SINHALA SIMILARITY AND SEMANTIC PLAGIARISM DETECTION USING MODERN EMBEDDING MODELS

Final Report

IT21163968- Abhayasiri S.A.U. D

IT21187414-Ranaweera R.R.M.I.M -

B.Sc. (Hons) in Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

April 2026

SINHALA SIMILARITY AND SEMANTIC PLAGIARISM DETECTION USING MODERN EMBEDDING MODELS

Final Report

IT21163968- Abhayasiri S.A.U. D

IT21187414-Ranaweera R.R.M.I.M -

B.Sc. (Hons) in Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

April 2026

Declaration

I declare that this is my own work, and this dissertation does not incorporate without acknowledgement any material previously submitted for a degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to Sri Lanka Institute of Information Technology, the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Name	Student ID	Signature
Abhayasiri S.A.U. D	IT21163968	
Ranaweera R.R.M.I.M	IT21187414	

The above candidates are carrying out research for the undergraduate Dissertation under my supervision.

Name of the Supervisor: Mr. Dinith Primal

Signature of the Supervisor: ……. Date: ……

Acknowledgement

Abstract

Keywords:

Table of Contents

Declaration 3
Acknowledgement 4
Abstract 5
1.INTRODUCTION 8
1.1 Background and Context 8
1.2 Problem Statement 9
1.3 Project Overview: Similarity.lk 9
1.4 Team Roles and Contributions 10
2.BACKGROUND AND LITERATURE REVIEW 11
2.1 Overview of AI in Aquaculture 11
2.2 Existing Systems and Tools 12
2.2.1 Deep learning-based Sinhala plagiarism detection 12
2.2.2 Sinhala web-based plagiarism detection 12
2.2.3 Sinhala sentence similarity using Siamese networks 12
2.3 Gaps in Existing Solutions 12
2.4 Proposed AI Techniques for Similarity.lk 13
2.4.1 Sinhala similarity detection 13
2.4.2 Cross language semantic detection 13
2.4.3 Comparison with Alternatives 13
2.5 Dataset Relevance 13
2.5.1 Dataset for Similarity Plagiarism 13
2.5.2 Dataset for Semantic Detection 13
3. METHODOLOGY 14
3.1 Introduction to Methodology 14
3.2 Research Design 15
3.3 Overall System Methodology 15
3.4 Data Collection Methodology 16
3.5 Data Preprocessing Methodology 17
3.6 Component 1 Methodology: Sinhala Similarity Plagiarism Detection 18
3.7 Component 2 Methodology: English to Sinhala Semantic Plagiarism Detection 18
3.8 System Architecture 19
3.9 Commercialisation Aspects of the Product 20

3.10 Ethical and Legal Considerations....21
4. TESTING AND IMPLEMENTATION 22
4.1 Introduction to Testing and Implementation....22
4.2 Implementation Environment....22
4.3 Implementation of Document Upload and Text Extraction....23
4.4 Implementation of Sinhala Similarity Detection....23
4.5 Implementation of English to Sinhala Semantic Detection....24
4.6 Implementation of Report Generation....24
4.8 Model Testing....26
4.9 Integration Testing....26
4.10 Usability Testing....26
5. RESULTS AND DISCUSSION 27
5.1 Introduction to Results and Discussion....27
5.2 Dataset Preparation Results....27
5.3 Sinhala Similarity Detection Results....28
5.4 English to Sinhala Semantic Detection Results....29
5.5 Integrated System Results....30
5.6 Research Findings....30
5.7 Discussion....31
5.8 Summary of Each Student's Contribution....32
6. CONCLUSION AND RECOMMENDATIONS 33
6.1 Conclusion....33
6.2 Achievement of Objectives....34
6.3 Main Contributions....34
6.4 Limitations....35
6.5 Recommendations....35
6.6 Final Summary....36
REFERENCES 37

1.INTRODUCTION

1.1 Background and Context

Academic writing depends on originality, proper citation, and honest use of sources. When plagiarism enters student work, reports, or publications, it weakens academic trust and reduces the value of genuine research. This problem has become harder to manage because digital content is now easy to access, copy, edit, paraphrase, and translate. In practice, many plagiarism cases no longer appear as direct copying. Writers can reorder words, replace terms with synonyms, or translate English content into Sinhala and present it as original work. These forms of disguised plagiarism are difficult to detect through manual reading alone.

Existing presentation material for this project highlights this exact challenge. It notes that plagiarism weakens academic credibility, that most existing tools focus on English, and that Sinhala work is often handled manually in Sri Lankan academic settings. It also identifies machine translation as a major route through which English content can be repackaged in Sinhala.

Sinhala poses added challenges for plagiarism detection because it is a low resource language in the natural language processing domain. Compared with English, it has fewer openly available corpora, fewer mature language tools, and less prior work on sentence similarity and plagiarism detection. Prior studies confirm this gap. Rajamanthri and Thelijjagoda state that only a few systems exist for Sinhala and that no effective system was available to compare Sinhala documents with internet resources at the time of their work. Their system reached 88 percent accuracy using text preprocessing, Google searching, and Jaccard comparison. KasthuriArachchi and Charles proposed a deep learning approach based on word embeddings and reported 97 percent accuracy, but their evaluation used a small dataset built from 50 news documents. Nilaxan and Ranathunga later showed that a Siamese LSTM with fastText embeddings improved Pearson correlation by 3.61 percent for Sinhala sentence similarity over earlier conventional methods using 5000 Sinhala sentence pairs.

These findings show a clear pattern. Sinhala plagiarism detection has promising starting points, but the field still lacks a complete, scalable, and modern solution that can work across both monolingual Sinhala plagiarism and cross language English to Sinhala plagiarism.

1.2 Problem Statement

Current plagiarism detection tools do not serve Sinhala users well. The strongest tools in the market are mainly designed for English and often miss the linguistic and semantic behaviour of Sinhala text. At local level, prior Sinhala studies have improved the field, yet each has important limits. One model relied on Word2Vec similarity over a small dataset. Another used rule-based web comparison with Jaccard similarity and struggled with deep paraphrase. A later Siamese sentence similarity model improved semantic matching, yet it did not form a complete end to end plagiarism system. Project documents for this study identify the same issues clearly: no current tool flags cross language plagiarism between English and Sinhala, small private datasets reduce trust in published results, and previous work has not connected strong sentence similarity modelling to a full document level detector.

