Improving and Measuring OCR Accuracy for Sinhala with Tesseract OCR Engine

A dissertation submitted for the Degree of Master of Computer Science

B. P. K. M. Balasooriya
University of Colombo School of Computing
2020

DECLARATION

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: Kasun Manoj Balasooriya

Registration Number: 2016/MCS/013

Index Number: 16440132

Signature: Date: 2020/11/17

This is to certify that this thesis is based on the work of

Mr. Kasun Manoj Balasooriya.

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name: V.W Welgama

Signature: Date: 2020/11/17

ABSTRACT

This research project proposes and implements a system to improve and measure the accuracy of the Sinhala OCR using the Tesseract OCR engine. The system implements modules to rectify the issues which are inherent to the Tesseract OCR engine when performing OCR for Sinhala language. During the course of the project, the world level accuracy was used to measure the accuracy of the output from the system.

As a baseline to compare the results of the proposed system which implements tesseract OCR, the software the OCR Engine "පෙළ කැටපත" was used. To improve the accuracy, a syntactical rule engine a module to detect and correct confusion character pairs and a rudimentary dictionary look up feature to detect and correct errors in word level has been implemented into the system.

During the initial stage in the project which implemented only the Tesseract OCR library functionality, the output was less accurate when compared with the OCR Engine "පෙළ කැටපත". But as the features were built into the system, it yielded significantly improved results which improved the word level accuracy from the original 53.22% to 86.16%.

ACKNOWLEDGEMENTS

I wish to express my gratitude to the supervisor of this project, Mr. Viraj Welgama, Senior Lecturer, University of Colombo School of Computing, for his valuable guidance and advices given throughout the project. I would also like to acknowledge all the lecturers of the MSC program, the course coordinators for their guidance and assistance. I would also like to acknowledge my teammates and managers from work who helped me out to balance out my work life with studies. I also like to extend my sincere gratitude to the classmates of the MSC program for their support in numerous ways throughout the course and the project. I would like to thank all the academic and non-academic staff of UCSC for their support in various ways in past years. Finally, I wish to pay my gratitude to my family members, relations and friends for their understanding and support given during my study period.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF GRAPHS

LIST OF APPENDICES

LIST OF ACRONYMS

OCR – Optical Character Recognition

COTS -Commercial Off The Shelf

UCSC – University of Colombo School of Computing

CHAPTER 1: INTRODUCTION

OCR Stands for "Optical Character Recognition." OCR is a technology that recognizes text within a digital image [3]. It is commonly used to recognize text in scanned documents, but it serves many other purposes as well [3]. OCR software processes a digital image by locating and recognizing characters, such as letters, numbers, and symbols [26]. Some OCR software will simply export the text, while other programs can convert the characters to editable text directly in the image [26]. Advanced OCR software can export the size and formatting of the text as well as the layout of the text found on a page [3].

The recognition process of the characters would be not accurate enough due to various reasons. Some reasons would be when the original image quality is poor and skewed. By rectifying the above issues using preprocessing techniques before feeding the image into the OCR engine. This would enable us to improve the result, and the validation processes could be applied to the output with much ease.

Post-processing is used to ensure the accuracy of the OCR output sequence to be as the same as that of the input source. If a particular word differs from the original source, a replacement suggestion is made to form a sensible and meaningful output.

Even after repeated attempts of producing accurate OCR output, although there are developments in strategies behind OCR, there are still problems when it comes to recognizing the correct character. However, the OCR knowledgebase has been widely utilized in increasing the accuracy of OCR recognition. Some of the techniques used to correct OCR errors are the usage of language models, Statistical information of N-grams, Grammar rules, Syntactic Analysis, Language Models and Lexicons.

The accuracy of the results from an optical character recognition engine may vary from the context and the language it’s being used. Tesseract is an OCR engine which is used to do OCR for many languages [1].

1.1 Motivation

Compared to the Latin script and other scripting languages like Chinese and Korean, Sinhala OCR is at a research-level for being able to use at a commercial level. However, the usages of OCR in Sinhala would be many as there are a lot of untapped resources which could be digitized to produce advancements in the respective fields.

Documents and scripts on indigenous medicine, archived historical documents which are decaying with time, voter registers, and government documents which have been stored through various departments like census and hospitals are some of the examples of applied usages which could benefit from an OCR solution.

Since the localization and introduction of computers and digitization to government offices in the recent past has amplified the necessity further. While OCR for Sinhala language and meaningful data extraction is far behind, several improvements have been achieved with respect to the research done using the tesseract OCR engine.

Since Tesseract OCR engine is being used by similar scripting languages like Devanagari, Gurmukhi, Sindhi, Tamil, Thai and Telugu which shares some commonalities, it logical to look into means of utilizing Tesseract OCR with the Sinhala language.

1.2 Aims and Objectives

While there have been attempts to use the Tesseract engine for Sinhala language, it is important to look into ways of improving the output [1]. Measurement is needed to decide on the progress made on such an attempt.

Techniques used to improve OCR accuracy can be classified into two main classes.

1) Preprocessing techniques

Preprocessing techniques are applied to enhance the quality of the source image before running an OCR. A better source image which has minimal skew, a minimum amount of noise (blots/stains) and clear input text are some of the example techniques that can be leveraged.

2) Post-processing techniques

The post-processing techniques are used to fine-tune the output. Most common techniques include running the output against a lexicon, running the output against a rule engine to process the grammar etc.

