Comprehensive Plagiarism Detection in Malayalam Texts Through Web and Database Integration

Parvathy Raj, Meharuniza Nazeem, Rajeev R R, Anitha R, Navaneeth S

International Centre for Free and Open Source Solutions (ICFOSS)
Government of Kerala, Thiruvananthapuram, Kerala,India
parvathyraj12@gmail.com, meharuniza@icfoss.org, rajeev@icfoss.in,
anitha@icfoss.org, navaneeths@icfoss.org

***Abstract*—Plagiarism detection techniques have become essential for recognizing instances of plagiarism, particularly in the domain of academics where scientific papers and documents are of prime importance. We propose an application that offers a comprehensive solution for detecting plagiarism in scholarly articles written in Malayalam, enabling users to submit texts, analyze them for plagiarism, and review the results interactively. With the increasing accessibility of digital content, maintaining originality in academic writing has become more tedious. Our research addresses this challenge by providing a solution tailored to the Malayalam language. The application aids researchers and academic institutions in detecting potential plagiarism by accessing web-based content and algorithmic text analysis. The study significantly contributes to the field of plagiarism detection for low resource language such as malayalam and offers a practical way to preserve the originality of Malayalam scholarly work. The performance of four algorithms SequenceMatcher, N-Grams, Rabin-Karp, and Cosine Similarity is thoroughly evaluated. Cosine Similarity, with a 92.45% detection rate, outperformed the others, significantly surpassing Rabin-Karp(65.3%), N-Grams(58.7%) and SequenceMatcher(51.4%). Using this improved efficiency, a user-friendly web application was developed that integrates web search and database comparison features with the Cosine Similarity algorithm.**

Keywords: **Plagiarism detection, Similarity Algorithms, Regional Language Plagiarism**

I. INTRODUCTION

Plagiarism occurs when someone presents another person's work (ideas, words, or creations) as their own without acknowledging the original source.This undermines the core values of honesty and integrity essential for success in academia, professions, and creative endeavors. At its core, plagiarism undermines originality, honesty, and integrity, all of which are essential to the pursuit of knowledge and creative expression. The scope of plagiarism extends beyond simply copying text verbatim. Plagiarism detection employs various methods, broadly classified as similarity-based and logic-based techniques. The similarity-based techniques are the methods focused on identifying textual similarities between the document in question and other sources. They compare the document's content, such as words, phrases, and sentence structures, to a vast database of existing texts to find potential matches. The logic based techniques are simple text matching and analyze the underlying logic and structure of the document. They may examine things like argumentation, citation patterns, and the overall flow of ideas to identify inconsistencies or evidence of plagiarism.

Algorithms commonly employed in plagiarism detection include Rabin-Karp, Sunday's algorithm, N-grams methods, cosine similarity, Jaccard similarity, and Latent Semantic Analysis (LSA). These techniques operate by comparing texts or semantics of two or more recomposed documents. In professional and academic domains, plagiarism casts doubt upon one's originality and integrity. In creative industries, it constitutes an infringement of intellectual property rights, with potential legal ramifications. In the last twenty years, automatic systems for the detection of copying have changed greatly, for instance, in relation to the English language. Plagiarism detection systems utilize various techniques, including string matching, semantic analysis, and machine learning. However, their effectiveness can be limited by their reliance on specific language features. This poses challenges when dealing with languages with limited linguistic resources. Therefore, developing effective plagiarism detection systems requires careful consideration of the unique characteristics of each language to ensure accurate and reliable results.

Malayalam, a prominent Dravidian language spoken by over 38 million people, possesses a rich linguistic heritage. Its growing digital presence necessitates the development of robust plagiarism detection tools.

While challenges like the potential for mistranslation of nuanced features and the unique morphological characteristics of Malayalam need to be addressed, effective plagiarism detection systems are crucial for maintaining academic integrity, upholding professional standards, and promoting the ethical development and use of the Malayalam language in the digital age. These systems would ensure the originality of student work, protect intellectual property rights, and foster a culture of academic and professional honesty within the Malayalam-speaking community.