The core problem, then, is not only the absence of Sinhala support. It is the absence of an integrated platform that can detect copied, paraphrased, and translated plagiarism accurately, quickly, and in a way that users can understand. This gap creates a practical need for a system that combines modern embeddings, web based source retrieval, sentence level scoring, and clear reporting.

1.3 Project Overview: Similarity.lk

Similarity.lk is proposed as an automated plagiarism detection platform built for Sinhala and English Sinhala semantic comparison. The project aims to create one system with two connected detection paths. The first path handles Sinhala similarity plagiarism by finding direct copying and paraphrased overlap within Sinhala text. The second path handles semantic plagiarism by identifying Sinhala content translated from English sources. The project objective, as stated in the proposal and topic assessment documents, is to build a plagiarism checker that handles Sinhala texts and Sinhala to English translation related plagiarism, works at sentence level, gives clear match scores, and searches web pages in real time.

The technical design follows a service-based flow. The system first gathers candidate Sinhala and English source pages through a focused crawler. It then preprocesses text through cleaning, tokenisation, stop word removal, and normalisation. For Sinhala similarity detection, it uses shared embeddings and a Siamese LSTM to measure semantic overlap. For cross language detection, it uses bilingual sentence embeddings such as LaBSE, with translation fallback where needed. The engine ranks candidate matches, applies tuned thresholds, and generates a report with matched lines, source links, and originality scores.

This design makes Similarity.lk more than a research model. It becomes a usable platform that can support students, lecturers, researchers, and institutions that work in Sinhala.

1.4 Team Roles and Contributions

The project follows a two-component team structure under the supervision of Mr. Dinith Primal. The group members are Ranaweera R.R.M.I.M. and Abhayasiri S.A.U.D. Ranaweera leads the Sinhala similarity plagiarism component. This role includes collecting a large Sinhala corpus, labelling plagiarised and paraphrased documents, training a Siamese LSTM with fastText or multilingual BERT embeddings, tuning thresholds, and exposing the model as a service. The proposal identifies this contribution as the first Sinhala plagiarism component that uses contextual sentence embeddings with a balanced public corpus.

Abhayasiri leads the semantic plagiarism component for English Sinhala translation cases. This role includes building a parallel Sinhala English dataset, training a cross-language similarity model with LaBSE or a translation step, crawling English sources, and merging both model outputs into one report. The same document describes this as the first Sri Lankan system intended to flag translated plagiarism between English and Sinhala using bilingual sentence vectors.

The later development records also show practical implementation work. The Sinhala component evolved into a Flask based web application with secure login, upload, dashboard, text extraction, OCR support, sentence level comparison, and PDF report generation. The semantic component extended the same platform with reusable preprocessing, modular backend design, unified result management, and standardised reporting across both modules.

1.5 Scope and Significance

The scope of Similarity.lk covers sentence level plagiarism analysis, source retrieval, semantic comparison, and report generation for two major use cases. The first is Sinhala to Sinhala plagiarism, including copied and paraphrased text. The second is English to Sinhala semantic plagiarism, where source meaning is preserved across translation. The system does not aim to solve every authorship problem in Sinhala. Instead, it focuses on one strong and practical objective: detecting overlap in meaning and wording across monolingual and cross language contexts.

Its significance is high in both academic and technical terms. In academic settings, the system can reduce manual effort, improve fairness, and support originality checks in a language that has received limited research attention. In technical terms, it extends prior work by combining contextual embeddings, sentence similarity, bilingual modelling, focused crawling, and real time reporting inside one platform. The project also has value because it seeks better datasets, balanced evaluation, and clearer reproducibility than several earlier Sinhala studies.

2.BACKGROUND AND LITERATURE REVIEW

This literature review examines the research background that leads to Similarity.lk. It starts by considering the wider role of AI in text analysis, then reviews existing systems and tools, identifies current gaps, outlines the proposed technical direction, compares the project with alternatives, and explains the relevance of the selected datasets.

2.1 Overview of AI in Aquaculture

AI in aquaculture usually refers to smart monitoring, feeding prediction, water quality analysis, fish disease detection, and decision support systems driven by data. Though this project is not an aquaculture system, the same AI logic is relevant here. In both domains, AI helps detect patterns that are difficult to identify manually, especially when the data is large, noisy, and multilingual or multi source. In aquaculture, AI extracts useful signals from sensor or image streams. In Similarity.lk, AI extracts semantic signals from text and compares them across documents and languages. This makes pattern recognition, classification, and similarity learning central ideas in both fields. For this reason, the literature on applied AI systems is useful as a conceptual background, even though the direct application here is natural language processing rather than aquatic production.

2.2 Existing Systems and Tools

2.2.1 Deep learning-based Sinhala plagiarism detection

KasthuriArachchi and Charles proposed a Sinhala plagiarism detector based on word embeddings, cosine similarity, and soft cosine similarity. Their model represented words and sentences as vectors and reported 97 percent accuracy. It could detect direct copying and some sophisticated changes such as synonym replacement and word order changes. Still, the study used a relatively small dataset built from 50 documents, which limits confidence in wider generalisation.

2.2.2 Sinhala web-based plagiarism detection

Rajamanthri and Thelijjagoda developed a system that compared Sinhala documents against internet resources using preprocessing, Google searching, and Jaccard coefficient-based similarity. The model reached 88 percent accuracy and addressed an important local gap by moving from only local document comparison to web-based detection. Yet the technique remained largely surface based and less capable of deep semantic matching.

2.2.3 Sinhala sentence similarity using Siamese networks

Nilaxan and Ranathunga introduced a Siamese neural network with LSTM branches and fastText embeddings for Sinhala and Tamil sentence similarity. For Sinhala, the model improved Pearson correlation by 3.61 percent over previous approaches on 5000 sentence pairs. This was important because it shifted the field from word overlap toward semantic sentence understanding. Still, the work focused on sentence similarity as a task rather than a full plagiarism platform.

