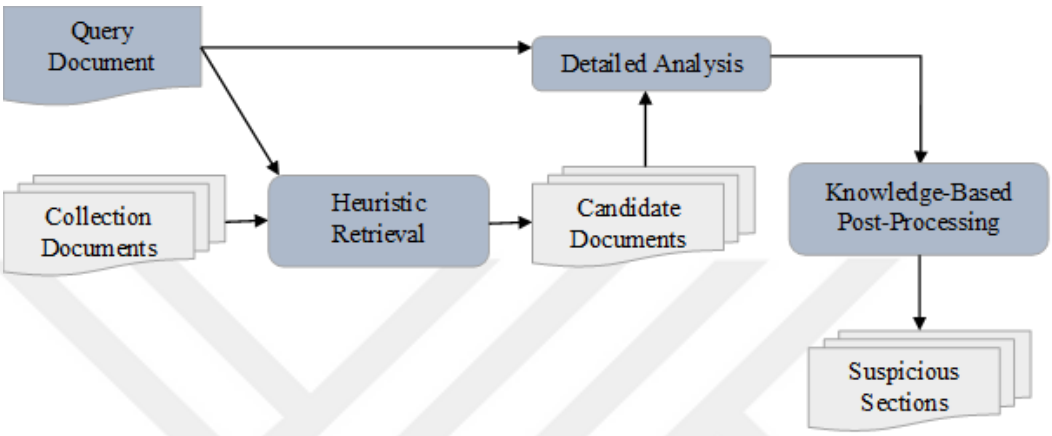

Figure 2.2 Structure of extrinsic plagiarism detection system

The extrinsic structure can be combined as follows: The inputs are a questionaries' document and a mention of addition which include the sources of plagiarism. Three major processes are required. Initial, a few catalogs of candidate documents that could be origins of plagiarism is retrieved from the reference collection using a specific search type. Following, a detailed pairwise characteristic analysis is done to contrast the query document to its candidates using a comparison unit. A knowledge-based post-processing step is carried out to integrate the detected little units into passages or parts and to show the outcome to a person who can determine that plagiarism has occurred. [16].

Thereat six common external plagiarism identification processes, including [6]:

1. **Plagiarism Detection According to Grammar:** This process operates a string-situated matching attitude to identify and measure similarities between official articles in the database. The grammar-based technique can be applied to identify duplicate documents, i.e., it cannot determine plagiarism in the paraphrased article.
2. **Plagiarism Detection According to Semantics:** This process observes identifying resemblances in the use of terms with articles holds in a particular repository using a vector space model. Additionally calculates the number of excess words used in the considered document. However, paraphrased articles do not provide accurate outcomes because the published research treatise cannot find the actual plagiarized part.

3. **Plagiarism Detection According to Clustering:** The cluster-situated Plagiarism determination process uses a grammar-situated method in three major stages: The first stage is pre-selection to use the same fingerprint continuously to narrow the detection range. The second is to discover and combine all the parts between both documents using the geographical location cluster technique. The final stage is named post-processing which creates agreements including several joining faults. There are two classic clustering algorithms applied in fingerprint-based document expression, working with multiple sets of similarity metrics to design a new central calculation method.
4. **Cross-Lingual Plagiarism Detection:** applied to identify suspicious articles copied from other linguistic sources. In this technique, the similarity between the suspect article and the original article is assessed using a statistical model to determine the likelihood that the suspect article is concerned to the source article, in any case of the sequence in which the concepts occur in the suspect articles and originals. This process requires the structures of a multilingual dataset.
5. **Plagiarism Detection According to Citation:** This process is applied to detect scientific articles that are read and used without reference to these documents. In fact, it is one of the semantic plagiarism determination methods as it points to detecting semantic content in citation utilized in a textual scientific article. It aims to detect same the model in the citation progression of scientific articles to calculate similarities.
6. **Character-Based Plagiarism Detection:** There are two sub-types of character-based plagiarism detection: fingerprint matching and string matching. In the fingerprint method, the preprocessing step includes selecting the plurality of substrings from the paper using n-grams to produce a representative extract of the document. These assimilate are called fingerprints.

Intrinsic Detection

Internal plagiarism detection refers to finding parts of plagiarism without the use of references. It depends on the idea in order to every writer has their own article format. Plagiarism is detected by determining the author's style or writing style in the article. If the reference is not mentioned, it means checking for plagiarism or approving a particular document using the stylometry. This is done by comparing the different properties of a section or part of a document with other sections or parts of the same document. A database of suspect records is not required to detect plagiarism in this way as they check the text against their own text to check for plagiarism [17]. The framework of the intrinsic plagiarism detection system can be seen in Figure 2.3.

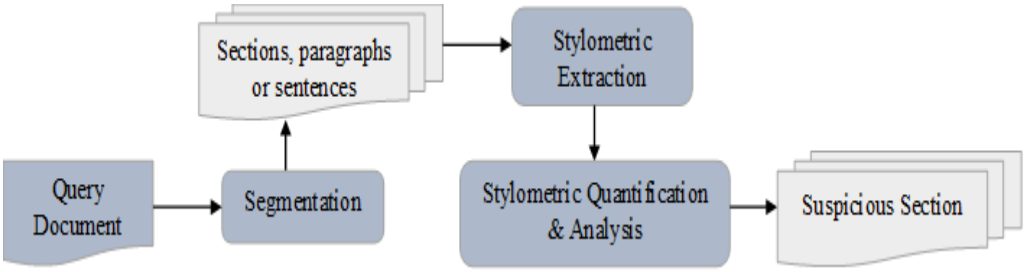

Figure 2.3 Framework of intrinsic plagiarism detection

The recap of the internal structure is maintained: The entry just contains a query document. There are three major processes involved. Initially, a query document is divided into smaller sections like chapters, paragraphs, or sentences. Text segmentation is performed at the word level. Following, the stylometric properties are taken from various parts. Third, stylometric measurements and quantization roles are used to analyze the difference of several formal attributes [16].

There are three general techniques for internal plagiarism detection methods, including [18]:

1. **Bag of Words (BoW):** BoW is used to divide the document into unique terms and count the frequency of each term to form a "baseline" function. The number of words is significant because it provides a basic input to usual class text classification methods.
2. **Latent Semantic Analysis (LSA):** LSA is a method for capturing latent semantic associations based on word usage. This technique is used as an information retrieval technique and more recently for plagiarism detection. It works by deriving a measure of semantic similarity between words in the text and mimicking human word classification and categorical evaluation.
3. **Stylometry:** This is based on the statistical use of computational algorithms to analyze the writing style of a particular author. This technique is used to clarify the differences between the two texts, and it is believed that all authors have a unique writing style that is unknowingly done.

2.3.2. Cross-Lingual

Cross-language plagiarism emerges when the source (or original) text is in one language and the plagiarized text is in another language. In recent years, cross-language plagiarism detection has magnetized the observation of the investigative society because large volumes of digital text are readily accessible in multiple languages across online digital repositories, and machine translation systems are smoothly available to perform cross-language easily and it is more difficult to detect plagiarism [19]. The structure of cross-lingual can be seen in Figure 2.4.

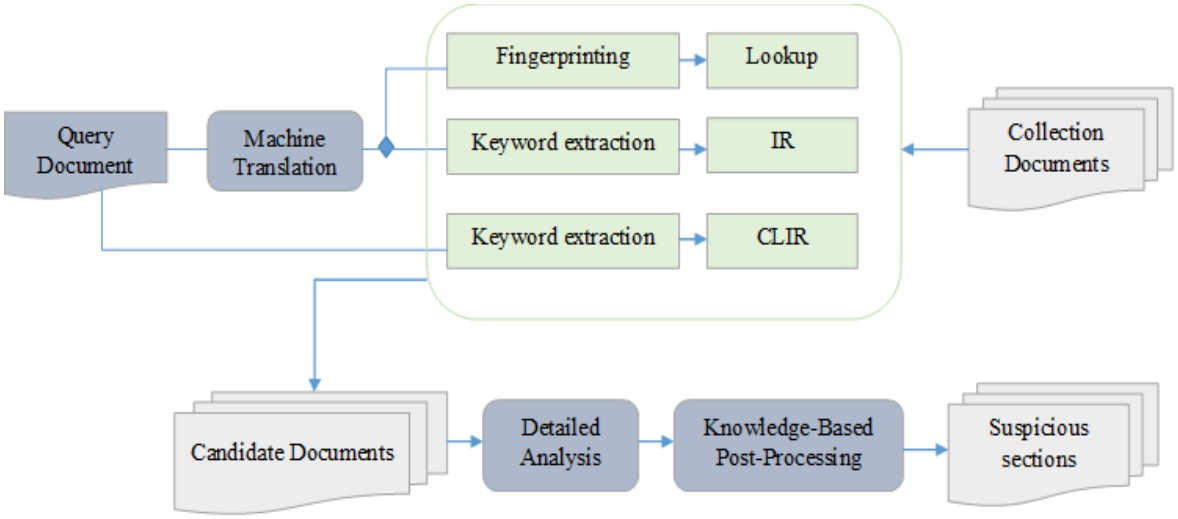

Figure 2.4 Framework of cross-lingual detection

2.4. Plagiarism Detection Techniques

Plagiarism detection techniques specify documents with a high probability of plagiarism due to the presence of original articles. The system analyzes resemblance standards among the writing and if it turns out that the similarity value among the two articles is high, the technique announces the two articles as a suspect [20]. External plagiarism detection techniques can be seen in Figure 2.5.