Considering these language barriers, it is vital to create plagiarism detection systems that consider specific characteristics of the Malayalam language. Such systems are important in upholding the ethical standards of scholarship as well as protecting the intellectual property of researchers, writers and content developers who write in Malayalam. It is important to ensure that such tools are developed because without them there is a high possibility of plagiarizing which will affect the standard and the integrity of scholarly and professional work produced in Malayalam.

This paper offers solutions for the detection of malayalam's plagiarism, which is not widely supported. It is the first study that employs four similarity-based methods: SequenceMatcher, NGrams, Rabin-Karp and Cosine Similarity for the analysis of Malayalam language. Cosine Similarity achieved the best performance when it comes to correlating comparable tasks in the Malayalam language. This piece also tackles the intricate structure of Malayalam as an agglutinative language. Quite importantly, a user-friendly web application is built for block multiplication practical purposes The methodology provided in this paper can help to form a base model for further application of the terms in the context of other morphologically elaborate and agglutinative languages in plagiarism detection.

II. RELATED WORKS

Garg et al. [1] introduced a plagiarism detection technology named Maulik, specifically designed for Hindi. It uses methods such as sentence structure analysis, word matching, and semantic analysis to effectively detect plagiarism in Hindi literature, particularly in academic settings. Gupta et al. [2] discussed a method for detecting external plagiarism (plagiarism occurring between documents) using fuzzy-semantic similarity and natural language processing (NLP) techniques, which produced encouraging results when tested on multiple Indian languages.

Lambda et al. [3] examined several plagiarism detection techniques for Indian languages like Marathi, Tamil, and Hindi. The study classified methods based on machine learning algorithms, NLP strategies, and similarity metrics, highlighting the challenges of plagiarism detection in regional languages due to the lack of linguistic resources and tools. D. Thenmozhi et al. [4] explored deep learning techniques for detecting paraphrases in Indian languages such as Hindi and Tamil, using encoder-decoder models to improve plagiarism detection accuracy. N. Naik et al. [5] focused on using semantic analysis for plagiarism detection in Marathi documents. Dixit et al. [6] reviewed various plagiarism detection tools and technologies, emphasizing their mechanisms and algorithms. Kulkarni et al.[7] explored different types of plagiarism, including syntactic and semantic plagiarism, with a focus on how techniques such as Naive Bayes, N-gram analysis, and NLP can yield positive results for regional languages like Hindi and Marathi. Lamba et al. [8] emphasized the need for customized methods for regional languages, proposing techniques like fingerprinting and winnowing.

Eman Al-Thwaib et al.[9] discussed a two-stage plagiarism detection technique for Arabic, leveraging a thesis repository and an algorithm that can detect various types of plagiarism, such as paraphrase and word order changes. This method also employed machine learning algorithms for cross-lingual plagiarism detection between English and Arabic, achieving significant improvements in performance when using Support Vector Classifiers (SVC) with multilingual encoders. The University of Jordan library's thesis repository is the first corpus that contains a number of PhD and postgraduate dissertations in order to detect plagiarism. The next stage involves developing a plagiarism detection system specifically for Arabic that can recognise several types of plagiarism, such as word order changes, synonym replacement, paraphrase, and exact copying. There are ways where semantic and syntactic information are extracted with the use of word location, word embedding, and word order where multilingual encoders are employed for sentence-level cross-language plagiarism detection between English and Arabic.

The identified characteristics are further integrated with several machine learning (ML) algorithms to assist in categorising phrases as either non-plagiarized or plagiarised content [10]. SemEval-2017 datasets have been utilised in the application and evaluation of this approach. A study of experimental data demonstrates that better outcomes are obtained when these collected features are combined with various machine learning (ML) classifiers. With an F1 score

of 0.879, the study also reveals that Support Vector Classifiers (SVC), when built with all characteristics, yield the best and most optimal outcome.