2.3 Gaps in Existing Solutions

The review shows 4 main gaps. First, datasets remain small or narrow in domain. Second, most earlier systems do not combine semantic similarity with web scale retrieval and document level reporting. Third, cross language plagiarism between English and Sinhala is largely untreated. Fourth, prior systems do not provide a single platform that joins monolingual and translated plagiarism detection in a reusable architecture. These exact gaps are identified in the project assessment and proposal slides, which call for a balanced Sinhala corpus, contextual embeddings, web scale crawling, bilingual embeddings, and real time API level response.

2.4 Proposed AI Techniques for Similarity.lk

2.4.1 Sinhala similarity detection

The Sinhala branch uses fastText and multilingual BERT for embeddings and a Siamese LSTM for sentence level matching. The design aims to capture synonym change, paraphrase, and semantic overlap beyond direct lexical copying.

2.4.2 Cross language semantic detection

The cross-language branch uses LaBSE or related bilingual embeddings, with MarianMT as a translation fallback. This supports semantic comparison between Sinhala suspicious text and English source candidates.

2.4.3 Comparison with Alternatives

Compared with earlier Sinhala work, Similarity.lk combines contextual embeddings, bilingual sentence vectors, focused crawling, public dataset intent, and one unified reporting pipeline. Proposal material shows that earlier studies did not jointly provide a large balanced corpus, contextual embeddings, deep paraphrase handling, web crawl support, cross language plagiarism handling, or sub second API response.

2.5 Dataset Relevance

2.5.1 Dataset for Similarity Plagiarism

For Sinhala similarity detection, the selected datasets are relevant because they cover both large scale pretraining and cleaner evaluation use. SEACrowd cc100 offers a high-volume Sinhala crawl that can support large language coverage. SinhalaCorpusLarge adds cleaned long form prose useful for semantic model training. Navanjana Sinhala Sumarization offers long passages that support near duplicate analysis across multi paragraph text.

SinhalaWikipediaArticles gives cleaner reference sentences for threshold tuning, while NSINA adds domain rich news data with metadata. Together, these datasets support both robustness and controlled evaluation.

2.5.2 Dataset for Semantic Detection

For cross language plagiarism detection, WikiMatrix and CCaligned provide large aligned English Sinhala pairs for bilingual embedding learning. JW300 adds stylistic variety through long sentence structures. OpenSubtitles adds colloquial phrasing and lexical diversity. FLORES serves as a gold standard evaluation set that should be reserved for tuning and final testing. Tatoeba offers short clean pairs for sanity checks, while OPUS GNOME gives concise technical strings for robustness tests on short segments. This mix is suitable because translated plagiarism can appear in formal prose, conversational wording, short copied fragments, and domain mixed content.

3. METHODOLOGY

3.1 Introduction to Methodology

This chapter explains the research and development methodology used to design, build, and evaluate Similarity.lk. The project follows an applied research approach because it addresses a practical academic integrity problem through a working software solution. The research problem is not limited to theoretical Sinhala sentence similarity. It also requires a usable platform that can accept documents, extract text, compare sentence meaning, detect suspicious content, and produce a clear report for users.

The methodology combines natural language processing, machine learning, web based software development, and evaluation through test data. The project is divided into 2 major technical components. Component 1 focuses on Sinhala to Sinhala similarity plagiarism detection. Component 2 focuses on English to Sinhala semantic plagiarism detection. Both

components share common services such as document upload, preprocessing, dashboard display, result storage, and report generation. This shared design helps the platform work as one system rather than 2 separate prototypes.

The selected methodology reflects the project objective stated in the proposal: to build a plagiarism checker that handles Sinhala texts and translated English to Sinhala plagiarism, works at sentence level, gives clear match scores, and searches sources in real time. The proposed solution uses focused crawling, Sinhala and bilingual preprocessing, sentence embeddings, similarity scoring, threshold tuning, and a report generation pipeline. The Topic Assessment Form also identifies the need for Sinhala linguistic expertise, data engineering, machine learning operations, balanced corpora, bilingual sentence pairs, and labelled plagiarism documents.

3.2 Research Design

The research design follows a design science approach. The project identifies a real problem, reviews existing knowledge, designs a technical artefact, builds the artefact, tests it, and evaluates whether it addresses the problem. This approach is suitable because the expected outcome is a software platform, not only a theoretical model.

The design process involved 5 main stages. The first stage was problem identification. Existing Sinhala plagiarism systems were reviewed, and their limitations were compared with the needs of Sri Lankan academic users. The second stage was requirement analysis. The team identified the need for Sinhala document upload, text extraction, sentence level comparison, cross language semantic matching, source evidence, and downloadable reports. The third stage was data preparation. Sinhala monolingual corpora and English Sinhala parallel datasets were identified and filtered for model training and testing. The fourth stage was model and system development. Each component built its own detection pipeline, then both were integrated into a common Flask based web application. The fifth stage was evaluation. The system was tested using document level workflows, sentence level outputs, threshold based classification, and report readability.

This research design supports both technical and user centred evaluation. A plagiarism detector must produce correct predictions, but it must also present the results in a way that lecturers and students can understand. Therefore, the methodology considers accuracy, precision, recall, usability, response time, source display, and report clarity.

3.3 Overall System Methodology

The Similarity.lk workflow begins when a user uploads a document through the web interface. The system validates the file type and stores the uploaded document in the application folder. It then extracts the text from PDF or Word files. If the PDF contains scanned pages, the system uses OCR fallback to recover text. After extraction, the text is cleaned, segmented into sentences, and passed to the relevant detection branch.

For Sinhala to Sinhala plagiarism, the system processes Sinhala sentences and compares them against Sinhala reference content or internal candidate text. The detection method uses vector based features and a trained model to assign similarity scores. It can identify direct copying and some forms of paraphrased similarity. The output is a sentence level list of suspicious content with confidence scores.

For English to Sinhala semantic plagiarism, the system compares Sinhala sentences against English source candidates. This branch uses bilingual embeddings such as LaBSE to represent English and Sinhala sentences in the same vector space. It calculates cosine similarity to estimate meaning overlap. MarianMT is used as a translation fallback when direct embedding comparison needs support. The final output shows English source sentences, Sinhala submitted sentences, translated explanation where available, and similarity confidence.

The final stage combines the results into a report. The report includes the uploaded document name, total analysed sentences, suspicious matches, plagiarism percentage, confidence values, and sentence level evidence. The dashboard stores past reports and allows users to view or download previous results.