This project is done with respect to the following project objectives.

The intention of this project is to improve the accuracy by using the above means to measure the accuracy against the accuracy measures of interest [2] mentioned below.

a) Character Accuracy

b) Word level accuracy

c) Accuracy by character class

d) Phrase accuracy

e) Non-stop word accuracy

• Developing an OCR solution for the Sinhala language by focusing on improving the accuracy of selected accuracy measures.

• Meaningful accuracy statistics extraction and Analysis

• Presenting results of the above OCR accuracy measures for the Sinhala language

It is intended to measure the OCR accuracy for Sinhala language using the above accuracy measures. While it is important to measure the OCR accuracy, it is important to evaluate best matrices that can be used to improve OCR accuracy for the Sinhala language.

1.3 Scope of the study

• The measuring will be done for the popular Sinhala Unicode font "Iskolapotha" and depending on the accuracy and the quality of the results, other fonts will be considered. (Newspaper fonts/type writer fonts etc.)

• Preprocessing techniques will be applied manually to create a better input image where applicable. (Will not be built into the system as a feature)

CHAPTER 2: LITERATURE REVIEW

The Tech Terms Computer Dictionary defines Optical character recognition (OCR) as follows. “Optical character recognition (OCR) is a technology that recognizes text within a digital image [3]. The common usage of Optical character recognition (OCR) is to digitally process scanned documents such as passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any suitable documentation [4].

Optical character recognition (OCR) software processes a digital image by locating and recognizing characters, such as letters, numbers, and symbols [3]. Some Optical character recognition (OCR) software will simply export the text, while other programs can convert the characters to editable text directly in the image [3].

Character recognition can be classified into two based on the input method [4]. They are on-line character recognition and Off-line character recognition. On-line character recognition is a real-time process which concentrates on capturing the motion of the characters/glyphs drawn rather than the shape of the character or glyph. Off-line character recognition focuses on scanning and analyzing the shapes of the characters and glyphs [3] [4].

Optical character recognition (OCR) fits under off-line character recognition in the above classification. Usually, the OCR system uses an optical input device (e.g., scanner) to capture images and to feed it to the recognition system. OCR systems can be further classified into two types, OCR systems to recognize printed text and OCR systems to recognize hand-written text [5].

Because of the existence of a variety of writing styles, it's often difficult to produce accurate and reliable output for hand-written text when compared with printed text. The printed text follows a font standard which can be processed relatively easily when compared with a hand-written text [5]. The widespread usage of OCR in the present spans from storing data in databases, processing text for translation, transliteration or converting text to speech, meaningful historical data archiving, digitization of historical documents to Automatic number plate recognition etc.

Most commonly used input formats in OCR software include JPG, TIFF, GIF, and PDF while the output formats are text, Microsoft Word, RTF, PDF etc. Some of the most widely used OCR software are Abbyy FineReader, Adobe Acrobat Professional and Google Tesseract OCR [6]. Some of the above software has leveraged commercial off-the-shelf (COTS) OCR software packages such as Tesseract making OCR software openly available [6] [7].

Commercially developed OCR systems demonstrate a high level of accuracy for Latin script [5]. The above OCR systems are the first OCR systems to be developed for commercial use. The accuracy of commercial OCR systems spans from71% - 98% [5]. The accuracy of the OCR depends on the quality of the scanned image (sharpness/skew, etc.) and the OCR software [5]. Some typical sample errors [5] are listed in table 2.1 below.

Table 2.1: OCR Error Examples

Error class	recognized word	correct word
Segmentation (missing space)	thisis	this is
Segmentation (split word)	depa rtme nt	department
Hyphenation error	de- partment	department
Character misrecognition	souiid	sound
Number substitution	0pporunity	Opportunity
Special char insertion	electi'on	election
Changed word meaning	mad	sad
Case sensitive	BrItaIn	Britain
Punctuation	this.is	this is
Destruction	NI.I II I	Minister
Currencies	?20	$20

Most OCR errors are primarily caused by noise (either inherent or introduced during the scanning process) in the document [5]. A two-pass approach to recognize characters is used in software like Cuneiform and Tesseract. They use the first pass to identify the letter shapes with a confidence level. Then the letters identified with high confidence is used in the second pass to recognize the remaining letters on the second pass. This approach can be beneficial for unusual fonts or low-quality scans or when the font is distorted (e.g. blurred or faded) [4].

Application of OCR in the modern world is diverse and the images to be scanned contain degraded images, heavy-noise, paper skew, low-resolution complex and various fonts /symbols/glossary words etc. Consequently, better OCR accuracy with better reliability has become an inevitability.

An OCR engine will typically mark some of the characters "suspicious", and the error correction is mostly based on verifying the above-identified characters [5]. However, there are other approaches like checking the OCR output against a dictionary [8], probabilistic approaches [9]

[10], advanced level of linguistic knowledge about grammar rules/syntax and semantics [11] [12] [13] which can be applied to improve OCR accuracy.

The current process of OCR can be studied under three main stages; Pre-processing, Recognition and Post Processing. Measures to improve accuracy can be employed in each of these stages. Pre-processing is used to prepare the source in the optimal quality possible before feeding into the OCR software. Then during the recognition stage, the source image is converted into a document with a digital representation of characters. During the post-processing stage error detection and correction of errors to improve accuracy is done. It is important to have an understanding of the above stages and discuss them in detail.