Research has been done on plagiarism detection methods that can detect syntactic parsing, synonym thesauri, and tracking citations. According to these findings, machine learning methods perform well when it comes to plagiarism detection [11]. While research on text-based plagiarism detection is advanced, methods for detecting images, drawn figures, tables, equations, and scanned documents are still lacking. Improved outcomes can be achieved by combing and combining several plagiarism detection techniques, particularly those based on machine learning. Combining textual and non-textual data with quality assessment criteria is one possible avenue for future research [12]. A benchmark corpus of 10,872 excellent documents organized at the sentence and paragraph granularity levels is offered by certain studies. This dataset is used for a number of tasks, such as intrinsic plagiarism detection, author clustering in Urdu, and verbatim text reuse identification. It also helps researchers and practitioners in natural language processing by making it easier to create plagiarism detection algorithms that are specific to the Urdu language [13]. These models help identify plagiarised information in Urdu literature by improving the precision of plagiarism detection, which is useful in publishing and education.

A collaborative test consisting of fifteen web-based text-matching algorithms appropriate for scenarios where plagiarism may be suspected has been researched. Researchers from seven different nations took part, using test materials in eight different languages to evaluate the performance of the algorithms on papers with many sources as well as single sources. The findings suggest that some systems do better than others when it comes to particular languages or language families. Major languages like English, German, and Spanish have more sources covered overall than minor languages like Czech or Slovak [14]. Numerous research examine approaches, features, classification, and obstacles related to the detection of plagiarism in source code and natural language. It gives a summary of popular plagiarism detection programs, their features, and the difficulties that can arise while using them. Additionally, the study establishes the foundation for creating future approaches and instruments that are more effective in addressing efficiency-related problems [15].

III. METHODOLOGY

This section describes the process, which includes managing text data in Malayalam, looking out pertinent online resources, and determining how comparable various data sources appear. The process begins with input documents containing Malayalam content in formats like DOC, TXT, ODT, or PDF. These documents are tokenized into smaller chunks for analysis, and the tokenized text is stored in an internal database for later comparison, as shown below. 1.

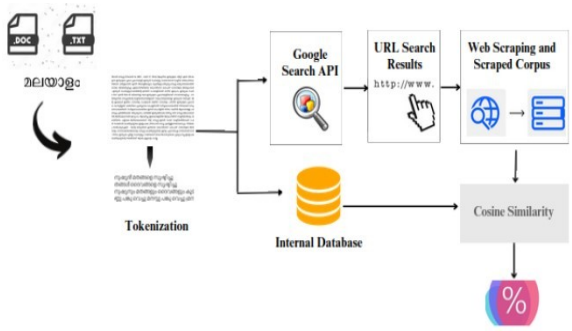

Fig. 1. Model Architecture

After tokenization, the system uses the Google Search API to retrieve URL search results related to the Malayalam input. The content from these URLs is extracted using web scraping, forming a ”scraped corpus.” Cosine similarity is then calculated between the scraped corpus and the original tokenized text. Cosine similarity measures the similarity between two text vectors in multidimensional space, providing a score between 0 (no similarity) and 1 (perfect similarity).

The last step is to find the cosine similarity between the text from the scraped web corpus and the tokenised internal text from the original documents. A popular metric in natural language processing is cosine similarity, which calculates the degree of similarity between two text passages by visualising them as vectors in a multidimensional space. The similarity score is given here as a percentage and runs from 0 (no resemblance) to 1 (perfect similarity). This score is useful for things like answering questions, detecting plagiarism, and cross-referencing material by indicating how closely the web text matches internal Malayalam papers. This methodology offers a solid strategy for evaluating Malayalam literature, gathering data from the internet, and contrasting it with regional statistics to determine relevance and accuracy.

IV.IMPLEMENTATION

A. Database creation and retrieval:

Data was collected from two sources: the Malayalam monthly Thudi and the Shodhganga thesis database. After identifying relevant papers, Tesseract OCR was used to extract Malayalam text from the PDFs, followed by pre-processing to clean the text. The cleaned text was stored in an SQLite database with metadata, including titles, authors, and publication dates.

1) Data Collection

The well-known Malayalam periodical Thudi and the sizable Indian theses and dissertations database Shodhganga served as the study's two primary data sources. The approach begins with locating relevant research papers in Malayalam and is subsequently refined by using certain keywords and filters. To locate pertinent records that are within the scope of the study, both sources are extensively investigated. After being identified, the documents are downloaded, typically in PDF format, and stored in a carefully organised folder to make processing easier later in the workflow.