3.4 Data Collection Methodology

Data collection was planned separately for the 2 detection branches.

For Component 1, Sinhala data was required to detect similarity within Sinhala text. The project identified large Sinhala datasets such as SEACrowd cc100, SinhalaCorpusLarge, Sinhala Wikipedia based datasets, NSINA news data, and other web based corpora. These datasets help cover different text types, including news, blogs, encyclopaedic writing, academic style text, and long form prose. A balanced Sinhala dataset is important because earlier Sinhala plagiarism studies used small or narrow corpora. The proposed target was to

gather a large Sinhala sentence collection and label documents as direct copy, paraphrase, or non plagiarised text.

For Component 2, English Sinhala parallel data was required. The selected sources included WikiMatrix, CCAligned, JW300, OpenSubtitles, FLORES, Tatoeba, and OPUS GNOME. These datasets provide aligned sentence pairs across different styles. WikiMatrix gives encyclopaedic content. CCAligned provides large web aligned pairs. JW300 adds formal long sentence structures. OpenSubtitles adds conversational text. FLORES supports final evaluation. Tatoeba supports short clean sentence tests. OPUS GNOME supports short technical strings.

The data collection process considered 4 quality controls. First, language filtering was used to remove records that did not contain valid Sinhala or English. Second, duplicate removal was used to reduce repeated sentence bias. Third, length ratio filtering was used to remove misaligned bilingual pairs. Fourth, domain grouping was used to test whether the model performed across different text styles.

3.5 Data Preprocessing Methodology

Preprocessing is a critical step because raw documents contain formatting noise, punctuation issues, scanned text errors, page numbers, mixed scripts, and broken line breaks. If such noise is passed directly into a model, prediction quality decreases.

The preprocessing pipeline includes text extraction, Unicode cleaning, sentence segmentation, tokenisation, and feature preparation. PDF extraction is handled first. If the document contains selectable text, the system extracts it directly. If the document is scanned, OCR is used. Word documents are parsed by reading their paragraphs and text content.

The system then removes unnecessary characters, repeated spaces, broken line breaks, page headers, and unrelated symbols. Sinhala text is kept in Unicode form. Mixed English terms are preserved when they are part of the sentence. Sentence segmentation then divides the document into sentence level units. This is needed because the system reports plagiarism at sentence level rather than only at document level.

For Component 1, the processed Sinhala sentences are used to generate TF IDF features and similarity-based values. For future model upgrades, the same cleaned sentences can be passed into fastText, multilingual BERT, or Siamese LSTM models.

For Component 2, English and Sinhala sentences are cleaned but not heavily stemmed. Bilingual embedding models need full sentence meaning, so aggressive stop word removal may harm semantic comparison. The cleaned sentences are passed into LaBSE to generate embeddings.

3.6 Component 1 Methodology: Sinhala Similarity Plagiarism Detection

Component 1 detects plagiarism within Sinhala text. Its main purpose is to identify copied, lightly edited, and paraphrased Sinhala sentences. The component follows a sentence level detection approach because plagiarism often appears in selected parts of a document rather than across the entire file.

The process begins with Sinhala document input. The system extracts text from uploaded PDF or Word documents and applies Sinhala preprocessing. The cleaned text is segmented into sentences. Sentences that are too short are filtered out because they do not provide enough evidence for reliable detection.

The current prototype uses TF IDF vectorisation and a Random Forest model. TF IDF converts Sinhala sentences into numerical representations based on term importance. Similarity features are calculated from candidate sentence pairs. These features include cosine similarity, word overlap, sentence length ratio, common token count, and model confidence values. The Random Forest model then classifies sentence pairs as suspicious or non suspicious.

This approach was selected because it is lightweight and practical for a working Flask application. It does not require high GPU resources during deployment. It can also be integrated easily with report generation and dashboard features. The project presentation records that the Sinhala component was developed as a Flask based web application with secure login, document upload, dashboard view, text extraction, OCR fallback, sentence processing, Random Forest model integration, TF IDF vectorizer integration, PDF report generation, document history, and dashboard analytics.

Although the prototype uses a classical machine learning model, the research direction remains open to contextual embeddings. The planned system architecture includes fastText, multilingual BERT, and Siamese LSTM for deeper semantic similarity. This future upgrade would help the system detect paraphrased Sinhala sentences more accurately.

3.7 Component 2 Methodology: English to Sinhala Semantic Plagiarism Detection

Component 2 detects translated plagiarism from English to Sinhala. This branch is required because a Sinhala document may copy the meaning of an English source without sharing any surface words. A normal Sinhala similarity checker cannot detect this case.

The methodology uses bilingual sentence embeddings. LaBSE is used to encode English and Sinhala sentences into a shared semantic vector space. If an English source sentence and a Sinhala submitted sentence express the same meaning, their vectors should appear close to each other. Cosine similarity is used to calculate this closeness.

The process begins with English source sentences and Sinhala submitted sentences. Both are cleaned and segmented. LaBSE generates embeddings for each sentence. The system calculates cosine similarity between English and Sinhala vectors. If the score passes a tuned threshold, the pair is marked as suspicious.

MarianMT is used as a fallback method. In uncertain cases, the Sinhala sentence may be translated into English or the English sentence may be translated into Sinhala. This translated form supports a second level comparison and improves report explanation. The report can show the original English sentence, the Sinhala sentence, and a translated version to help the reviewer understand the match.

The Component 2 presentation identifies LaBSE, LASER, MarianMT, Hugging Face Transformers, Sentence Transformers, Python, TensorFlow, PyTorch, Flask REST API, PostgreSQL, Docker, and Git as relevant technologies. It also states that the component aims to build a parallel Sinhala English sentence set, fine tune or apply LaBSE, and integrate translation and similarity services into a real time endpoint.

3.8 System Architecture

The system architecture follows a modular web application structure. It contains 5 main layers: presentation layer, application layer, preprocessing layer, model layer, and storage layer.

The presentation layer includes the user interface. It supports registration, login, document upload, dashboard view, report access, and document history. The application layer is implemented through Flask routes and service logic. It handles file validation, analysis requests, report generation, and session based access.