2.1 Pre-processing

There are four steps in preprocessing. Namely Image acquisition, Transformation, Segmentation and Feature extraction [14].

2.1.1 Image Acquisition

Converting a document to a numerical representation is image acquisition. It acquires the image of a document in color, grey-levels, and in binary format. The image is scanned first. The resolution depends on the purpose of the application and the nature of the material. Then the scanned image is sampled and quantified into a number of grey levels. Coding techniques are used to reduce the size of data representing.

2.1.2 Transformation

Transformation of the image to image is portrayed by input-output relationship. It involves refining the data in the representation image in several methods such as Geometrical transformation, Filtering, Background Separation, Object Boundary Detection, and Structural Representation [3].

2.1.3 Segmentation

In the segmentation stage, the layout information is extracted by breaking down the image into lines and further into characters [15]. The number of lines and the number of words/characters can be extracted as Metadata which could be used to improve accuracy during this stage. Tesseract supports and can be compiled to support a variety of page segmentation modes depending on the user preference [16].

```
0    Orientation and script detection (OSD) only.
1    Automatic page segmentation with OSD.
2    Automatic page segmentation, but no OSD, or OCR.
3    Fully automatic page segmentation, but no OSD. (Default)
4    Assume a single column of text of variable sizes.
5    Assume a single uniform block of vertically aligned text.
6    Assume a single uniform block of text.
7    Treat the image as a single text line.
8    Treat the image as a single word.
9    Treat the image as a single word in a circle.
10   Treat the image as a single character.
11   Sparse text. Find as much text as possible in no particular order.
12   Sparse text with OSD.
13   Raw line. Treat the image as a single text line,
                      bypassing hacks that are Tesseract-specific.
```

Figure 2.1: list of supported page segmentation modes in Tesseract

2.1.4 Feature extraction

Feature extraction will classify symbols into classes. Feature extraction captures the distinctive characteristics of the digitized characters for recognition [3].

2.2 Recognition

Recognition involves sensing, feature selection and Creation, pattern recognition, decision making, and system performance evaluation [3]. A vertical projection is used, and it scans a line from top to bottom in character separation [15].

2.2.1 Feature selection and Creation

Feature selection is applied to reduce sample complexity, computational cost, and to overcome performance issues during recognition. There are three approaches to feature extraction and Creation [17]. Filter approach; used to filter out some features before applying a classifier, Wrapper approach which wraps the feature selection algorithm with computational cost and an unbiased classifier, Hybrid model which fits the subset of features and the accuracy of matching to a classifier [17].

2.2.2 Pattern Recognition

Pattern recognition will assign a given pattern into one of the known classes. There are two commonly used methods; template matching and classification on feature space [14] [17] [18].

Template matching compares the pattern with stored models of known patterns and selects the best match [18]. Template matching can be applied when the number of classes and variability within a class is small [14]

When classifying based on the feature space features are summarized and classified using statistics, syntax, neural networks or a combination of above methods [17].

2.3 Post Processing

The human eye with the aid of the human brain is able to read and process most of the texts irrespective of the font, style, skew, distortion missing characters etc. But in contrast, the OCR systems like most machines mimicking human behavior exhibit poor accuracy when compared to that of humans. Hence, improving the accuracy of OCR output has become imperative. Hence to improve the accuracy of the OCR output, post-processing is exploited.

Most errors in recognizing characters are introduced in segmentation and classification stages, mainly due to low-quality images [5]. Post-processing is harnessed to correct errors and/or resolve ambiguities in OCR results by using at the levels of context, word, sematic and sentence.

One such post-processing techniques are character level contextual post-processing [19]. Character level contextual post-processing is mainly based on lexicon methods and statistical methods [19].

2.3.1 Lexicon based post-processing

In lexicon-based post-processing approach, a lexicon is applied to individual characters which are reliably segmented in a word [20]. There are three approaches used in lexical based post-processing approach [21].

1) Bottom-up approach
2) Top-down approach
3) Hybrid approach

2.3.2 Statistical based post-processing

In the statistical method, letter n-grams are used to filter out unacceptable candidate words from the recognizer. An n-gram is a letter string of size n [12]. The probability of n-gram appears in a word is considered for each candidate word for the selection. In this case, conditional probabilities in forwarding and backward directions are considered. Widely used n-grams are bi-grams and trigrams.

While there are post-processing techniques which operates at a character level, another type of popular post-processing technique is to operate at the word level. The dictionary lookup method is the most commonly used post-processing technique which operates at a word level [12] [6].

2.3.3 Context-based post-processing

Context-based post-processing is another post-processing technique. One such context-based post-processing is the usage of syntactic properties of a language like grammar rules to check for illegal character combinations [12]. Looking for a presence of two consecutive vowels or a word string with a forbidden consonant or vowel can be given as an example for such grammar rules.

CHAPTER 3: PROBLEM ANALYSIS AND METHODOLOGY

3.1 Research Problem

3.1.1 Sinhala Language

The Sinhala alphabet consists of 18 vowels 41 consonants and two semi-consonants, which will total into 61 letters [22] as shown in Figure 3.1.