SQLite was chosen for this study due to its simplicity, ease of integration, and sufficiency for handling the scope of the application. The study required a lightweight, file-based database that could manage structured text data with metadata such as titles and authors. SQLite provided an efficient way to store and query this data without the overhead of a server-based database system.

2) Data Extraction using Tesseract OCR

Tesseract OCR (Optical Character Recognition) is used to transform the text from the downloaded PDFs into a machine-readable format after data collection. In order to ensure correct Malayalam text extraction, the method starts with setting Tesseract OCR with a Malayalam language pack.

We create a script to do the following tasks automatically:

- Open every PDF file.
- Create a picture from each page if necessary.
- To extract text from the photos or straight from the PDF, use Tesseract.

However, because of the intricacies of the Malayalam script and frequent OCR faults, the recovered text frequently has mistakes or unnecessary letters. The collected text goes through a pre-processing step in order to address this:

- Cleaning the text: content cleaning includes deleting superfluous characters, correcting frequent OCR misunderstandings, and standardising content to ensure consistency.
- Tools for normalising text: The output is refined using regular expressions and language-specific libraries to make sure the retrieved text is precise and prepared for analysis.

The text is saved in a SQLite database for convenient future access after it has been extracted and cleaned. Each document's key metadata is represented by clearly specified tables and columns in the database structure. Among these columns are:

- Title
- Author
- Publication date
- Cleaned text content

3) Database Construction

Because every document entry is saved along with its associated metadata, future analysis will be able to search, retrieve, and compare documents with ease. The content is readily available for analysis, such as content comparison or plagiarism detection, thanks to the organised database format.

B. Pre-processing and Tokenization:

Tokenising the gathered content is the first step in the research process; this entails dividing the text into more digestible, smaller chunks known as tokens. Tokenisation is an important step because it makes the text easier to absorb and analyse by breaking it up into discrete words, sentences, or other relevant parts. Finding patterns, connections, and underlying structures in the text is made possible by this segmentation, and this is crucial for tasks like content comparison and plagiarism detection.

Users can enter the text directly into a specific text area on the interface or submit a document containing the text in the earliest stages of the procedure. After submission, the text is tokenised, which divides the content into manageable chunks or sentences using the period (.) as a delimiter. Every sentence or part is handled as a unique token, which is subsequently investigated further on its own. The algorithm can do more thorough evaluations by independently evaluating these smaller pieces, making it easier to spot patterns like unique phrases, repeating structures, or replicated information. Due to the independent processing of each token, this granular method enables more accurate analysis and provides deeper insights into the structure and meaning of the information.

C. Web search and corpus creation

This stage involved searching the internet for each token or passage of the supplied text using the Googlesearch Python package. To make sure that only genuine web pages were taken into consideration, the search results were filtered to retain only authentic,

non-empty URLs. Using Python's urlparse package, the domains of these URLs were extracted in order to count and prevent duplicate domains. The requests library was used to make an HTTP GET request with a 30-second timeout for each valid URL. BeautifulSoup was then used to parse the HTML content of the website and extract the primary text, with an emphasis on paragraph (¡p¿) components. In the following stages of the study, similarity analysis was performed on the corpus that contained the retrieved text.

D. Feature extraction and Similarity checking:

In document comparison and plagiarism detection, feature extraction and similarity verification are essential procedures. The process of feature extraction entails locating and separating important elements from text, such as linguistic patterns, sentence structures, and word frequencies. These elements function as a condensed depiction of the content in the document. Following that, similarity checking compares several texts to ascertain how much they resemble one another using these features that were extracted. The paragraph discusses the application of three alternative algorithms for similarity analysis, suggesting a comparative comparison of several techniques for determining textual similarity.

This study's utilisation of several algorithms enables a thorough assessment of various similarity detecting methods. Every algorithm probably uses a different approach to compute similarity scores between texts and to mathematically represent the collected information. With this strategy, researchers can evaluate how effective different techniques are, pinpoint each one's advantages and disadvantages, and even combine methods to get more precise results. Research of this kind is essential to the development of content recommendation systems, plagiarism detection programs, and other applications that use textual similarity analysis. This study's comparative design indicates an attempt to raise the precision and dependability of similarity analysis in diverse settings.