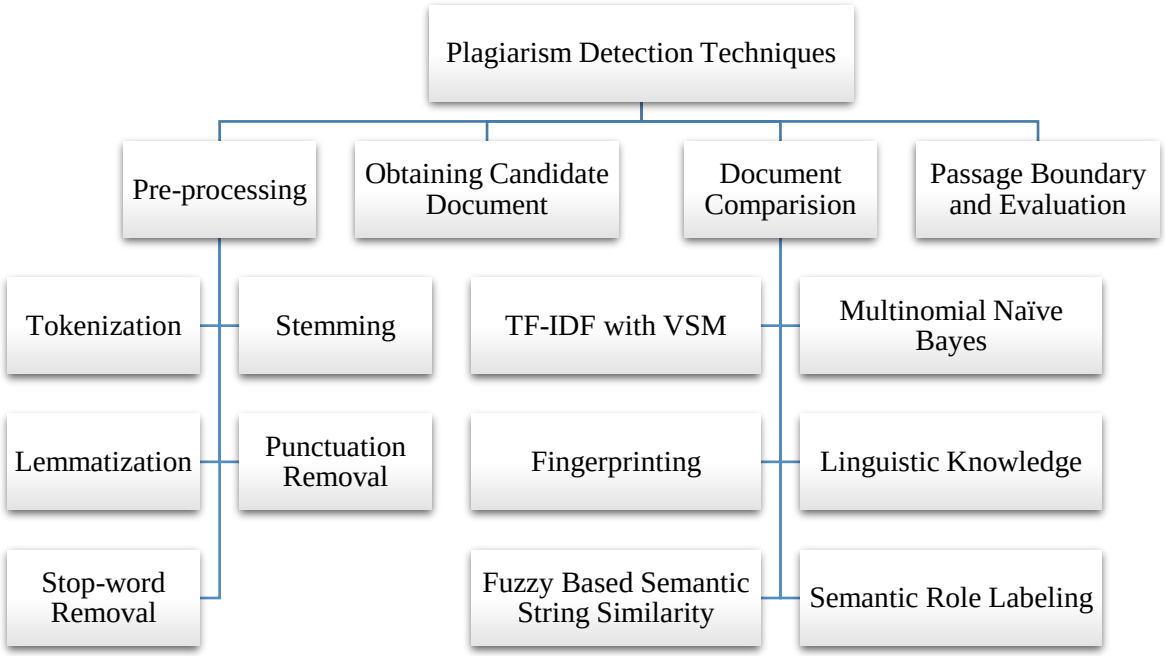

Figure 2.5 Plagiarism detection techniques

2.4.1. Pre-Processing Techniques

Text preprocessing methods strongly supports plagiarism detection operations to increase precision and is extremely powerful in excluding unrelated information from words. Text pre-processing implements the following steps [2, 15]:

1. **Tokenization:** is the process of separating the introductory sentence into single words, tokens, and other related word units.
2. **Punctuation Removal:** Punctuation marks are an effective fragment of the text, however, using them alone makes no sense. In text processing, punctuation does not provide meaning, so during plagiarism detection processing, all punctuation marks are removed from the suspicious articles.
3. **Stop-word Removal:** These are an integral section of every conversation, but they are considered insignificant when compared to documents. Frequent occurrence of stop-words is usually elevated in compilation, so before using plagiarism detection techniques, these stop words should be deleted and the overall size of the document minimized. This process enhances the complete success and efficiency of the plagiarism detection system.
4. **Lemmatization:** Words can have dissimilar shapes due to the addition of suffixes to their roots. These attachments can be eliminated by lemmatization. Thus, various shapes of the same word are decreased to identical word.
5. **Stemming:** Stemming converts terms to origin shape, which generalizes the content for similarity analysis.

2.4.2. Obtaining Candidate Documents

In this step, each suspected text is compared with all original texts. One or more similarity metrics are implemented to assign a similarity outcome to each pair of suspicious-source texts. The probability is found by setting a threshold for the similarity outcome. When a pair achieves a specific threshold, the pair is recorded as an applicant match. Else, the pair is thrown away as not being plagiarized. Jaccard Similarity is an effective measure of similarity with overlapping n-gram words. This technique compares texts in pairs and extracts the word n-gram from the documents and uses the Jaccard coefficient to determine likeness [2].

$$Jaccard\ Similarity = \frac{amount\ of\ overlapping\ n-grams}{union\ of\ n-grams\ in\ both\ texts} \tag{2.1}$$

2.4.3. Document Comparison Techniques

The techniques that can be used to compare documents can be classified as follows:

Term Frequency and Inverse Document Frequency (TF-IDF) with Vector Space Model (VSM)

VSM is a productive representation that is frequently applied to retrieve data or other word processing tasks. For the detection of plagiarism, VSM can be viewed as a global technique for measuring similarity. Clauses extracted from suspicious articles and source documents are considered to be independent groups of words. A weight calculated by TF-IDF is given for each term. The similarity of two clauses can be measured using the cosine similarity as follows [2]:

$$D(I_S, I_R) = \frac{\sum_{k=1}^{n} WS_K * WR_K}{\sqrt{\sum_{k=1}^{n} WS^2{}_K + \sum_{k=1}^{n} WR^2{}_K}} \quad (2.2)$$

I_S and I_R, a pair of clauses taken from the suspect article S and original article R; $WS_K$and $WR_K$are weights of words in S and in order of R.

Semantic Role Labeling (SRL)

SRL is the procedure for defining and labeling discussions in writing. The main concept is that the semantic analysis of the text at the sentence level specifies the items and name of the script. SRL prolonged to the explanation of occurrence like determining of “how”, “where”, “who” did “when” with “whom”, also “what”. A sentence's base upon (usually a verb) determines "what" occurs, and different sections of clause indicate different discussions of the clause (like "when" and "who"). The main duty of characterizing tasks of semantic is to specify what semantic relationship exists between an attribute and related contributors or features, these relationships come from a predefined series of feasible semantic tasks for that attribute or set of attributes. Common terms in SRL are agent, area for the organizations involved in the occurrence, and patient. These designations can be prolonged to include additional particular arguments like duration and location in a text [21].

Multinational Naïve Bayes

The Naïve Bayes classifier is acceptable for pattern identification and can be used to find plagiarism. This classifier is based on Bayes' hypothesis. When S with a few numbers of classes or results depends on several characteristics represented by t1, t2 ... tn [2].

Using Bayes' theorem:

$$\mathrm{P}(\mathrm{S}|t_1\ t_2 \ldots\ t_n) = \frac{P(S).P(t_1 \ldots t_n)|S}{P(t_1,t_n)} \tag{2.3}$$

Using qualified possibility:

$$P(S|t_1, t_2, \ldots\ t_n = P(S).P(t_1, t_2, t_n)|S \tag{2.4}$$

Finger Printing Based Plagiarism Detection

The main substance of a paper is displayed handling fingerprints. Small sections of articles are extracted as a sequence of the whole number named fingerprints. N-gram based techniques are most commonly used for fingerprinting. A sustained set of fixed-size strings shapes an n-gram. There are various techniques to select fingerprints, including [20]:

1. **N-gram:** Each part of the paper is marked as a fingerprint. The benefit of this technique is that it is simple to perform, and the downside is that there is no similarity when text eliminating, adding, or substituting text. For instance, even if a single character is added at the beginning of the document, all the fingerprints move to one location. Therefore, similarity recognition is not performed because the modified document does not have a common fingerprint with the source document.
2. **0 mod p hash:** For inter rate p, these techniques choose fingerprints discovered each 0 mod p. plagiarized content is recognized if the hash of the copied text belongs to the hash chosen by 0 mod p.
3. **Winnowing:** The first Winnowing selection strategy was suggested by Schleimer in 2003. First, the n-gram technique was used to create document fragments. A hash value is then generated for these text parts using a hash function. The hash value is a numerical representation. The next other constant length window is applied to repeat the preceding stage for the created hash value place of the source text. Finally, a minimum hash value is selected in each window. If there are plural hash values within the lowest values, the hash value on the rightmost is selected. For all text documents entered, a series of fingerprints that represent them is obtained by winnowing the outcome. The input for this task is a document that went through several preprocessing steps to extract and normalize the text. The normalized text is then broken down into sentences, each of which performs a sieving step to generate a fingerprint. In order to detect plagiarism, pairwise comparisons are made between each of the feasible integrations of sentence fingerprints.