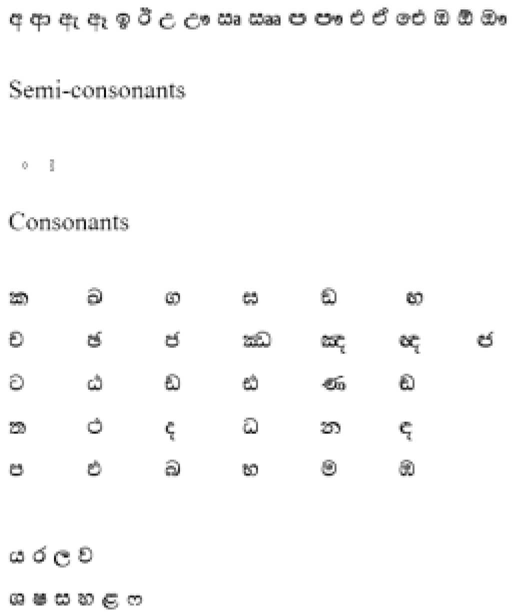


Figure 3.1 Sinhala Alphabet

The usage of semi consonants is to enable writing vocal strokes with speech sounds. A strong relation is present between the speech sound and the consonant when compare to the English language [23].

Table 3.1 Different Consonant modifier combinations with the consonant ක.

consonant	vowel	composite	Vowel form	sequence
ක	ආ	කා	ා	ක ා
ක	ඇ	කැ	ැ	ක ැ
ක	ඈ	කෑ	ෑ	ක ෑ
ක	ඉ	කි	ි	ක ි
ක	ඊ	කී	ී	ක ී
ක	උ	කු	ු	ක ු
ක	ඌ	කූ	ූ	ක ූ
ක	ඍ	කෘ	ෘ	ක ෘ
ක	ඎ	කෲ	ෲ	ක ෘ ෘ
ක	එ	කෙ	ෙ	ෙ ක
ක	ඒ	කේ	ේ	ෙ ක ්
ක	ඓ	කෛ	ෛ	ෙ ෙ ක
ක	ඔ	කො	ො	ෙ ක ා
ක	ඕ	කෝ	ෝ	ෙ ක ා ්
ක	ඖ	කෞ	ෞ	ෙ ක ෟ

More often, the composite characters have a different shape to its base (core) character but its shape is a combination of the consonant and the modifier both together. (Figure3.2a)

Consonant has an inherent vowel ‘a’ sound and its pure form is obtained by removing that using “al-lakuna” (්). Sometimes, the composite characters have totally different shapes compared to the base character [25]. (Figure 3.2b) Some modifiers figures out different shapes for different base characters. (Figure 3.2c) This is valid for “al-lakuna”, “papilla” and “diga papilla”. For “Al-lakuna” forms are named as “kodiya” and “raehaena” whereas for papilla they are called “wak papilla” and “kon papilla” [23].

Even for the similar shaped composite characters as in Figure 3.2a, the modifier may be differing in size, orientation and appearance. (Figure 3.2d) Some modifiers have totally different shapes for different base characters too. (Figure 3.2e). Any vowel, consonant or composite character may be preceded to a semi-consonant.

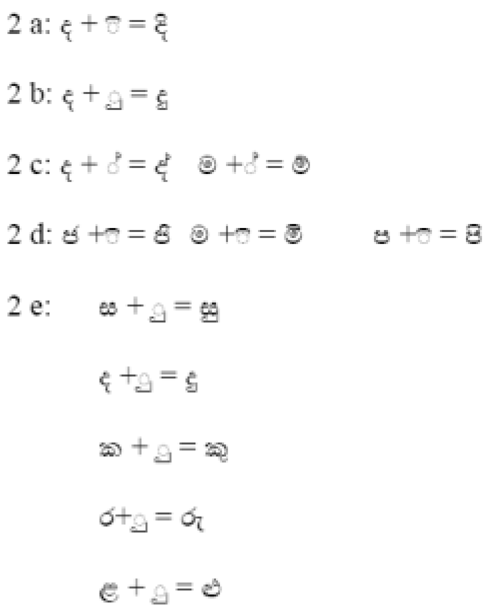


Figure 3.2 Different Consonant Modifier Combinations

3.1.2 Generating Sinhala words

Some Statistics for the language by using UCSC lexicon [24] as the data store is as follows.

Number of words = 6, 57,131

Number of Unique words = 70,131

Shortest Word = ද

Longest Word = ජ්‍යොතිශ්ශාස්ත්‍රඥයින්ගේ

A root word is used in the Sinhala language to generate many numbers of word forms in the Sinhala language [25]. The root word is the smallest building block and the word which will invoke the meaning. Inflectional root words are stems, and they are formed by the root word. The same word stem is able to generate several numbers of nouns, adjectives, adverbs or verbs, considering tense, number, person and purpose etc. This enables a word in the Sinhala language to be separated into prefix, stem, and suffix triples.

3.2 Methodology

3.2.1 Measuring OCR accuracy of Sinhala language

The research problem is to measure the OCR accuracy using the following pre-defined matrices.

Some of the accuracy measures that are of interest are given below [2]:

1) Character Accuracy

The text generated by a page-reading system is matched with the correct text to determine the minimum number of edit operations (character insertions, deletions, and substitutions) needed to correct the generated text [2]. This quantity is termed the number of errors. If there are n characters in the correct text, then the character accuracy is given by (n-#errors)/n

2) Word level accuracy

A popular use of a page-reading system is to create a text database from a collection of hard-copy documents. Information retrieval techniques can then be applied to locate documents of interest. For this application, the correct recognition of words is paramount. We define a word to be any sequence of one or more letters. In word accuracy, we determine the percentage of words that are correctly recognized. Each letter of the word must be correctly identified. Errors in recognizing digits or punctuation have no effect on word accuracy [2].