1) N grams similarity algorithm:

The n-gram technique detects plagiarism by comparing the n-grams, or substrings of length n, of two text passages and calculating how similar they are First, the original text and the possibly plagiarized text are divided into n-grams [16], or overlapping sequences of n characters, in the context of Malayalam. The two texts are then compared using these n-grams to find any overlaps or commonalities.

The program then determines the frequency with which each n-gram occurs in the two texts. By applying a similarity metric such as the Jaccard similarity coefficient, the technique evaluates how many n-grams are shared between the two texts relative to the total number of unique ngrams. A greater probability of plagiarism is indicated by a higher similarity score [17]. The following are the steps in the n-gram comparison algorithm:

N-gram Extraction: Every input string is divided into n-character sequences that overlap. For instance, the n-grams for the string ”hello” with n=3 would be ”hel”, ”ell”, and ”llo”. Counting N-grams: Following the separation of the strings into n-grams, the next stage is to tally the number of times each distinct n-gram appears in each of the two strings. This aids in determining whether n-grams are more prevalent or overlap between the two strings [18]. Finding Similarity: The method counts the two strings and then compares their n-gram sets. The Jaccard similarity metric is used to obtain the similarity score by calculating the degree of overlap between the two sets.

$$\text{Jaccard Similarity} = \frac{|A \cap B|}{|A \cup B|} \qquad (1)$$

Where:

- $A \cap B$ is the number of common n-grams (intersection),
- $A \cup B$ is the total number of unique n-grams from both strings (union).

2) Cosine similarity

Cosine similarity, which converts two texts into vectors and measures their similarity, is a commonly used method in Malayalam plagiarism detection. In a multi-dimensional space, every text or document is represented as a vector, with each dimension denoting a distinct word across the text corpus. The degree to which the texts are similar based on word usage is indicated by the cosine similarity [19], which calculates the cosine of the angle between these two vectors. Since the focus of this method is on the direction of the vectors rather than their magnitude, it may identify similarities in texts of different lengths, making it especially effective for Malayalam.

In order to apply cosine similarity in Malayalam, the texts must first undergo tokenisation, a preprocessing step in which every word is dissected and stop words—common, unimportant words—are eliminated. These terms are frequently vectorised using the term frequency inverse document frequency (TF-IDF) approach. For every vector in the dataset, the frequency of a word is correlated with its relevance. The cosine of the angle between these vectors is used to compute the similarity score [20]. A value of 1 indicates that the vectors and, by extension, the texts, are identical, while a value of 0 indicates that they are wholly unlike. This technique is robust for

detecting semantic similarities in Malayalam texts and performs well even when the plagiarism comprises small adjustments like sentence reordering or synonym replacement.

$$\text{Cosine Similarity} = \frac{A \cdot B}{\|A\|\|B\|} \quad (2)$$

Where:

- $A \cdot B$ is the dot product of vectors A and B,
- $\|A\|$ is the Euclidean norm (magnitude) of vector A,
- $\|B\|$ is the Euclidean norm of vector B.

The cosine similarity value ranges from -1 to 1:

- 1 indicates perfect similarity (the vectors are identical),
- 0 indicates orthogonality (no similarity),
- -1 indicates complete dissimilarity (opposite vectors).

3) The Rabin Karp Algorithm

Hashing is a technique used by the RabinKarp algorithm to quickly identify a pattern in a longer text. Rather than comparing the complete pattern to every substring in the text, it computes the pattern's hash value first, then compares it to the hash values of all the text's substrings that have the same length shown below 2. The method then verifies that the pattern has been found by looking at the real characters if the hash values match. This two-step procedure, which compares hashes and then verifies characters, considerably accelerates the search process.