Fuzzy Based Semantic String Similarity (FUZZ)

FUZZ uses four stages. The first stage is the pre-processing such as: tokenizing the documents, deleting stop words, and handling. The following stage is to itemize an applicant's articles with the assist of the Jaccard coefficient and shingling. The final stage is examining every doubtful article and thoroughly examining the applicant's articles. Resemblance among clauses pair is calculated using the term to-phrase element finding mutuality in order to every term (T_q) in shady phrase $P_q$and the phrase P_c of the candidate [22]. SVD of sentence-by-document similarity can be seen from Figure 2.6.

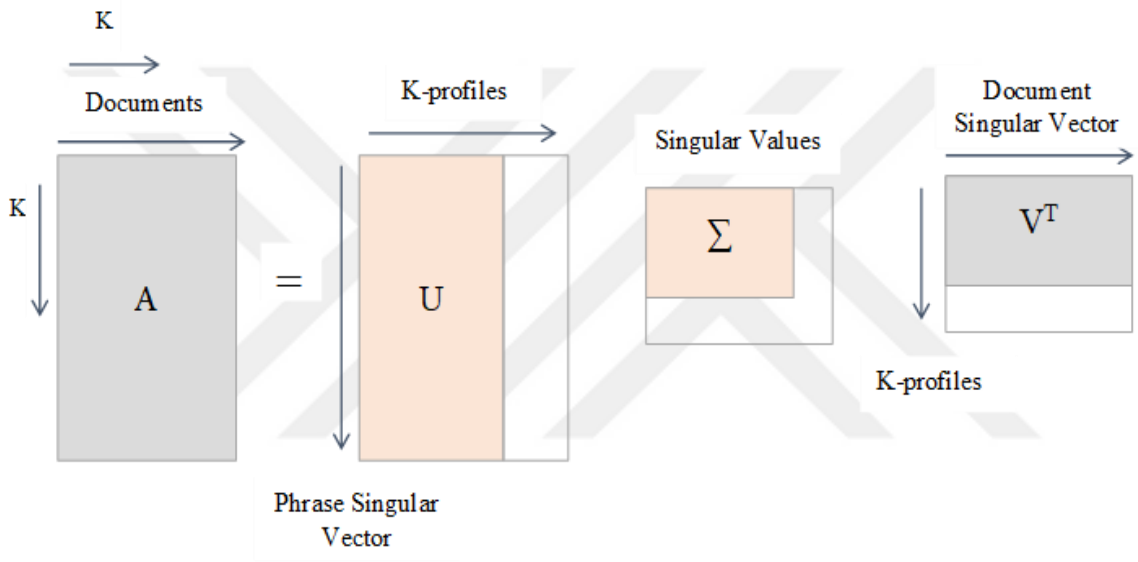

Figure 2.6 SVD of clause-by-article mold

Following formula:

$$\sigma_{q,c} = 1 - \prod t_k \, P_c(1 - £_{q,k}) \tag{2.5}$$

where, $t_k$are the words in $p_c$and $£_{q,k}$is the fuzzy identity between words $t_k$and $t_q$that is computed as under:

$$£_{q,k} = \begin{cases} 1 & while\ t_k\ and\ t_q\ are\ same \\ 0.5 & if\ t_k\ is\ in\ the\ synonym\ of\ t_q \\ 0 & elsewhere \end{cases} \tag{2.6}$$

Similarity outcome among (p_c,p_q) is a fuzzy value sequence I within 0 and 1 calculated applying the following formula:

$$Sim(P_q, P_c) = (\sigma_{1,c} + \sigma_{2,c} + \cdots + \sigma_{q,c} + \cdots + \sigma_{n,c}/z \quad (2.7)$$

where, z is the entire count of words in p_q.

Plagiarism Detection using Linguistic Knowledge (PDLK)

With the PDLK technique, the copying of someone else's work is determined by semantic and syntactic criteria. After implementing the preprocessing, it gives a pair of the suspect and original clauses as an entry for elaborated contradiction. After the term row has been created, a semantic vector and a syntax vector for both sentences are created and determined using the word set. Steps for similarity computation PDLK are shown in Figure 2.7. These values are accessed in the linear function below for the total identity outcome [22].

$$Sim_{sentences}(S_q, S_x) = \alpha\, Sim_{semantic}(S_q, S_x) + (1 - \alpha).Sim_{syn}\,(S_q, S_x) \quad (2.8)$$

where 0< α <1 is a weighting parameter that determines the impact of semantic and syntactic dimensions on the similarity outcome. As the value of α increases, the semantic measure also increases.

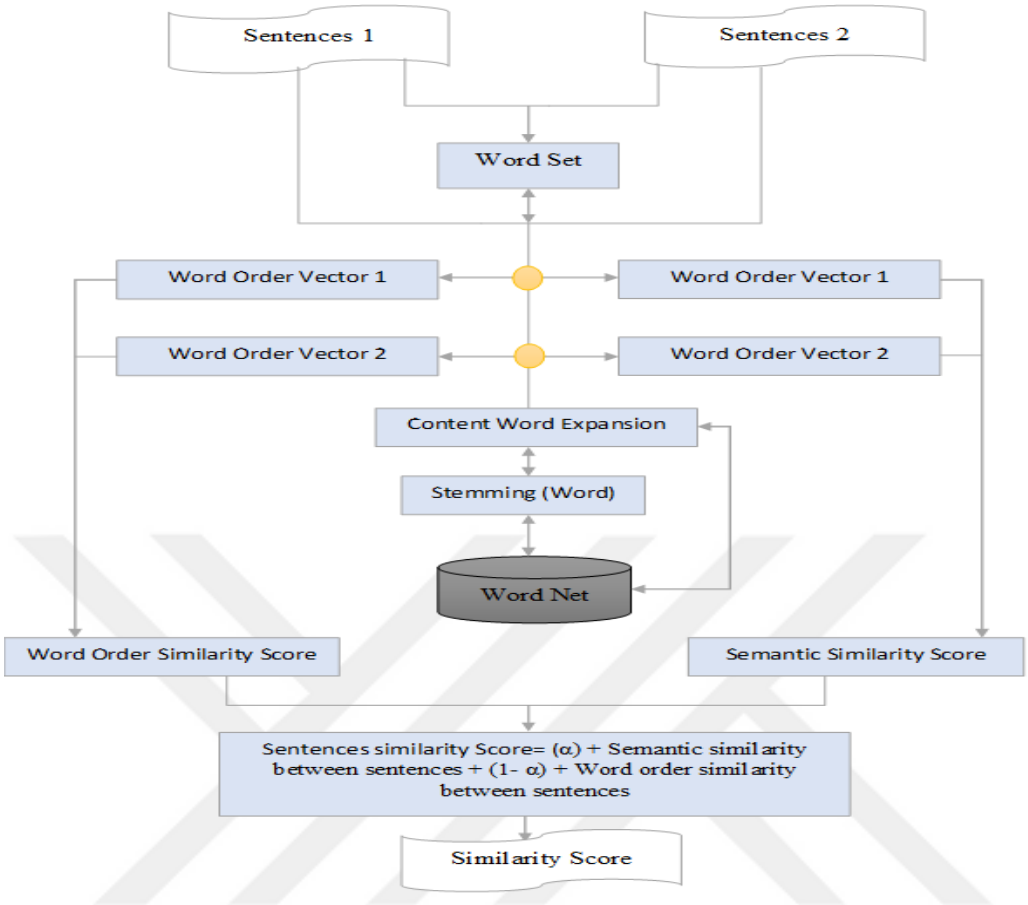

Figure 2.7 Steps for similarity computation PDLK

2.4.4. Passage Boundary Detection and Evaluation (PBDE)

When a suspicious and a source paper are issued, matches among the two papers are determined using some basic heuristics. Heuristic seeds are usually extracted by a measure of similarity among seeds. By creating as many logical heuristic seeds as possible, the next step of converting them into aligned text passages becomes much easier. Seeds can be phrases, terms, or n-grams of characters. After seeding, the initial matches are merged into aligned text passages of the highest length among the two papers which are then marked as plagiarism detections. The reason for merging seed matches is to find out if a paper includes plagiarized passages at all instead of just coincidentally matching seeds and to identify a plagiarized passage as a whole rather than just its parts [2].