3) Accuracy by character class

The character set (alphabet) is divided into several classes, and the percentage of characters in each class that were correctly recognized is determined.

4) Phrase accuracy

Users search for documents containing specific phrases. We define a phrase of length L to be any sequence of L words.

For example, the phrases of length 3 in "University of Nevada, Las Vegas" are "The University of Nevada," "of Nevada, Las," and "Nevada, Las Vegas."

For a phrase to be correctly recognized, all of its words must be correctly identified. Phrase accuracy is the percentage of phrases that are correctly recognized, and we have computed it for L = 1 through 8. The phrase accuracy for length 1 is equal to the word accuracy.

Phrase accuracy reflects the extent to which errors are bunched or scattered within the generated text. Suppose two-page readers, A and B, have the same word accuracy but A has a higher phrase accuracy than B. Then A's errors are more closely bunched, and hence, easier to correct, than B's errors.

5) Non-stop word accuracy

Stop words are common words such as the, of, and, to, and a in English; de, la, el, y, and en in Spanish; and der, die, in, und, and von in German and සහ, තෙක්, සමග, හෝ in the Sinhala language.

These words are normally not indexed by a text retrieval system because they are not useful for retrieval. Users search for documents by specifying non-stop words in queries. With this in mind, we wish to determine the percentage of non-stop words that are correctly recognized, i.e., the non-stop word accuracy. To do this, a list of stop words for the Sinhala Language is required.

In each of the above measures, the Sinhala language satisfies the need of having the features which are needed to apply the above measures and feed data into the variables of each of the above categories. Considering the features and structure of the Sinhala language, all the above matrices can be applied to measure OCR accuracy for the Sinhala language. The rationale in choosing the above five categories to measure OCR accuracy is as follows:

1) Character Accuracy

As noted under section 3.1.1 Sinhala alphabet consists of 18 vowels 41 consonants and two semi-consonants which are unique from each other. Hence the accuracy measure (n-#errors)/n can be used to measure accuracy.

2) Word level accuracy

A Sinhala word lexicon such as the UCSC lexicon [24] can be used as the word database, and the percentage of correctly identified words in each OCR run can be measured

3) Accuracy by Character class

The Sinhala language has been divided into character classes as vowels, consonants and semi-consonants. These classes can be used to determine the percentage of characters in each class that were correctly recognized. The density of characters from each class can be tweaked as input parameters to generate result sets for accuracy

4) Phrase Accuracy

The Sinhala language contains words which will combine to formulate phrases. The phrases in an OCR output for the Sinhala language can be identified and used to determine the phase accuracy. The phrase length and number of phrases in the input document can be adjusted as variable inputs.

5) Non-stop word accuracy

Exploiting stop words similar to සහ, තෙක්, සමග, හෝ in the Sinhala language the Non-stop word accuracy can be measured with respect to a Sinhala OCR output. The density of stop words included in an input document can be exploited as a variable input to measure the accuracy.

In addition to the changing of the above variable input parameters in each of the above five measures, the following variables can be used for all of the above measurements as another input variable.

1) The font size of the text in the input document

2) Spacing between words in the input documents

3) Basic font styles of the input text (italic/bold)

CHAPTER 4: IMPLEMENTATION

A POST PROCESSING BASED METHODOLOGY TO INCREASE OCR ACCURACY FOR SINHALA SCRIPT ERROR HANDLING

In the OCR output some words that are identified are correct while some of the words identified are incorrect. If the words in the output does not match the words in the original document, the identified word is incorrect.

4.1 Unicode Errors

Due to how the Sinhala Unicode characters are implemented, an additional effort is needed to correct the errors observed in the output. The primary error that is affecting the output is the order of the Unicode characters and the modifiers. The Unicode of a modifier and the letter in the Sinhala alphabet does not follow their graphical representation sequence for the consonants. When writing the modifier is followed by the consonant. But in the Unicode representation the Unicode of the consonant is followed by the modifier.

Example

කෙ comprises of the modifier ෙ◌ (Kombuwa) and the consonant ක.

The individual Unicode strings for the characters are:

ෙ◌ - \udd9

ක - \ud9a

Although when writing the letter, the modifier is followed by the consonant, the Unicode sequence is as follows:

\ud9a\udd9

Furthermore, the above rule changes when it comes to vowel modifiers. Whenever a vowel is associated with a modifier, the character is considered a new character and gets its own Unicode value.

Example

The character අ can be associated with the modifier ◌ැ (Adapilla) and the output character ඇ is represented as a single Unicode string.

අ - \ud85

ැ - \udd0

ඇ - \ud87

The Unicode sequence will take the visual sequence and the result output string will be represented incorrectly.

කටයුතු කිරීම ෙහ්තුෙවන් ෙලා්ක ෙසෟඛ්‍ය සංවිධානයට ඇෙමරිකාව විසින් සපයන අරමුදල් කපා හරින බවට තර්ජනය

ෙකල්යංඹහු වැඩි දුරටත් කියා සිටිෙය් ෙකාවිඩ් සූ වසංගතය පාලනය සම්බන්ෙධයන් තම වගකීම ඉටු කිරීමට ෙලා්ක

ෙලා්ක ෙසෟඛ්‍ය සංවිධානෙය් යුෙරා්පිය කලාපෙය් අධ්‍යකභි තාඅාචාර්ය හාන්ස් ක්ලුජ්තා

Figure 4.1 Tesseract Output without normalization

<u>Example</u>

The output from the tesseract OCR engine will represent the character ඇ as අැ.