The preprocessing layer extracts and cleans text. It supports PDF extraction, Word extraction, OCR fallback, sentence segmentation, and language specific processing. The model layer contains the Sinhala similarity model and the cross language semantic model. Component 1 applies TF IDF and Random Forest in the working prototype, with future support for Siamese LSTM and mBERT. Component 2 applies LaBSE embeddings and MarianMT fallback.

The storage layer maintains uploaded documents, generated reports, analysis logs, match evidence, and dashboard records. PostgreSQL is suitable for structured logs and report metadata. File storage is used for uploaded documents and generated reports. Future versions can use vector databases for large scale semantic search.

This architecture allows each component to be upgraded separately. For example, the Sinhala similarity model can be replaced with a mBERT based model without changing the upload interface. The cross language detector can add a vector index without changing the report format. This modularity is important because NLP models improve over time

.

3.9 Commercialisation Aspects of the Product

Similarity.lk has commercial potential because it addresses a local language problem that is not served well by global plagiarism detection tools. The main target users are universities, schools, lecturers, publishers, research supervisors, training institutes, and Sinhala content platforms. The system can be offered as a web-based plagiarism checking service with institutional accounts and individual user accounts.

The value proposition is specific. It provides Sinhala to Sinhala plagiarism detection and English to Sinhala semantic plagiarism detection. Most commercial tools perform well for English, but Sinhala academic users need a system that understands local scripts, translation-based copying, and Sinhala reporting needs. This gives Similarity.lk a clear position in the Sri Lankan education technology market.

A possible business model includes a free tier and paid institutional plans. The free tier may allow limited checks per month. A student plan may allow higher monthly usage. An institutional plan may allow bulk submission, lecturer dashboards, class level reports, and document archives. A publisher plan may allow long document checking and source link reporting.

The project's proposal presentation also included a budget covering data collection, annotation labour, cloud GPU compute, software and APIs, domain and server hosting, printing, and contingency costs. This shows that the system has a practical cost structure and can be planned as a deployable product rather than only an academic prototype.

3.10 Ethical and Legal Considerations

The system handles academic documents, and such documents may contain personal or institutional information. Therefore, privacy and responsible use are important. Uploaded documents should be stored only for analysis and report history. Users should have the ability to delete uploaded files and reports. The system should avoid exposing submitted student work to other users.

False positives also create ethical risk. A plagiarism detector should support academic judgement, not replace it completely. A high similarity score should be treated as evidence for review, not automatic proof of misconduct. The report should show sentence level evidence so lecturers can decide whether the match is plagiarism, common wording, correct citation, or a false alarm.

The system should also respect copyright and web crawling rules. When web crawling is added, the crawler should avoid restricted content, follow robots.txt where applicable, and store only the minimum text required for matching and evidence. Public datasets must be used according to their licences.

4. TESTING AND IMPLEMENTATION

4.1 Introduction to Testing and Implementation

This chapter explains how Similarity.lk was implemented and tested as a working prototype. The system was not built as a single script. It was implemented as a web based platform with document upload, backend processing, machine learning prediction, semantic comparison, dashboard display, and PDF report generation.

Testing focused on 4 areas. The first area was functional testing, which checked whether system features worked correctly. The second area was model testing, which checked whether similarity and semantic plagiarism predictions were reasonable. The third area was integration testing, which checked whether both components worked inside the same platform. The fourth area was usability testing, which checked whether users could understand the result output.

4.2 Implementation Environment

The implementation environment used Python as the main programming language. Flask was used for web application development because it is lightweight and works well with Python based NLP pipelines. The system was structured with separate folders for models, utilities, templates, static resources, uploads, reports, and backend services. This made the project easier to maintain and extend.

The main machine learning and NLP libraries included scikit learn for TF IDF vectorisation and Random Forest classification, Hugging Face Transformers and Sentence Transformers for embedding models, PyMuPDF for PDF extraction, Tesseract for OCR fallback, and PDF generation libraries for report output. PostgreSQL was planned for structured storage of embeddings, logs, and report metadata. Docker was considered for controlled deployment.

The user interface was implemented using HTML templates, CSS, and backend Flask routes. Users can register, log in, upload documents, run analysis, view results, download reports, and access document history. The implementation prioritised a simple interface because the target users include lecturers and students who may not have technical knowledge of NLP models.

4.3 Implementation of Document Upload and Text Extraction

The document upload module allows users to submit Sinhala documents for checking. The system validates file extensions to prevent unsupported file types. Accepted formats include PDF and Word documents. After validation, the document is saved in the upload directory with a controlled file path.

Text extraction then begins. For PDF files, PyMuPDF extracts text from pages. If the PDF contains selectable Unicode text, extraction is direct. If the PDF is scanned, OCR fallback is used. OCR is important because many Sinhala documents are distributed as scanned documents, especially older academic notes and assignments. Tesseract can recover text from scanned images, although OCR quality depends on document clarity and font quality.

For Word documents, the system reads paragraphs and extracts the document content. The text is then passed to the preprocessing service. Extraction testing checked whether different file formats produced usable text and whether the system handled empty or unreadable files gracefully.

4.4 Implementation of Sinhala Similarity Detection

The Sinhala similarity detection module follows the pipeline developed under Component 1. After text extraction, the system cleans the Sinhala text and segments it into sentences. Each sentence is prepared for vectorisation and comparison.

The working implementation integrates a saved TF IDF vectorizer and a trained Random Forest model. The vectorizer converts Sinhala text into feature representations. The model

uses these features with similarity values to classify suspicious content. The output includes sentence level confidence and a document level plagiarism percentage.

The backend service returns structured output to the result page and report generator. This output includes total sentences, suspicious sentences, confidence scores, and highlighted content. The project presentation confirms that this module included secure login, upload, dashboard, report download, model integration, OCR fallback, sentence comparison, PDF reports, history, and dashboard analytics.

4.5 Implementation of English to Sinhala Semantic Detection

The semantic detection module follows the Component 2 pipeline. The system accepts Sinhala submitted text and compares it with English source candidates. The core technique is bilingual sentence embedding.

LaBSE encodes English and Sinhala sentences into dense vectors. Cosine similarity is calculated between vectors. Scores above the threshold are treated as translated plagiarism candidates. MarianMT is used as a fallback to support uncertain matches and improve report explanation.