This technique may calculate hash values for numerous patterns at once, making it very useful when numerous patterns need to be searched at once. The efficacy of this method improves with the quantity of patterns it searches, which makes it perfect for extensive text matching jobs like plagiarism detection. When looking for many patterns or lengthy texts, the algorithm is faster than standard search methods because it uses hashing to limit the amount of needless character comparisons.

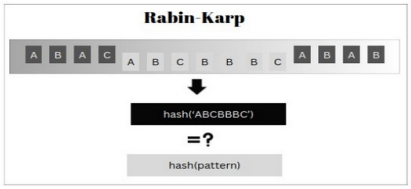

Fig. 2. Rabin- Karp hash table

$$h(S) = \sum_{i=0}^{m-1} S[i] \cdot d^{m-1-i} \mod q \quad (3)$$

Where:

- S is the string (either the pattern or a substring of the text), m is the length of the string S,
- d is the number of possible characters in the input (for example, the size of the alphabet), • q is a prime number used to reduce the hash value.

V. RESULT AND DISCUSSION

This study tested four algorithms: SequenceMatcher, N-Grams, Rabin-Karp, and Cosine Similarity. Cosine Similarity achieved the highest detection rate at 92.45%, followed by Rabin-Karp (65.3%), N-Grams (58.7%), and SequenceMatcher(51.4%). A web application using the Cosine Similarity algorithm was developed, offering a user-friendly interface for plagiarism detection. The similarity percentages for all four algorithms are displayed in Figure 3 shown below, which was created as a means of testing each algorithm's accuracy in detecting similarities between texts.

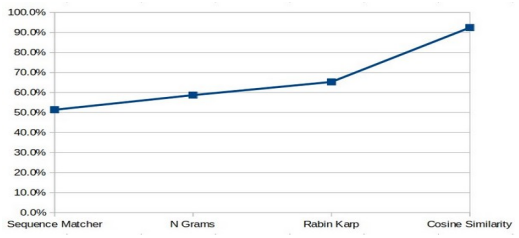

Fig. 3. Similarity percentage of different algorithms

However, it was less successful in identifying content that had been paraphrased. By calculating the cosine of the angle between two vectors in multidimensional space, the Cosine Similarity algorithm, which was able to discover both precise and close matches, finally displayed the highest similarity percentage (92.45%).

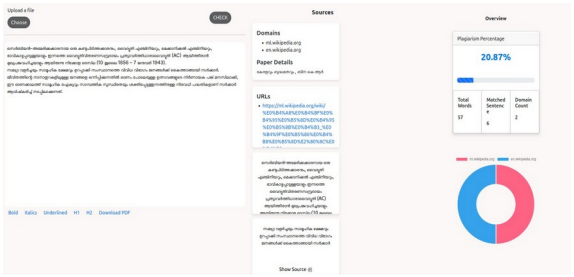

Fig. 4.Malayalam Plagiarism Checker

Using the Cosine Similarity algorithm, an application was created in the second research phase with an emphasis on efficiency and simplicity. To make the plagiarism-checking process go more smoothly, the application is divided into three columns, each of which has a specific function. Users can upload or enter text for analysis in the first column, and there are options to format the text and get a comprehensive PDF report that includes a summary of the results.

Convenience and flexibility are provided by the ability for users to paste text directly or upload text files.

The subsequent column provides the findings of the plagiarism check in an easy-to-read, organised format. Important details are highlighted in this column, such as the proportion of identified plagiarism, pertinent sources, and lines that align with other content. The results are initially suppressed for clarity, but they can be extended for a more thorough investigation. To improve user engagement, more tools and options are available in the third column. This includes tools like interactive visualisations that visually show the analytic findings and filters to narrow down results and prioritise information. As seen in Figure 5, these features facilitate users' interpretation and action on the data.

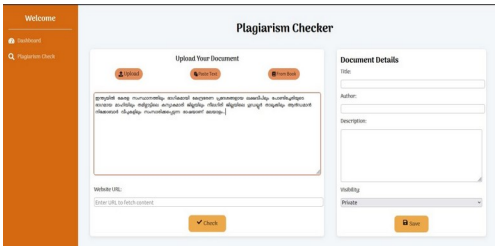

Fig. 5.Malayalam Plagiarism Checker

The user interface of a plagiarism checker program is seen in this image. Three pieces make up the main screen: a document details panel on the right, a central area for text input and document upload, and a navigation menu on the left. The application provides several methods for entering text for verification, such as importing from a book, direct text pasting, and file upload. For content fetching, you can also specify the URL of a website. The Malayalam text that is shown in the input box indicates that the interface accepts scripts other than Latin.