Hence to resolve the above issue, normalization engine is built into the proposed system which will process the output and change the Unicode sequence to the correct value or replace the Unicode with the correct Unicode value.

4.2 Syntactical Errors

Another means of improving the accuracy is to identify the syntactical rules in Sinhala language.

1) Some of the syntactical rules that has been identified is as follows [22][27][23].
2) The characters ඏ (SINHALA LETTER ILUYANNA) and ඐ (SINHALA LETTER ILUUYANNA) are currently not in use
3) In addition, the letter ඎ is very rarely in use.
4) No modifiers are used with ඞ (KANTAJA NAASIKYAYA)
5) A word cannot start with a consonant or semi consonant.

6) Usually a vowel will not be in the middle of a word. For that the dependent vowel form is used. [23][25]

7) ඞ can be replaced with the letter ං, but not vice versa.

8) The only word that starts with ඥ is ඥය

However, building all these rules into a syntactical rule engine and rectifying errors can be difficult to achieve. But some of these syntactical rules have been built into the rule engine and the system has been tested for any improvements in accuracy.

4.3 Confusion Pairs

Confusion pairs are a common OCR problem which occurs during the recognition phase. The problem is the OCR engine confusing the source text with a visually similar character. In Sinhala language following are some of the most commonly found visually similar confusion pairs.

Figure 4.2 Common Confusion Pairs

4.4 Word Level Errors

Contextual word recognition in post processing is performed on the OCR data stream at one level above character recognition, called the word level. By working at the word level, certain interferences and error rectifications are possible, which would not be feasible at the character level.

The most common post-processing technique operates at the word level is the dictionary look up method [12]. Techniques based on statistical information about the language are also used as well [12]. In statistical method, an n-gram, a letter string of size n [12] is used to filter out unacceptable candidates, on which substrings of n-grams cannot be generated, from the recognizer.

In order to correct word level errors caused by confusion pairs, the dictionary look up method was in used. The wordlist used is the UCSC Lexicon [24] which contains 70131 unique words.

The technique used is to look for characters which are in the confusion pairs and if the source text which is not a word in the lexicon and by replacing a confusion character, if a word can be generated, the current word will be replaced with the proper word to increase accuracy.

4.5 Error handling using post processing

The objective of post-processing is to correct errors or resolve ambiguities in OCR results by using contextual information at the character level, word level, at the sentence level and at the level of semantics.

Character level contextual post processing is mainly of two types Statistical methods and using a Lexicon [19]. The both methods involve in detecting and correcting of one or more errors. In Statistical method conditional probability of n-grams are gathered with training data to apply them to the testing data. If all the n-grams for the word existed, the word is considered as correct. In the other method, dictionary is used. If the word is found in the dictionary it is assumed that all its characters have been correctly recognized. Otherwise the same dictionary is used for correcting the errors in the recognized characters.

In addition, syntactical methods like grammar rules can also be incorporated to check for illegal character combinations. Some of such grammar rules are presence of two consecutive vowels or a word starting with a forbidden consonant or vowel [12].

4.6 Evaluation Approach

The evaluation approach that is used for this project will be experiment based. The datasets used for training tesseract will be the generic dataset that is already provided with the Tesseract OCR project. However, the input dataset for the OCR process will be generated in the following order to automate the error detection and analysis process.

- Create input as a text file with the desired word combinations. To extract meaningful text which has context, news articles from Sinhala e-newspapers will be used.
- Generate an image for OCR from the above text file. (The tool JtessBox editor [28] which is a tool used to create OCR training data will be used)

To compare the accuracy of the OCR output, the input image will be fed into few of the readily available Sinhala OCR tools. As of now some of the OCR tools which uses the tesseract OCR engine in the core are as follows.

1. "පෙළ කැටපත" - මුද්‍රිත අකුරු හඳුනාගැනීමේ මෘදුකාංගය [29]
2. Optical Character Recognition System for Sinhala [30]

The output from the OCR engine from this project will be compared against the output of the above engines to compare the accuracy.

To quantify and measure the accuracy of an input document against the original text the following accuracy measure [31] will be used.

- Word level accuracy

A popular use of a page-reading system is to create a text database from a collection of hard-copy documents. Information retrieval techniques can then be applied to locate documents of interest. For this application, the correct recognition of words is paramount. We define a word to be any sequence of one or more letters. In word accuracy, we determine the percentage of words that are correctly recognized. Each letter of the word must be correctly identified. Errors in recognizing digits or punctuation have no effect on word accuracy [31].

The Figure 4.3 shows the evaluation approach used to measure the OCR accuracy.