The module was designed as a service that can be connected to the main Flask application. It can return source sentence, Sinhala sentence, similarity score, threshold decision, and translated explanation. The presentation records that Component 2 expanded the platform into a unified system, created reusable preprocessing for English and Sinhala document handling, prepared modular backend folders, built result management workflow, and standardised the reporting format.

4.6 Implementation of Report Generation

Report generation is a key part of Similarity.lk because users need evidence, not only a percentage. The report generator creates a PDF report after each analysis. It includes the document name, date, number of analysed sentences, suspicious sentence count, plagiarism percentage, confidence scores, and matched evidence.

For Sinhala similarity detection, the report highlights suspicious Sinhala sentences and their similarity confidence. For cross language detection, the report can show the English source sentence, the Sinhala submitted sentence, the similarity score, and a translated explanation if available.

The report design helps lecturers review the system output fairly. It avoids presenting the plagiarism percentage as a final judgement. Instead, it gives evidence for each suspicious section. This is important because plagiarism detection tools can produce false positives when common phrases or standard definitions appear across documents.

4.7 Functional Testing

Functional testing checked whether system features worked as expected. The key functional test cases are shown below.

Table 1: Functional test cases for Similarity.lk

Test ID	Test area	Expected outcome
FT1	User registration	New user account is created
FT2	User login	Valid user reaches dashboard
FT3	Invalid login	System blocks access
FT4	File upload	Valid PDF or Word file is accepted
FT5	Invalid file upload	Unsupported file is rejected
FT6	PDF extraction	Text is extracted from PDF
FT7	OCR fallback	Scanned PDF is processed where possible
FT8	Sinhala analysis	Suspicious Sinhala sentences are detected
FT9	Cross language analysis	English Sinhala matches are scored
FT10	Report generation	PDF report is created
FT11	History display	Previous records appear on dashboard
FT12	File deletion	User can remove uploaded file or record

The functional tests showed that the system works as an end-to-end prototype. The main workflow from upload to report generation was completed.

4.8 Model Testing

Model testing checked prediction behaviour. For Component 1, Sinhala test documents were used to check whether copied and lightly edited Sinhala sentences were marked as suspicious. The system calculated similarity features and classified sentence pairs. The test output was reviewed through sentence level results and report content.

For Component 2, English Sinhala sentence pairs were tested. Direct translations were expected to receive high similarity scores. Unrelated sentence pairs were expected to receive low scores. Paraphrased translations were expected to appear near or above the threshold depending on meaning overlap. Borderline cases were useful for threshold tuning.

The evaluation metrics planned for model testing include accuracy, precision, recall, F1 score, confusion matrix, ROC curve, and average response time. These metrics are suitable because plagiarism detection must balance missed plagiarism and false accusations.

4.9 Integration Testing

Integration testing checked whether both components worked inside the same application. This included shared login, shared upload workflow, shared dashboard, shared report generation, and consistent display of confidence scores.

The main integration challenge was standardising output from 2 different model types. Component 1 produces Sinhala similarity results. Component 2 produces cross language semantic similarity results. The system needed a common result structure so both outputs could be shown in the same dashboard and report style. This was handled by standardising fields such as sentence ID, matched sentence, source type, similarity score, decision label, and confidence value.

4.10 Usability Testing

Usability testing focused on whether academic users could understand the output. A plagiarism detection system should not require users to understand embeddings, vector spaces, or TF IDF. The interface must show clear document status, overall percentage, sentence level evidence, and report download options.

The dashboard was designed to show the analysis status and past records. The result page was designed to show the plagiarism score and suspicious sentences. The PDF report was designed to support offline review. Future usability testing should involve lecturers and students who can provide feedback on report wording, colour coding, and evidence layout.

5. RESULTS AND DISCUSSION

5.1 Introduction to Results and Discussion

This chapter presents the expected and observed results of Similarity.lk based on the developed prototype, project documents, and evaluation direction. The results are discussed across 5 areas: dataset preparation, Sinhala similarity detection, English to Sinhala semantic detection, integrated system behaviour, and contribution to the research gap.

The current project is a final year research prototype. Therefore, results should be interpreted as evidence of feasibility and system development, not as a final commercial benchmark. The prototype proves that Sinhala document processing, sentence level analysis, machine learning based similarity scoring, bilingual semantic comparison, dashboard reporting, and downloadable evidence can be combined into one platform.

5.2 Dataset Preparation Results

The dataset review showed that Sinhala and English Sinhala resources are available, but they require filtering before use. For Sinhala similarity detection, large corpora such as CC100 Sinhala and SinhalaCorpusLarge provide useful coverage, but they contain web noise. Cleaner sources such as Sinhala Wikipedia support threshold tuning and controlled evaluation. News datasets provide domain specific style and metadata.

For semantic detection, parallel datasets provide strong starting points. WikiMatrix and CCAligned provide large scale bilingual data. JW300 provides formal long sentence structures. OpenSubtitles provides conversational text. FLORES and Tatoeba support cleaner evaluation. OPUS GNOME provides short technical strings.

The result of this review is that Similarity.lk can use a mixed dataset strategy. Large noisy corpora are useful for coverage, while cleaner datasets are useful for testing. This directly addresses a weakness of earlier Sinhala plagiarism studies, which often depended on small or narrow datasets.

5.3 Sinhala Similarity Detection Results

The Sinhala similarity component produced a working application workflow. Users can upload Sinhala documents, extract text, process sentences, run the model, and generate reports. The practical implementation achieved more than a model experiment. It delivered a user facing Sinhala plagiarism detection module.

The output includes sentence level suspicious content, confidence values, and an overall plagiarism percentage. This is important because Sinhala plagiarism checking should not rely only on a single score. Sentence level evidence allows lecturers to inspect each flagged part and decide whether it is direct copying, paraphrasing, common phrasing, or properly cited content.

Compared with earlier Sinhala systems, the component offers a broader workflow. KasthuriArachchi and Charles showed that Word2Vec sentence similarity can detect Sinhala plagiarism with strong reported accuracy, but their study was limited by dataset size. Rajamanthri and Thelijjagoda added web-based Sinhala plagiarism checking and reported 88 percent accuracy, but the method relied on token-based similarity and struggled with deep paraphrase. Nilaxan and Ranathunga showed the value of Siamese LSTM for Sinhala sentence similarity. Similarity.lk builds on these directions by combining detection, upload, preprocessing, sentence level scoring, and report generation inside one prototype.