Users can add metadata, including author, title, and description, to the submitted content by going to the document details area. A visibility setting is also there, and since it is currently set to "Private," users may be able to restrict who can view their documents. Orange and white make up the majority of the colour scheme, which is simple and clean overall. This tool's features and style suggest that it is made to be user-friendly, however it has extensive plagiarism checking capabilities that can be used to a wide range of sources and document types.

VI. CONCLUSION AND FUTURE WORKS

This study highlights the importance of addressing plagiarism detection in regional languages. The spread of digital content has made it easier for people to share knowledge in local languages, but it has also made plagiarism in these languages more likely. It is essential to create plagiarism detection techniques that are specifically suited to regional languages in order to preserve academic integrity, safeguard intellectual property, and encourage originality in scholarly work. Several plagiarism detection methods for Malayalam texts were investigated throughout the experimental phase. With a high similarity detection rate of 92.45%, Cosine Similarity was shown to be the most successful among them. This indicates its potential as a strong option for thorough Malayalam plagiarism detection. The accuracy of Rabin-Karp (65.3%), N-Grams (58.7%), and SequenceMatcher (51.4%) was much lower, highlighting the need for more sophisticated algorithms to handle the subtleties of academic literature. Building on these discoveries, the subsequent phase would entail incorporating the Cosine Similarity algorithm into an online application, providing a dependable and easy-to-use instrument for identifying and averting plagiarism in Malayalam academic writing.

Integrate semantic analysis for plagiarism detection to identify closely paraphrased content and cross-lingual copying. Leveraging transformer models can effectively capture these subtle nuances. Incorporating NoSQL databases like Elasticsearch would be highly beneficial for scaling the current system to production. A web-based plagiarism detection tool leveraging this technology will significantly enhance and uphold academic integrity in research publications, particularly in Malayalam.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to Dinesh Lal D L for his invaluable coordination and support throughout our study. Special thanks are also extended to Sabeerali K P, B Arya Jagish, , Adarsh J, and Yadu Krishnan P K for their exceptional technical expertise, which was instrumental in the successful completion of this research.

REFERENCES

[1] A.Garg and M.Goyal, Maulik: A Plagiarism Detection Tool for Hindi Documents, Proceedings of the 9th International Conference on Natural Language Processing (ICON 2016), 2016, pp. 80–85.

[2] S.Gupta R.Chatterjee, and S.Bhatnagar, Using Natural Language Processing Techniques and Fuzzy-Semantic Similarity for Automatic External Plagiarism Detection, International Journal of Computational Linguistics Research, vol. 5, no. 2, pp. 42-49, 2014.

[3] N.Lamba and S.Govilkar, A Survey on Plagiarism Detection Techniques for Indian Regional Languages in 2017 International Conference on Computing Communication

Control and Automation (ICCUBEA), Pune, India, 2017, pp. 1–6.

[4] D.Thenmozhi and S.Kayalvizhi, Paraphrase Detection in Indian Languages Using Deep Learning in Proceedings of the International Conference on Computational Intelligence in Data Science (ICCIDS 2020), Chennai, India, 2020, pp. 30–42.

[5] N.Naik and K.Landge, Plagiarism Detection in Marathi Language Using Semantic Analysis in Proceedings of the 4th IEEE International Conference on Computing Communication, Control and Automation (ICCUBEA), Pune, India, 2019, pp. 1–7.

[6] Dixit R, Hegde S,and P. Pai, A Literature Review on Plagiarism Detection in Computer Programming Assignments, International Research Journal of Engineering and Technology (IRJET), 2022.

[7] Sagar Kulkarni,Sharvari Govilkar,and Dhiraj Amin , Analysis of Plagiarism Detection Tools and Methods, Proceedings of the 4th International Conference on Advances in Science and Technology (ICAST2021), 17 Jun 2021.