Given a sequence of aligned passages, the passage filter eliminates all aligned passages that do not meet specific criteria. The reason is mainly to handle duplicate passages and discard exceedingly brief passages [23]. Eventually, the assessment is performed using the four standard measures Recall (Rec), Precision (Prec), Granularity (large), and F Measure.

2.5. String Matching Algorithms

This is an algorithm based on a mathematical process that demands one value or sequence values like entrance as well as generates one value or sequence values like outputs. Consequently, an algorithm is defined as a list of mathematical stages which convert inputs into outputs. The algorithm defines a particular computation method to achieve this input/output relationship informally [24].

2.5.1. Exact String Matching Algorithms

Exact string matching algorithms are concerned with finding all parts of the P pattern in the T text. An exact match compares the characters found in a pattern window with a text showcase. The length of the two windows must be the same during the comparison stage. Moving letters during inconsistency are required to advance effective algorithms [25].

2.5.2. Approximate String Matching Algorithms

The near-duplicate string matching algorithms determines substrings that are near to a particular pattern string. Near duplicate algorithm is in contrast to the exact match algorithm, which waits for a perfect match. In this situation, it is difficult to determine the degree of intimacy, but what is interesting depends on the implementation and difficulty of the problem. This approach contains detecting all substrings S with K or smaller variances with stated text t such that d (p, S $\geq$ K), where p indicates a short pattern string with length m, d indicates distance function, and K indicates an integer with value K $\geq$ 0. That is, the algorithm counts the number of matches and modifies some thresholds as K when matching substrings to strings. This approach is often used when there is a K mismatch between the pattern and the text provided. Approximate string matching algorithms uses two common distance measures: The Hamming distance measure is the number of positions with mismatching characters between two rows of identical distance and the Levenshtein distance function is the lowest number of letter adding, removing, and replacing that needed transforming of one string to another. These algorithms are used to deal with mistyped words, poor quality text, and the difficulty of searching for foreign names. Near identical algorithms can be classified as [25]:

a) **Filtration-based algorithms:** These algorithms include two steps as follows: the initial step is to determine the possible locations of pattern formation in the text. In the following step, all these locations are totally verified. Newly developed algorithms which conform to this process also include these steps. On the other hand, these algorithms are not effective at worst. Almost all filtration indices generally vary in words of text sampling, pattern sampling, and placement criteria.

b) Trackback-based algorithms: This technique is often prolongations of the exact string matching algorithms. This method modifies the identical matching pattern to allow near-identical match applying edit distance techniques. For such approaches, the use of concise (compressed data) and suffix-based data structures is recommended. In most cases, near duplicate algorithm is not working properly in the medical field. It expects about 100% contest to determine an answer besides guesswork. In this case, exact string matching has an additional benefit than near-identical matching.

2.6. Similarity Measure

While similarity expresses resemblance, similarity in similar appearance, form, personality, or quantity, besides being identical. Classification of similarity measures can be seen from Figure 2.8. It includes two kinds of similarities: String-based Similarity Metric and Language-based Similarity Metric [5]:

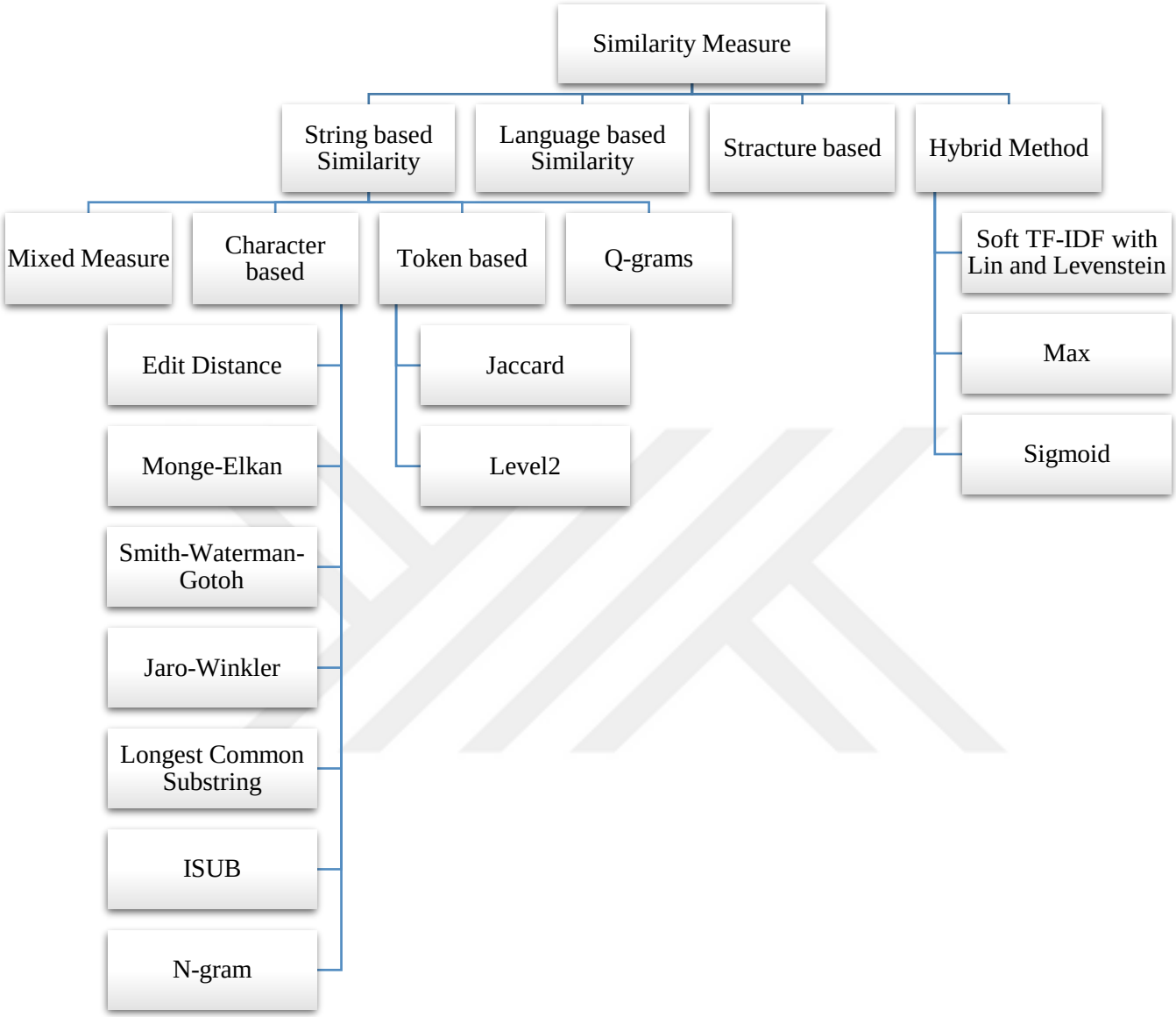

Figure 2.8 Types of similarity measure

2.6.1. String Based Similarity

The similarity metric of the string computes the syntactic resemblance of two names and can be separated into two sections. Character-based criteria measure identity solely on the basis of the shape and order of the characters. The token-based measure of the similarity of strings is to first identify two strings as a series of tokens (words) and then calculate the similarity among the two sets. Algorithms based on Edit Distance [26]:

Character-Based

1. **Edit distance:** is a major category of string metrics that finds the distance among two strings S and T by computing the expense of the optimum series of machining processes that transform S to T. Ordinary editing operations are the adding, removing, and replacing of characters and the transshipment of characters. There are different ways of computing the matching distance depending on which operations are permitted [26].
2. **Monge-Elkan:** is an alteration of Smith-Waterman where the match matrix gives 0 to all inputs, besides the exact match is 5, and the approximate match is 3. If they fall into one of the following sets, the two characters would nearly match: {d t} {g j} {l r} {m n} {b p v} {a e i o u} {, .}. An affine gap model with gap initial penalties of -5 and intermittent continuous penalties of -1 is used. Monge-Elkan is mainly appropriate for shortened forms of name variations [26].
3. **Smith-Waterman-Gotoh:** advances Smith-Waterman scaling by adding supposed affine space costs. Two insertion costs are presented: open space, a penalty corresponding to the starting of the incompatible string, and widening of the gap, a penalty for continuing. Moreover, the conversion amid identity letters (e.g. {d, t}, {g, j}) is dedicated a dissimilar weight from the match/ mismatch weights. For instance, five units are given for matching characters, three units for characters that are similar-sounding, and -3 units for mismatched characters [7].
4. **Jaro-Winkler:** dedicated pair sequences S1 = a1 ... a N and S2 = b1 ... bL , if there is a letter bj = ai in S2, describe the letter ai in S1 to be ordinary, so that | i - j | <min. (| S1 |, | S2 |) / 2. Let S‘1 = a‘1 ... a‘m, be the ordinary letters in S1 (in the same sequence) and S‘2 = b‘1 ... b’m be the ordinary letters in S2, describe the transposition S‘1 and S‘2 to be a spot i so that ai ≠ bi. Let m be the quantity of the ordinary letters and t be half the number of transpositions [26].
5. **Longest common substring (LCS):** is applied in implementations like clinical patient record matching and text synthesis, but not for title comparisons. It computes the longest substring found together in two strings. The outcome is standardized by separating by the letter extent of the longest string [7].
6. **ISUB:** is known as a new metric designed particularly for ontology classification, and it claims that the similarity is based on commonality but the dissimilarity amid two sequences being set against [26].
7. **N-gram:** N-gram first transforms every string into an n-gram set. For instance, if the array is "paper" and n = 3, the outcomes would be pap, ape, per. A progress is to add particular characters before the beginning and after the end of the array, from which you will get {## p, #pa, pap, ape, per, er%, r %%}[26].

Q - Grams

Q-Grams calculate the variety of text of length q that are usual to two strings. Intuitively, the order of the letters is more significant than just the letters. Various measures have been developed with varied values of q and various normalizations. The Jaccard and Dice coefficients calculate the value of general bigrams (2 grams). Jaccard separates the summation by the complete amount of individual bigrams in both strings, and Dice separates it by the complete amount of bigrams in both strings. Character-based and q gram-based similarity metrics operate effectively for typographic faults. On the other hand, they cannot detect the resemblance of strings while the sequence is replaced (e.g. Manta Café vs. Café Manta). Token-based metrics attempt to satisfy the issue [7].

Token Based

Token-based measures transform the strings to tokens and dispose of the arrangement in which the tokens be seen in the two strings [7].

1. **Jaccard Measure:** is explained as the junction of two sequences of terms shared according to their combination. Let A and B be phrases s1 and s2 separately, then [26]:

$$\text{Jaccard}\,(S_1, S_2) = \frac{|A \cap B}{|A \cup B|} \tag{2.9}$$

2. **Level2:** In addition, Monge and Elkan suggested a sentence-based similarity metric in their work. The Second String library performs this metric kind of Level2. That is why it is named Level2. It specifies sim0 as the default similarity metric (e.g. Levenstein, Jaro-Winkler etc.) and {a1, a2 ... an}, {b1, b2 ... bm} the set of terms in s1 and s2, separately. Then the similarity formula is [26]:

$$sim\,(s_1, s_2) = \frac{1}{|S_1|} \sum_{i=1}^{m} max_{j=1}^{n} \{sim'(a_i, b_j)\} \tag{2.10}$$

Mixed Measure

The basis of mixed metrics is to implement a character-based metric (secondary metric) to entire token pairs among the two strings and simply examine tokens that meet a specific criterion (for example, a threshold value) as input for a token-based metric. Monge-Elkan performs the mean score of the top matching tokens in the secondary metric like Levenshtein, Jaro, and Smith-Waterman. Soft-TFIDF integrates TF-IDF and Jaro-Winkler measurements. Primarily Jaro-Winkler (sim') is implemented to each of the token pairs among two strings, and the TF-IDF measure is implemented to tokens whose identity value according to Jaro-Winkler is greater than or equal to the threshold ($\theta \geq 0.9$) [7].

2.6.2. Language-Based Similarity Metric

The words may be different but have similar meanings, for example, "car" and "auto". WordNet can analyze this issue. Wu and Palmer's similarity is used to make this scenario work. If the strings contain a set of terms, the maximum degree of similarity of every possible word set pair is used. However, WordNet's weakness is that it consists only of a subset of complete terms in the language. Using vector representations of terms in Word2vec templates alleviates this issue. The cosine similarity among the two vector representations of the terms is computed. If the strings contain more than one term, the Phrase2vec algorithm is performed [28].

2.6.3. Hybrid Methods

The version of hybrid methods includes follows [26]:

- **Soft TF-IDF with Lin and Levenstein (SLL):** Unlike the soft TF-IDF measurement, while computing identity among tokens, at the beginning it uses the Lin measure. If the value returned by the Lin measure is less than a predefined threshold (for example, 0.8), then it applies the Levenstein measure.
- **Max:** This technique uses the supreme similarity value for two characteristics calculated from several measures, such as aggregated similarity.
- **Sigmoid:** This technique applies a sigmoid operation to highlight elevated particular similarity results as well as a nominal particular similarity result. Total identity is the mean of particular sigmoid values of several identities.

2.6.4. Structure-Based

A structure-based similarity metric is also implemented: measurements of string similarity and linguistic similarity among feature parents and feature paths are fully listed. These similarity measures involve the assumption that the corresponding characters have similar parents and a similar position in the hierarchy [28].

3. SYSTEM DESIGN

All detection systems' goal is to find plagiarized questionable phrases and their matching equivalent in existing original documents. Suspicious entry items are compared with available sources to determine if they have been copied or tampered with. The origin database or corpus can be the whole web, a particular library, or a specific-domain database. Due to the existence of the database for collation, it behaves such as an article collation operation applying a similarity program [29]. This section of the thesis classifies the plagiarism detection techniques and organizes their explanations in accordance with our plagiarism typology. The overall proposed system is designed for both Natural Language Plagiarism (NLP) and Source Code detection.

3.1. The Purpose of the System

One of the challenges over the past decades has been copying someone else's work without giving them permission, which makes an immense issue for those who upload their work over the internet. In addition to plagiarizing everyday language, uploading plenty of source code for various programming languages over the Internet increases plagiarizing of coding. However, various algorithms were developed to detect the plagiarized document through a plagiarism detection mechanism. Meanwhile, plagiarism removal approaches appeared to decrease the plagiarism rate. However, these types of attempts are still attracting so many individuals, and algorithms are not powerful enough to detect various plagiarized fields such as everyday languages, source code, and disguised words. Therefore, previous techniques can be no longer used in today's developed technology, since now plagiarism techniques become a real challenge, especially in the academic area.

To prevent copying someone else's work from the network and boost plagiarism rate, this study proposes a powerful hybrid approach of approximate string matching algorithms and term frequency that can detect exact and disguised textual in everyday language and source code. In order to achieve our goals, we followed the following steps: initially, suspicious document compared with the source documents, after determining the minimum edit distance, the text considered as plagiarized if the founded threshold between the words is greater or equal to 0.50 which we used as the key value to detect plagiarism between suspect and source documents. Later, if the previous step is satisfied, TF-IDF counts the word as its original, otherwise considered as not plagiarized. Eventually, cosine similarity determines the similarity measures between the vectors.

3.2. The Proposed Methods

Two techniques are combined as a hybrid method is proposed: Levenshtein Edit Distance (LED), an approximate string matching algorithm, and Term Frequency Inverse Document Frequency TF-IDF. This is followed by finding Vector Space Model (VSM). Finally, cosine similarity is calculated to determine the percentage of similarity between documents.

3.2.1. Edit Distance and TF-IDF Based Algorithms

Levenshtein Edit Distance

One of the effective methods of comparing strings, two terms, or mainly two sequences is the edit distance which calculates the cost of the optimal sequence of the editing process by adding, removing, and replacing. There are multiple differences in the computation of the edit distance, while arguably the Levenshtein distance proposed in 1966 is the most powerful. Distance is the minimum number of changing processes to convert one string to another. Allowed editing operations are adding, removing, and substituting individual characters [5].