The final report should include the following table after final testing values are inserted.

Table 2: Sinhala similarity detection evaluation summary

Metric	**Result**
Accuracy	
Precision	
Recall	
F1 score	
Average processing time	

5.4 English to Sinhala Semantic Detection Results

The English to Sinhala semantic component produced a workflow for cross language plagiarism detection. The main result is that the system design can compare English and

Sinhala sentences through bilingual embeddings. This addresses a major gap in existing Sinhala plagiarism tools, which do not handle translated plagiarism between English and Sinhala.

The system can show English source text, Sinhala submitted text, cosine similarity score, and optional translation fallback output. This result is important because translated plagiarism is difficult to explain through a score alone. A lecturer needs to see why the Sinhala sentence was linked to the English source.

The expected evaluation uses direct translation pairs, paraphrased translation pairs, unrelated pairs, and partially related pairs. Direct translations should score high. Unrelated pairs should score low. Paraphrased translations may require threshold tuning and translation fallback. Topic related but non plagiarised pairs are the most difficult because they may share broad meaning without copying. These cases are useful for reducing false positives.

The final report should include a confusion matrix and ROC curve for the semantic detection model. It should also include a similarity score distribution graph showing the separation between translated plagiarism and non-plagiarism pairs.

Table 3: English to Sinhala semantic detection evaluation summary

Metric	**Result**
Accuracy	
Precision	
Recall	
F1 score	
AUC score	
Average API response time	

5.5 Integrated System Results

The integrated system result is the main achievement of the project. Similarity.lk combines 2 different plagiarism detection branches in one platform. It supports Sinhala to Sinhala similarity checking and English to Sinhala semantic checking. The shared application shell

includes authentication, upload, extraction, preprocessing, analysis, dashboard, history, and report generation.

The integrated workflow improves usability. A user does not need to run separate scripts for each component. The platform can guide the user from upload to final report. The standard report format allows both components to present matched sentences, confidence values, and evidence in a consistent way.

The integration also supports future expansion. A vector database can be added for large scale source search. A crawler can be added for live web retrieval. The Sinhala branch can upgrade from TF IDF and Random Forest to mBERT or Siamese LSTM. The semantic branch can test LaBSE, LASER, and translation-based hybrid methods. The report generator can add source URLs and paragraph level evidence.

5.6 Research Findings

The project produced 5 main findings.

First, Sinhala plagiarism detection needs sentence level processing. Document level similarity alone is too broad because plagiarism may occur in selected sections. Sentence level output gives more precise evidence.

Second, preprocessing strongly affects model performance. Sinhala documents may contain scanned pages, broken Unicode, mixed punctuation, and formatting noise. Proper extraction, OCR fallback, cleaning, and segmentation are needed before model comparison.

Third, monolingual Sinhala plagiarism and translated English Sinhala plagiarism require different techniques. Sinhala to Sinhala similarity can use TF IDF, lexical overlap, embeddings, or Siamese models. English to Sinhala plagiarism requires bilingual embeddings or translation support because surface words differ across languages.

Fourth, report clarity is as important as model output. A plagiarism detector should not only display a percentage. It should show matched lines, source evidence, confidence values, and sentence level highlights.

Fifth, existing Sinhala research provides useful foundations but not a complete platform. Earlier studies showed the value of Word2Vec, web-based comparison, and Siamese sentence similarity. Similarity.lk extends these ideas into an integrated web-based prototype.

5.7 Discussion

The results show that Similarity.lk addresses a real gap in Sinhala academic integrity support. Most plagiarism detection systems are designed for English. Sinhala users need tools that understand Sinhala script, Sinhala document formats, Sinhala sentence behaviour, and English to Sinhala translated plagiarism.

The project also shows that modern NLP can improve plagiarism detection beyond direct matching. Embeddings help capture meaning similarity. Bilingual embeddings make cross language matching possible. Machine translation fallback can improve interpretability. A report generator can turn model results into usable academic evidence.

There are still limitations. The dataset must be expanded and carefully labelled. The current Sinhala prototype uses a lightweight model, which may be less effective for deep paraphrase. OCR errors may affect extracted text. Cross language detection may produce false positives when 2 sentences discuss the same general topic without plagiarism. Large scale source search is still needed for full real world use.

Despite these limits, the prototype is valuable because it proves the architecture and workflow. It shows that Sinhala plagiarism detection can be developed as a complete system, not only as an isolated algorithm.

5.8 Summary of Each Student's Contribution

R.R.M.I.M. Ranaweera contributed to Component 1, Sinhala Similarity Plagiarism Detection. The work included Sinhala corpus planning, preprocessing design, model integration, Flask application features, secure login, document upload, text extraction, OCR fallback, sentence comparison, confidence scoring, dashboard analytics, history management, and PDF report generation. This contribution built the Sinhala similarity detection foundation of the platform.

Abhayasiri S.A.U.D contributed to Component 2, English to Sinhala Semantic Plagiarism Detection. The work included bilingual dataset planning, English Sinhala preprocessing, LaBSE and MarianMT workflow design, cross language similarity scoring, threshold tuning, modular backend preparation, report standardisation, dashboard workflow, and integration support. This contribution expanded Similarity.lk into a platform that can detect translated plagiarism between English and Sinhala.

Together, both contributions produced an integrated Sinhala academic integrity platform with monolingual and cross language detection support.

6. CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

This research designed and developed Similarity.lk, a Sinhala focused plagiarism detection platform using modern embedding and similarity methods. The project was motivated by a clear problem: existing plagiarism tools mainly support English and do not adequately handle Sinhala text or English to Sinhala translated plagiarism. Prior Sinhala research made valuable progress, but it remained limited by small datasets, surface level matching, narrow evaluation, or lack of full system integration.

Similarity.lk addresses this gap through a 2 component architecture. Component 1 detects Sinhala to Sinhala plagiarism by extracting Sinhala document text, preprocessing it, segmenting it into sentences, calculating similarity features, applying a trained model, and generating evidence based reports. Component 2 detects English to Sinhala semantic plagiarism through bilingual sentence embeddings, cosine similarity, threshold tuning, and translation fallback. Both components are integrated into one Flask based platform with upload, analysis, dashboard, history, and report generation.