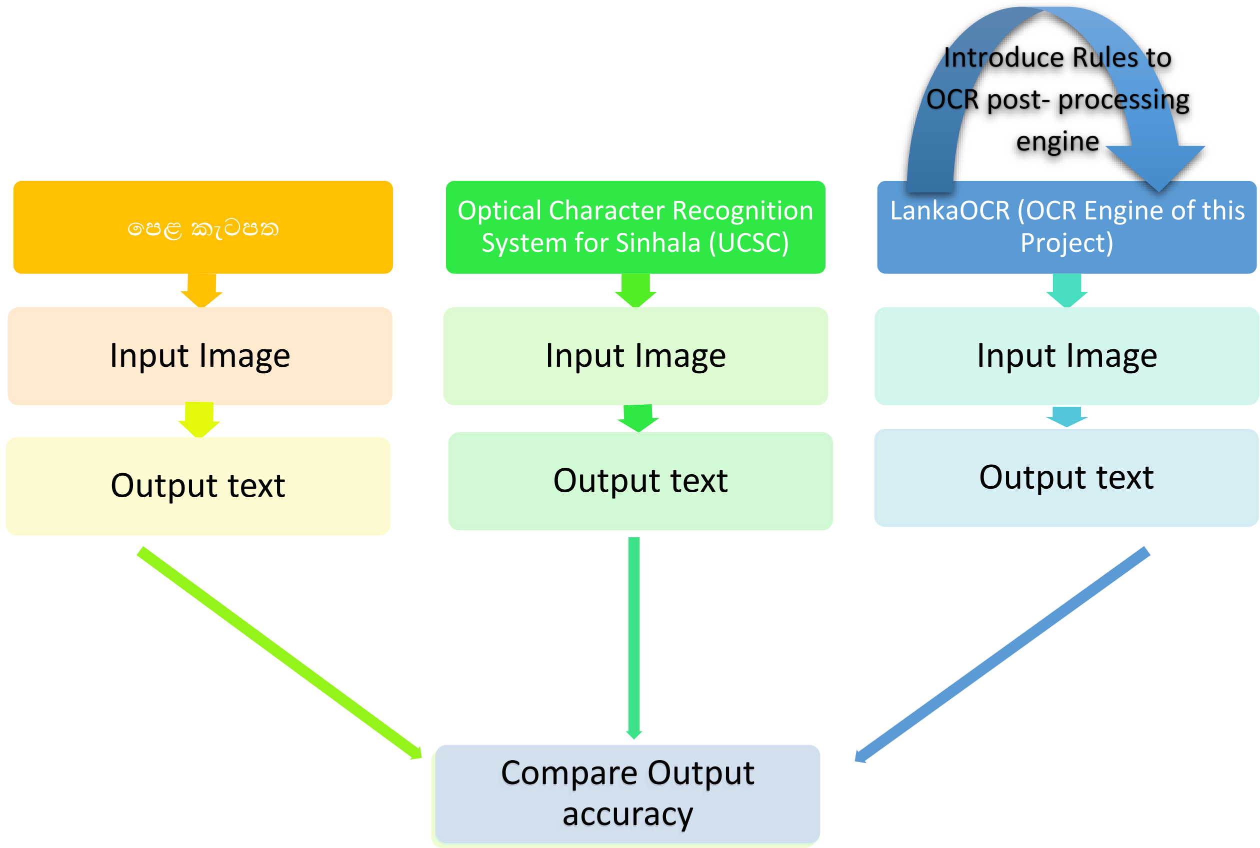


Figure 4.3 Evaluation approach used to measure the OCR accuracy

The experiment is based on the research hypothesis, the output of the tesseract OCR engine can be improved using post processing techniques. As exhibited in the Figure 1.1, with each iteration the OCR post processing engine will be refined with new rules and features to improve OCR accuracy. With each iteration the accuracy of the output will be diffed with the original text and the accuracy of the output text will be measured with the matrix word level accuracy.

Furthermore, the improvement will be compared keeping the output accuracy of the other two OCR engines "පෙළ කැටපත" [29] and Optical Character Recognition System for Sinhala [30].

4.7 Algorithms

4.7.1 Algorithm for the system

Generate OCR output in the HOCR format and using Tess4J [32].

// apply Unicode normalization for the output text in word level

Repeat for all the OCRed words in the output file

    Extract a word

    Apply Vowel Normalization Rules

    Apply Consonant Normalization Rules

    Apply the syntactic error correction rules

// check whether word is available to apply confusion rules

    If Sinhala word search it in the dictionary

        If a match found, write into the output

        Else

            Generate words with confusion pair list1

            If word with confusion character found

                Write the best match into the output

            Else

                Write the current word to the output

4.7.2 Algorithm for confusion pairs

Repeat for each component in a string from left to right

For each confusion pair in the list {

If match found

Generate word replacing component with confusion

Test the word against the Dictionary

If a hit add the word to candidate list

And manipulate the likelihood}

Select the highest scored candidate

CHAPTER 5: EVALUATION

The training dataset used to train the tesseract OCR engine is the readily available training data set, which is available in the tesseract project. The image format used as input source is tif. The input sample is an extract from a Sri Lankan E-newspaper. The font is “Iskoolapotha”.