A threshold between suspicious papers and the source document repository is established. If the predefined threshold is greater than or equal to 0.50, the word is considered similar to the original. Later, TF-IDF, which is explained below, is used to determine the frequency of the words to have a better understanding of the plagiarism rate.

TF-IDF Weighting

TF-IDF is performed as an important determinant in text mining and information retrieval, which enables the establishment of a vector space where every vector indicates how a word is significant for a paper in an aggregation through the compound of Term Frequency TF (t, d) and Inverse Document Frequency IDF (w) [30]:

- Term Frequency TF: It refers to the number of times a word is included in a document. Since the length of each document is various, a term may appear much more in long documents than in short documents. TF is defined as:

$$TF_{(t,d)} = \frac{O}{t} \tag{3.1}$$

where O refers to the number of times that a term t occurs in a document, and t is the number of words in the paper.

- Inverse Document Frequency IDF is a statistical weight performed to measure the significance of a word in a collection of text documents. It also has a built-in IDF

functionality that decreases the weight of terms that appear regularly in the document set and increases the weight of words that appear infrequently. IDF is defined as:

$$IDF_{(t,d)Iog} = \frac{|D|}{N} \tag{3.2}$$

Where | D | is the complete number of documents and N is the number of documents with the term t.

- Term Frequency-Inverse Document Frequency TF-IDF is computed for every word in the article by combining TF and IDF.

$$TF - IDF(t, d, f) = TF(t, d) * IDF(t, d) \tag{3.3}$$

TF-IDF algorithm is performed to detect the relevance to a query paper and TF-IDF weighting for representing the percentage of terms in a paper. Terms with high TFIDF weighting indicate a powerful connection to the document, including the paper query.

3.2.2. Vector Space Model with Cosine Similarity Measure

The vector space model (VSM) is the common method that uses the lexical and syntactic characteristics and describes the article in vector space. Several weighting schemes are then used to do comparison. Term-frequency-inverse document frequency (TF- IDF) and term frequency-inverse sentence frequency (TF-ISF) are mainly used as two weighting schemes [29].

The TF-IDF weighting technique is frequently used in the vector space model along with similarity methods to find the similarity among two articles. Common measurements based on the vector space model are cosine similarity and Jaccard coefficient, performed to represent text papers (usually every object) as vectors in multidimensional space. In this study, a cosine similarity measure is applied.

The cosine similarity is a measure of the similarity that is computed by multiplying the cosine angles of the two vectors to be checked against. The cosine 0 ° is 1, which is less than 1 to the value at any other angle. Thus, the similarity values of the two vectors are: If the cosine similarity value is 1, they are similar. This technique is a conventional technique that is frequently applied and assembled with the TF-IDF. The calculation of cosine similarity is implemented according to the below equation [31]:

$$Cos\alpha = \frac{A*B}{|A|*|B|} = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt[1]{\sum_{i=1}^{n} (A_i)^2} * \sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{3.4}$$

where A is the weight of every attribute of vectors A and B is the weight of every attribute in vector B. Subsequently generating the vectors and converting the attributes in the vectors to weighted values, we can use cosine similarity measure to compute the likeness of those vectors. Basics of cosine similarity, the bigger angle shaped among two coordinate vector comparison papers, a lower degree of similarity of papers. Moreover, the smaller degree of cosine similarity level means the similarity rate will be bigger [31].

3.3. System Architecture

The recommended system implements the next stages shown in Figure 3.1.

1. Levenshtein Edit Distance Algorithm is implemented to find near-identical information between the source and suspicious document. After the simulation test, the optimum threshold is found to be 0.5. If the determined threshold is greater or equal to 0.50 then the word is considered as the same (plagiarized). Later, TF-IDF counts the word as it is original, which increases the rate of detecting plagiarism. Otherwise, the word is considered not plagiarized.
2. Cosine similarity is used as an efficient way to determine the similarity measures.

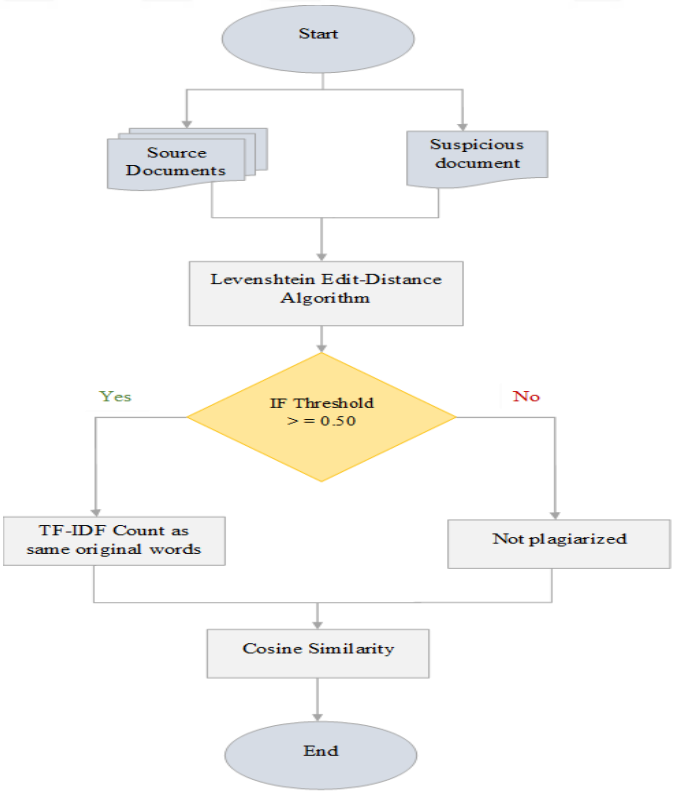

Figure 3.1 Proposed system architecture for plagiarism detection system

4. MATERIAL AND METHOD

In this thesis, three different datasets have been used to improve the applicability and success of the developed hybrid algorithm. Also, two similarity measure is implemented. Initially, if the similarity measure reaches the required value, then TF-IDF counts the word as the same as the original. Eventually, cosine similarity measures the overall plagiarism rate of a document.

4.1. Dataset

A common technique for assessing a plagiarism detection system is testing on a dataset. This usually requires access to a number of plagiarized and non-plagiarized texts, and the system has to figure out which class a specific case belongs to. In preliminary works, the dataset applied in investigations was not made to measure the intention of finding plagiarism. Therefore, they have not completely reflected the varieties of plagiarism that are available in an actual situation. [1].

This research also contributes to providing a dataset for natural English language and source code plagiarism detection. The dataset includes a variety of plagiarism occurrences consisting of basic copy/paste, word shuffling, phrase shuffling, removed characters, added diacritics, and paraphrasing. The corpus is expected to perform plagiarism detection approaches of available source documents comparable to each other. Plagiarism text files specific to this dataset are created for plagiarism purposes.

Three different corpora are applied for testing:

1. Automated machine paragraphs, the size of the dataset is 60.8MB which contains 79,970 documents, and it is separated into two sets: first original articles (39,241) and suspicious document (40,729) [32].
2. Misspelled words contain more than 6,000 misspelled words [33].
3. Java source codes depend on Github GH and stack overflows SO. It contains 180 java files, 97 of them are Github GH source code and 83 of them are on stack overflow SO [34].

4.2. Methods

We propose an improvement to an implementation of the plagiarism detection algorithms to increase the score of similarity. The algorithms are implemented in this study are including as follows:

- Levenshtein edit distance, an approximate string matching algorithm.
- Term frequency-inverse document frequency TF–IDF algorithm.

Also, Cosine similarity has been implemented to determine the similarity between two files. Java Programming Language is used to run the codes using Eclipse.

4.3. Similarity Measures

The advanced plagiarism detection system is based on two similarity metrics. Accordingly, a threshold is usually predefined in order to determine the cases of plagiarism. It helps to identify near-identical plagiarized information. If the threshold is greater or equal to 0.50, the system decides that there is plagiarism and TF-IDF counts as it is textual information identical to its original, Later, the Cosine similarity measure is used to find the distance between the original and suspect document. Moreover, if the threshold is smaller than 0.50, plagiarism will not be found.

4.4. Tools and Requirements

Java belongs to a family of languages strongly influenced by C ++. It is safe, portable, maintainable, and has better high-level competitive tools than any other language. In addition, because its own virtual machine can be run on different types of computers without any problems, a robust and easy to write and run Java programming language has been implemented in this project. Eclipse is used as a tool to execute the codes.