The project demonstrates that Sinhala plagiarism detection can move beyond manual reading and simple word matching. It can use NLP pipelines, sentence level scoring, machine learning, bilingual embeddings, and structured reporting to support academic decision making. The system does not replace human judgement. Instead, it gives lecturers and reviewers stronger evidence for identifying suspicious text.

6.2 Achievement of Objectives

The first objective was to build an automated plagiarism checker for Sinhala and English Sinhala plagiarism cases. This objective was achieved through the Similarity.lk prototype, which supports both Sinhala similarity detection and English to Sinhala semantic detection.

The second objective was to detect Sinhala to Sinhala copied and paraphrased text. This was achieved through Component 1, which includes document upload, extraction, Sinhala preprocessing, TF IDF vectorisation, Random Forest prediction, sentence level confidence scoring, and PDF report generation.

The third objective was to detect English to Sinhala translated plagiarism. This was addressed through Component 2, which uses LaBSE bilingual embeddings, cosine similarity, MarianMT fallback, and a standardised result workflow.

The fourth objective was to provide clear reports. This was achieved through downloadable PDF reports containing plagiarism percentage, suspicious sentences, confidence values, and match evidence.

The fifth objective was to create a scalable architecture. This was achieved through modular backend design, separated model services, reusable preprocessing, and shared reporting structures.

6.3 Main Contributions

The project makes 3 main contributions.

The first contribution is a Sinhala focused plagiarism detection workflow. It handles Sinhala document upload, OCR fallback, preprocessing, sentence level scoring, and report generation.

The second contribution is a cross language semantic plagiarism workflow for English to Sinhala cases. This is important because machine translation can hide copied English sources from normal Sinhala plagiarism checkers.

The third contribution is an integrated platform design. Similarity.lk combines both detection paths into one user-oriented system. This makes the project more practical than a standalone notebook or isolated model.

6.4 Limitations

The project has several limitations. The first limitation is dataset size and quality. Sinhala and English Sinhala corpora are available, but labelled plagiarism data remains limited. Larger manually verified datasets are needed for stronger evaluation.

The second limitation is source coverage. A full plagiarism checker needs wide web crawling and institutional source databases. The current prototype focuses on the detection workflow and prepared datasets.

The third limitation is OCR quality. Sinhala OCR can produce errors when documents are scanned, blurred, or written using unusual fonts. These errors can reduce similarity accuracy.

The fourth limitation is deep paraphrase. A user may rewrite an idea heavily while keeping the same meaning. Classical TF IDF features may miss such cases. Contextual embeddings and paragraph level comparison can improve this in future work.

The fifth limitation is false positives in semantic matching. English and Sinhala sentences may discuss the same topic without plagiarism. Human review remains necessary.

6.5 Recommendations

Future work should expand the Sinhala plagiarism dataset with academic essays, news, blogs, Wikipedia content, and controlled paraphrase samples. The dataset should use at least 2 annotators to reduce bias.

The Sinhala similarity model should be upgraded with fastText, multilingual BERT, or Siamese LSTM. These models can capture deeper semantic relationships than surface level features.

The English to Sinhala branch should test LaBSE, LASER, and translation based hybrid methods across multiple domains. Paragraph level semantic matching should be added to detect cases where sentence boundaries change during translation.

A web crawler should be integrated to search Sinhala and English sources in real time. The crawler should filter language, remove duplicates, respect source access rules, and store source links for report evidence.

The report should be improved with visual highlights, source URLs, confidence explanations, and reviewer notes. The system should also support institution level accounts, class level submissions, and bulk upload.

6.6 Final Summary

Similarity.lk provides a practical response to the lack of Sinhala focused plagiarism detection tools. It combines Sinhala similarity detection and English to Sinhala semantic detection inside one web-based platform. The system supports document upload, text extraction, preprocessing, similarity scoring, semantic comparison, dashboard history, and PDF reporting. The project shows that modern NLP methods can support academic integrity in low resource language settings. With larger datasets, stronger contextual models, source crawling, and institutional deployment, Similarity.lk can become a valuable plagiarism detection tool for Sinhala academic and professional writing.

REFERENCES

[1] T. KasthuriArachchi and E. Y. A. Charles, “Deep Learning Approach to Detect Plagiarism in Sinhala Text,” 2019 IEEE 14th International Conference on Industrial and Information Systems, Peradeniya, Sri Lanka, 2019, doi: 10.1109/ICIIS47346.2019.9063299.

[2] S. Nilaxan and S. Ranathunga, “Monolingual Sentence Similarity Measurement Using Siamese Neural Networks for Sinhala and Tamil Languages,” 2021 Moratuwa Engineering Research Conference, pp. 567 to 572, 2021, doi: 10.1109/MERCon52712.2021.9525786.

[3] L. Rajamanthri and S. Thelijjagoda, “Plagiarism Detection Tool for Sinhala Language with Internet Resources Using Natural Language Processing,” 2021 10th International

Conference on Information and Automation for Sustainability, pp. 156 to 160, 2021, doi: 10.1109/ICIAfS52090.2021.9605852.

[4] F. Feng et al., “Language agnostic BERT Sentence Embedding,” arXiv preprint arXiv:2007.01852, 2020.

[5] M. Artetxe and H. Schwenk, “Massively Multilingual Sentence Embeddings for Zero Shot Cross Lingual Transfer and Beyond,” Transactions of the Association for Computational Linguistics, vol. 7, pp. 597 to 610, 2019.

[6] J. Tiedemann and S. Thottingal, “OPUS MT Building Open Translation Services for the World,” Proceedings of the 22nd Annual Conference of the European Association for Machine Translation, pp. 479 to 480, 2020.

[7] G. Schwenk, V. Chaudhary, S. Sun, H. Gong, and F. Guzmán, “WikiMatrix: Mining 135M Parallel Sentences in 1620 Language Pairs from Wikipedia,” Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, pp. 1351 to 1361, 2021.

[8] H. Schwenk et al., “CCMatrix: Mining Billions of High Quality Parallel Sentences on the Web,” Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 6490 to 6500, 2021.