[8] Harshall Lamba, Sharvari Govilkar , A Survey on Plagiarism Detection Techniques for Indian Regional Languages, International Journal of Computer Applications (0975 – 8887) Volume 164 – No 4, April 2017.

[9] Eman A Thwaib, Bassam H Hammo and SaneYag, An academic Arabic corpus for plagiarism detection: design, construction and experimentation, International Journal of Educational Technology in Higher Education, 2020.

[10] Naif Alotaibi and Mike Joy, English-Arabic Cross-language Plagiarism Detection, Proceedings of Recent Advances in Natural Language Processing, pages 44–52 ,Sep 1–3, 2021.

[11] Anchal Pokharana and Urvashi Garg, A Review on diverse algorithms used in the context of Plagiarism Detection, 2023 International Conference on Advancement in Computation & Computer Technologies (InCACCT), 05-06 May 2023.

[12] Tomas Foltynek, Norman Meuschke, and Bela Gipp, Academic Plagiarism Detection: A Systematic Literature Review, ACM Computing Surveys, Vol. 52, No. 6, Article 112. Publication date: October 2019.

[13] Muhammad Haseeb, Muhammad Faraz Manzoor, Muhammad Shoaib Farooq, Uzma Farooq and Adnan Abid, A versatile dataset for intrinsic plagiarism detection, text reuse analysis, and author clustering in Urdu, Published by Elsevier Inc. This is an open access article under the CC BY license.

[14] Tomas Foltynek, Dita Dlabolova, Alla AnohinaNaumeca, Salim Razi, Julius Kravjar, Laima Kamzola, Jean Guerrero-Dib, Ozgur Celik, and Debora Weber-Wulff, Testing of support tools for plagiarism detection, International Journal of Educational Technology in Higher Education,2020.

[15] MAC Jiffriya, MAC Akmal Jahan, and RG Ragel, Plagiarism detection tools and techniques: A comprehensive survey, Journal of Science-FASSEUSL (2021) 02(02) 47-64.

[16] Nicholas Gahman and Vinayak Elangovan, A Comparison of Document Similarity Algorithms, International Journal of Artificial Intelligence and Applications (IJAIA) , Vol. 14, No.2, March 2023.

[17] Abdul Fadlil, Sunardi, and Rezki Ramdhani,
Similarity Identification Based on Word Trigrams Using Exact String Matching Algorithms, INTENSIF, Vol.6 No.2 August 2022.

[18] Ifeanyi-Reuben Nkechi J, Ugwu Chidiebere, and Nwachukwu E. O, Comparative Analysis of Ngram Text Representation on Igbo Text Document Similarity, International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868.

[19] Tebatso Gorgina Moape, Oludayo O. Olugbara, and Sunday O. Ojo, Integrating Lesk Algorithm with Cosine Semantic Similarity to Resolve Polysemy for Setswana Language, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 15, No. 4, 2024.

[20] Kylie L. Anglin, Vivian C. Wong, and Arielle Boguslav, A Natural Language Processing Approach to Measuring Treatment Adherence and Consistency Using Semantic Similarity Arielle Boguslav, AERA Open January-December 2021, Vol. 7, No. 1, pp. 1–18.

[21] Vipkas Al Hadid Firdaus, Implementation of Rabin Karp Algorithm in E-Commerce Search Box Feature (Case Study: Sinar Baja Store), 22 Jun 2023 - SISFORMA - Vol. 10, Iss: 1, pp 19-25.

[22] Dimas Dwi Ichtiarto, Penerapan Algoritma Rabin-Karp Pada Sistem Tracer Study Fakultas Teknologi Informasi UNISBA Blitar Berbasis Web, 05 Mar 2024 - Journal Zetroem .

[23] Riskya Fajar Ningtyas, Siti Nurhayati, Muhammad Riandi Widiyantoro, Salahudin Robo, and Mursalim Tonggiroh, The application of RabinKarp algorithm and stemming to detect similarity of electronic documents, Journal of Nucleation and Atmospheric Aerosols, 01 january 2024.