5. RESULTS AND DISCUSSION

Initially in this study, Levenshtein edit distance is implemented to determine near-identical similarity measure. Following, TF-IDF is used to only detect the term frequency of usage of the words in the documents. With the help of cosine similarity, the overall plagiarism rate between the documents can be revealed. Therefore, the hybrid algorithm of combining edit distance and TF-IDF is implemented with cosine similarity to find out a similarity rate between files. The outcomes demonstrate that the hybrid algorithm is more powerful than the original TF-IDF since it can detect disguised or mistyped words.

5.1. Similarity Measure Based on Edit Distance Algorithms

Levenshtein edit distance has been used to determine the similarity between suspicious articles and source documents. However, in the case of having any mismatched character, the rate of plagiarism will be decreased even if the word has the same meaning. For example, if the character s added to the end of the word “include” becomes “includes” , the rate of plagiarism will be changed because of adding mismatched character to the plagiarized word, the similarity measure will be decreased even if the determined words are identical to its original. Given Table 5.1 for machine paragraph dataset, Table 5.2 for the mistyped dataset and Table 5.3 for the source code dataset shows the percentage of plagiarism between source and suspicious documents by using Levenshtein edit distance algorithm with the using three different datasets of both everyday language and source codes to decide whether the documents are plagiarized or not.

Table 5.1 Plagiarism rate based on edit distance algorithm for machine paragraph dataset

Original Document	Suspicious Document	Edit Distance
ORIG-4	**SPUN-4**	%53
ORIG-6.txt	**SPUN-6.txt**	%41
ORIG-10.txt	**SPUN-10.txt**	%66
ORIG-14.txt	**SPUN-14.txt**	%59

Table 5.2 Plagiarism rate based on edit distance algorithm for mistyped dataset

Original Document	Suspicious Document	Edit Distance
Orig-1.txt	**Sus-1.txt**	%78
Orig-2.txt	**Sus-2.txt**	%86

Orig-3.txt	**Sus-3.txt**	%82
Orig-4.txt	**Sus-4.txt**	%76

Table 5.3 Plagiarism rate based on edit distance algorithm for source code dataset

Original Document	Suspicious Document	Edit Distance
9_so.java	**9_gh.java**	%28
39_so.java	**39_gh.java**	%34
54_so.java	**54_gh.java**	%23
57_so.java	**57_gh.java**	%44

Given Table 5.1 for machine paragraph corpus, Table 5.2 for mistyped corpus and Table 5.3 for source code corpus shows the similarity measures of suspect document compared with original document by using edit distance algorithm. Experimental outcomes declare that any mismatched or mistyped character affects the results and plagiarism rate is decreased.

5.2. Similarity Measure Based on TF-IDF Algorithms

TF-IDF is one of the most powerful algorithms used for detecting the same words. However, the downside of TF-IDF is that it can only find the exact match, even with any small changes such as adding one character to the plagiarized word, it is considered as not plagiarized and the plagiarism rate will be zero for that specified word. Given Table 5.4 for the machine paragraph dataset, Table 5.5 for the mistyped dataset and Table 5.6 illustrates the similarity outcomes using TF-IDF algorithm on the same datasets shown above.

Table 5.4 Plagiarism rate based on TF-IDF algorithm for machine paragraph dataset

Original Document	Suspicious Document	TF-IDF
ORIG-4	**SPUN-4**	%24.4
ORIG-6.txt	**SPUN-6.txt**	%39
ORIG-10.txt	**SPUN-10.txt**	%22.6
ORIG-14.txt	**SPUN-14.txt**	%43.6

Table 5.5 Plagiarism rate based on TF-IDF algorithm for source code dataset

Original Document	Suspicious Document	TF-IDF

9_so.java	**9_gh.java**	%30.8
39_so.java	**39_gh.java**	%35
54_so.java	**54_gh.java**	%40.2
57_so.java	**57_gh.java**	%42.2

Table 5.6 Plagiarism rate based on TF-IDF algorithm for mistyped dataset

Original Document	Suspicious Document	TF-IDF
Orig-1.txt	**Sus-1.txt**	%5.7
Orig-2.txt	**Sus-2.txt**	%11.7
Orig-3.txt	**Sus-3.txt**	%20
Orig-4.txt	**Sus-4.txt**	%19.1

Given Table 5.4 for machine paragraph corpus, Table 5.5 for source code corpus and Table 5.6 for mistyped corpus shows the similarity measures of suspect document compared with original document using TF-IDF algorithm. Experimental outcomes reflect that the plagiarism rate with edit distance based approximate string matching algorithm is higher than the TF-IDF algorithm, since the TF-IDF can only detect exact matches and returns zero plagiarism rate in case of having any mismatched characters.

5.3. Similarity Measure Based on the Hybrid Algorithms

Once the combination of edit distance and TF-IDF is implemented, the outcome illustrates the success and effectiveness of the hybrid algorithm. In this project, the edit distance measure is used as a consideration, not as a result. If the rate of plagiarized words with edit distance based algorithm reached the required threshold, then the implemented TF-IDF algorithm counts the disguised words the same as the original, which will increase the percentile of accepting those two documents as plagiarized. For the same samples, Table 5.7 for the machine paragraph dataset, Table 5.8 for the mistyped dataset and Table 5.9 for the source code dataset demonstrates the similarity outcomes of the hybrid algorithm among suspicious and source documents.

Table 5.7 Plagiarism rate based on the hybrid algorithm for machine paragraph dataset

Original Document	Suspicious Document	Hybrid Algorithm
ORIG-4	**SPUN-4**	%60.9
ORIG-6.txt	**SPUN-6.txt**	%65.4
ORIG-10.txt	**SPUN-10.txt**	%69.4

ORIG-14.txt	**SPUN-14.txt**	%71.4

Table 5.8 Plagiarism rate based on hybrid algorithm for source code dataset

Original Document	Suspicious Document	Hybrid Algorithm
9_so.java	**9_gh.java**	%81.8
39_so.java	**39_gh.java**	%50
54_so.java	**54_gh.java**	%45.1
57_so.java	**57_gh.java**	%45

Table 5.9 Plagiarism rate based on hybrid algorithm for mistyped dataset

Original Document	Suspicious Document	Hybrid Algorithm
Orig-1.txt	**Sus-1.txt**	%100
Orig-2.txt	**Sus-2.txt**	%100
Orig-3.txt	**Sus-3.txt**	%100
Orig-4.txt	**Sus-4.txt**	%100

Experimental outcomes specify that the hybrid approach consisting of approximate string matching algorithm and term frequency is more powerful and achieves the highest plagiarism rate detection compared to TF- IDF or edit distance techniques alone. Table 5.9 shows the %100 plagiarism rate because the hybrid algorithm is applicable to detect grammatically changed or mistyped words and effectively boosts the plagiarism rate.

5.4. Results

The results show that the hybrid algorithm can successfully and effectively boost the rate of plagiarism detection. Table 5.10 demonstrates the result of combining approximate string matching algorithms and term frequency for the cumulative of the above datasets of machine paragraphs, mistyped words, and source code. The values indicate the percentage of plagiarized documents for Edit Distance, TF-IDF, and The Hybrid Algorithm.

Table 5.10 Comparison of plagiarism rate for algorithms

Algorithms	Machine Paragraph Plagiarism Rate	Mistyped Words Plagiarism Rate	Source Code Plagiarism Rate
Edit Distance	55%	82%	32%
TF-IDF	32.40%	14.13%	37.05%
The Hybrid Algorithm	66.78%	100%	55%

Table 5.10 shows the cumulative of the above three datasets with the comparison of algorithms. The results reflect that the combination of string matching algorithms and term frequency can successfully enhance the plagiarism rate.

5.5. Evaluation Measures

In order to evaluate the success of the result, three evaluation criteria are implemented in this study: precision and recall are two values that cooperatively are applied to assess the efficiency of performance of information retrieval systems. The F1 score is also a useful indicator for comparing two classifiers. F1 score produced by determining the harmonic mean of precision and recall. The results show that the hybrid algorithm reached a score of 0.741 for precision, 1 for recall, and 0.851 for F1, while edit distance had a score of 0.564 for precision, 1 for recall and 0.721 for F1 and TF-IDF pointed out a 0.279 precision score, 1 recall score, and 0.436 F1 score.

6. CONCLUSIONS

Edit distance provides an indication of similarity that may be too close in some cases. If a user duplicates java code and performs several alternates, such as change of variable names, the addition of two comments, the edit distance between the source and copy will be close. On the other hand, TF-IDF is the most commonly used weighting scheme for keywords to simplify all related articles. However, the current TF-IDF technique does not take into account the semantic correlations between terms, which can lead to a less relevant search for documents. Therefore, this study combined both the Levenshtein edit distance and the term frequency inverse document frequency TF-IDF, thereby increasing the score of similarity measured using cosine similarity. The robust hybrid algorithm is also able to detect plagiarism from both the natural language and source codes. Different forms of plagiarism can be detected such as (reorganization of words and inter-textual similarity of insertion/deletion). It can detect different types of plagiarism, such as added comments in Java source code or changes in the data fields and methods, etc. In this investigation three different corpora are used for testing: automated machine paragraphs, mistyped words, and java source codes. From the results, we find that the proposed algorithms are capable of detecting disguised and exact copies of articles which boosts the rate of plagiarism detection.

In the future, we plan to extend the hybrid algorithm to be able to detect paraphrased text. We also plan to expand the scope of the source dataset with advanced programming topics (e.g. finding and sorting) and code files from other programming courses (e.g. object-oriented programming or algorithms and data structures).

REFERENCES

[1] Chong, M.Y.M. (2013). A study on plagiarism detection and plagiarism direction identification using natural language processing techniques, University of Wolverhampton, England.

[2] Lamba, H. and Govilkar, S. (2017). A Survey on Plagiarism Detection Techniques for Indian Regional Languages, Int. J. Comput. Appl, C Vol.

[3] Rani, S. and Singh, J. (2017). Enhancing levenshtein's edit distance algorithm for evaluating document similarity. in International Conference on Computing, Analytics and Networks. Springer, Singapore.

[4] Cherroun, H. and Alshehri, A. (2018). Disguised plagiarism detection in Arabic text documents. in 2018 2nd International Conference on Natural Language and Speech Processing (ICNLSP). IEEE, pp. 1-6. doi: 10.1109/ICNLSP.2018.8374395.

[5] Zouhir, A., El Ayachi, R., and Biniz, M. (2021). A comparative Plagiarism Detection System methods between sentences. in Journal of Physics: Conference Series. IOP Publishing, Morocco.

[6] Naik, R.R., Landge, M.B., and Mahender, C.N. (2015). A review on plagiarism detection tools, International Journal of Computer Applications, C Vol. 125 (11), pp.

[7] Gali, N., Mariescu-Istodor, R., and Fränti, P. (2016). Similarity measures for title matching. in 2016 23rd International Conference on Pattern Recognition (ICPR). IEEE, pp. 1548-1553, doi: 10.1109/ICPR.2016.7899857.

[8] Qinqin, L. and Chunhai, Z. (2017). Research on Algorithm of Program Code Similarity Detection. in 2017 International Conference on Computer Systems, Electronics and Control (ICCSEC). IEEE, pp. 1289-1292, doi: 10.1109/ICCSEC.2017.8446728.

[9] Wang, X., Ju, S., and Wu, S. (2008). Challenges in Chinese text similarity research. in 2008 International Symposiums on Information Processing. IEEE, pp. 297-302, doi: 10.1109/ISIP.2008.76.

[10] Liu, X., Xu, C., and Ouyang, B. (2015). Plagiarism detection algorithm for source code in computer science education, International Journal of Distance Education Technologies (IJDET), C Vol. 13 (4), pp: 29-39.

[11] Ragkhitwetsagul, C., Krinke, J., and Clark, D. (2017). A Comparison of Code Similarity Analysers, RN, C Vol. 17 (04), pp: 04.

[12] Al-Khamaiseh, K. and ALShagarin, S. (2014). A survey of string matching algorithms, Int. J. Eng. Res. Appl, C Vol. 4 (7), pp: 144-156.

[13] El-Shishtawy, T. (2013). A hybrid algorithm for matching arabic names, arXiv preprint arXiv:1309.5657, C Vol.

[14] Ekbal, A., Saha, S., and Choudhary, G. (2012). Plagiarism detection in text using vector space model. in 2012 12th international conference on hybrid intelligent systems (HIS). IEEE, pp. 366-371, doi: 10.1109/HIS.2012.6421362.

[15] Ali, W., Rehman, Z., Rehman, A.U., and Slaman, M. (2018). Detection of plagiarism in Urdu text documents. in 2018 14th International Conference on Emerging Technologies (ICET). IEEE, pp. 1-6, doi: 10.1109/ICET.2018.8603616.

[16] Alzahrani, S.M., Salim, N., and Abraham, A. (2011). Understanding plagiarism linguistic patterns, textual features, and detection methods, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), C Vol. 42 (2), pp: 133-149.

[17] Saini, A., Sri, M.R., and Thakur, M. (2021). Intrinsic Plagiarism Detection System Using Stylometric Features and DBSCAN. in 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS). IEEE.

[18] AlSallal, M., Iqbal, R., Palade, V., Amin, S., and Chang, V. (2019). An integrated approach for intrinsic plagiarism detection, Future Generation Computer Systems, C Vol. 96 700-712.

[19] Haneef, I., Adeel Nawab, R.M., Munir, E.U., and Bajwa, I.S. (2019). Design and development of a large cross-lingual plagiarism corpus for Urdu-English language pair, Scientific Programming, C Vol. 2019.

[20] Sindhu, L. and Idicula, S.M. (2015). Fingerprinting based detection system for identifying plagiarism in Malayalam text documents. in 2015 International Conference on Computing and Network Communications (CoCoNet). IEEE.

[21] Osman, A.H., Salim, N., Binwahlan, M.S., Twaha, S., Kumar, Y.J., and Abuobieda, A. (2012). Plagiarism detection scheme based on Semantic Role Labeling. in 2012 International Conference on Information Retrieval & Knowledge Management. IEEE.

[22] Sahi, M. and Gupta, V. (2016). Efficiency comparison of various plagiarism detection techniques. in 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT). IEEE.

[23] Wang, S., Qi, H., Kong, L., and Nu, C. (2013). Combination of VSM and Jaccard coefficient for external plagiarism detection. in 2013 international conference on machine learning and cybernetics. IEEE.

[24] Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C., (2009). Introduction to algorithms, MIT press,

[25] Hakak, S.I., Kamsin, A., Shivakumara, P., Gilkar, G.A., Khan, W.Z., and Imran, M. (2019). Exact string matching algorithms: Survey, issues, and future research directions, IEEE access, C Vol. 7 69614-69637.

[26] Sun, Y., Ma, L., and Wang, S. (2015). A comparative evaluation of string similarity metrics for ontology alignment, Journal of Information &Computational Science, C Vol. 12 (3), pp: 957-964.

[27] Chen, H., (2012). String metrics and word similarity applied to information retrieval, Itä-Suomen yliopisto. Ph.D. thesis, Finland.

[28] Bulygin, L. and Stupnikov, S.A. (2019). Applying of Machine Learning Techniques to Combine String-based, Language-based and Structure-based Similarity Measures for Ontology Matching. in DAMDID/RCDL, Moscow, Russia.

[29] Gupta, D. (2016). Study on Extrinsic Text Plagiarism Detection Techniques and Tools, Journal of Engineering Science & Technology Review, C Vol. 9 (5), pp.

[30] Mahmoud, A. and Zrigui, M. (2017). Semantic similarity analysis for paraphrase identification in Arabic texts. in Proceedings of the 31st Pacific Asia conference on language, information and computation, Tunisia

[31] Lahitani, A.R., Permanasari, A.E., and Setiawan, N.A. (2016). Cosine similarity to determine similarity measure: Study case in online essay assessment. in 2016 4th International Conference on Cyber and IT Service Management. IEEE, pp. 1-6, doi: 10.1109/CITSM.2016.7577578.

[32] Foltynek, T., Ruas, T., Scharpf, P., Meuschke, N., Schubotz, M., Grosky, W., Gipp, B. (2019). Detecting Machine-obfuscated Plagiarism [Data set], University of Michigan - Deep Blue

[33] Wahle, J.P., Ruas, T., Foltýnek, T., Meuschke, N., and Gipp, B. (2021). Identifying Machine-Paraphrased Plagiarism, arXiv preprint arXiv:2103.11909, C Vol.

[34] Baltes, S., Kiefer, R., and Diehl, S. (2017). Attribution required: Stack overflow code snippets in GitHub projects. in 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C). IEEE.

CURRICULUM VITAE

Zina BALANI

[redacted]

ACADEMIC ACTIVITIES

Paper:

1. **Cloud Computing Security Challenges and Threats**
2. **Combining Approximate String Matching Algorithms and Term Frequency in the Detection of Plagiarism**