Input contains 2 tif pages which includes punctuations and numbers. (Arabic Numerals)

ට්‍රම්ප් පරිපාලනය මෙන්ම රිපබ්ලිකන් නියෝජිතයන්ද වෛරස ව්‍යාප්තිය පිළිබඳව චීනයේ සංඛ්‍යා ලේඛන සහ
දත්ත වැරදි
සහ නොමග යවන සුළු බව චෝදනා කරන අතර ලෝක සෞඛ්‍ය සංවිධානය එම වැරදි දත්ත මත පිහිටා කටයුතු
කරමින්
ලොව පුරා වෛරසය ව්‍යාප්ත වීමට ඉඩ ලබා දුන් බව චෝදනා කරයි.චීනය සමස්ත කොරෝනා මරණ ප්‍රමාණය
3335
ලෙස වාර්ථා කලත් වූහාන් වල පමණක් 40000 කට අධික පිරිසක් මියගොස් ඇති බව නිල නොවන වාර්ථා
හෙලිකරන
බව ඔවුන් ප්‍රකාශ කරයි.පසුගිය සතියේ රිපබ්ලිකන් සෙනේට් සභික මාර්කෝ රූබියෝ ප්‍රකාශයක් කරමින්
ලෝක සෞඛ්‍ය සංවිධානයේ අධ්‍යක්‍ෂ ජනරාල් ආචාර්ය ටෙඩ්‍රොස් අද්නාම් ගෙබ්‍රෙයිසස් ඉල්ලා අස්විය යුතු බවට
ප්‍රකාශ කලේය.ගෝලීය ප්‍රජාව නොමග යැවීමට චීනයට උදව් කල බවට ඔහු ලෝක සෞඛ්‍ය සංවිධානයට
චෝදනා කලේය.

Figure 5.1 Sample image of the input used for OCR

The input image has 419 words in total.

The table 5.1 gives a summary of the words in the input source.

Table 5.1 Summary of words in the Input Source

Total Number of Words	419
Number of Unique words	239
Most Frequent word	ලෝක (9 occurrences)
Number of punctuations	16
Number of Arabic Numeral occurrences	6
Number of words which occur More than once in the text	66

5.1 Results from the output using the default training data without any post processing

The output from the tesseract using the readily available sin.traindata (tesseract language training data file for the font Iskoolapota)) produced the following results.

Total Number of words in the input = 419

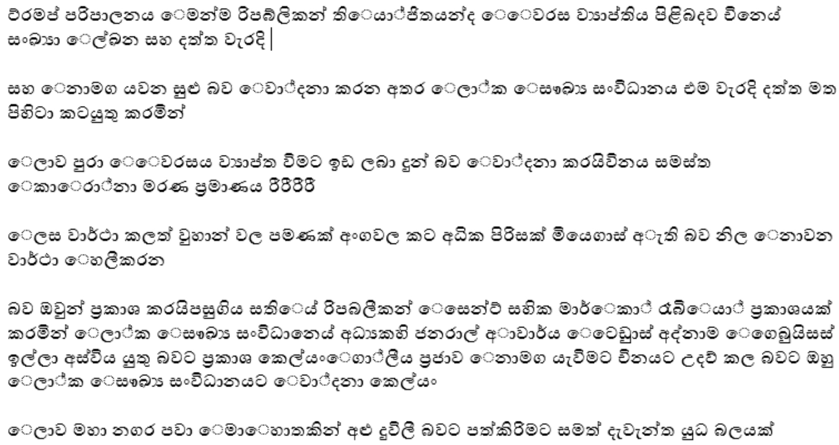
Total Number of words in the output = 419

No of words in identified correctly = 223

Misrecognized words = 196

The word level accuracy (223/419) * 100 = 53.22%

The figure 5.2 contains a screenshot from the output and the figure 5.3 is a screenshot from the comparison between the original text and the output text.



ට්‍රම්ප් පරිපාලනය ෙමන්ම රිපබ්ලිකන් තිෙයාර්ජිතයන්ද ෙෙවරස ව්‍යාප්තිය පිළිබඳව චීනෙය්
සංඛ්‍යා ෙල්ඛන සහ දත්ත වැරදි

සහ ෙනාමග යවන සුළු බව ෙචාර්දනා කරන අතර ෙලාර්ක ෙසෟඛ්‍ය සංවිධානය එම වැරදි දත්ත මත
පිහිටා කටයුතු කරමින්

ෙලාව පුරා ෙෙවරසය ව්‍යාප්ත වීමට ඉඩ ලබා දුන් බව ෙචාර්දනා කරයිවිනය සමස්ත
ෙකාෙරාර්නා මරණ ප්‍රමාණය ට්ට්ට්ට්

ෙලස වාර්ථා කලත් වුහාන් වල පමණක් අංගවල කට අධික පිරිසක් මියෙගාස් අැති බව නිල ෙනාවන
වාර්ථා ෙහලිකරන

බව ඔවුන් ප්‍රකාශ කරයිපසුගිය සතිෙය් රිපබලිකන් ෙසෙන්ට් සභික මාර්ෙකාර් රූබිෙයාර් ප්‍රකාශයක්
කරමින් ෙලාර්ක ෙසෟඛ්‍ය සංවිධානෙය් අධ්‍යකහි ජනරාල් අාචාර්ය ෙටෙඩ්‍රාස් අද්නාම ෙගෙබ්‍රයිසස්
ඉල්ලා අස්විය යුතු බවට ප්‍රකාශ කෙල්යංෙගාර්ලීය ප්‍රජාව ෙනාමග යැවීමට චීනයට උදව් කල බවට ඔහු
ෙලාර්ක ෙසෟඛ්‍ය සංවිධානයට ෙචාර්දනා කෙල්යං

ෙලාව මහා නගර පවා ෙමාෙහාතකින් අළු දූවිලි බවට පත්කිරිමට සමත් දැවැන්ත යුධ බලයක්

Figure 5.2 output from the default training data

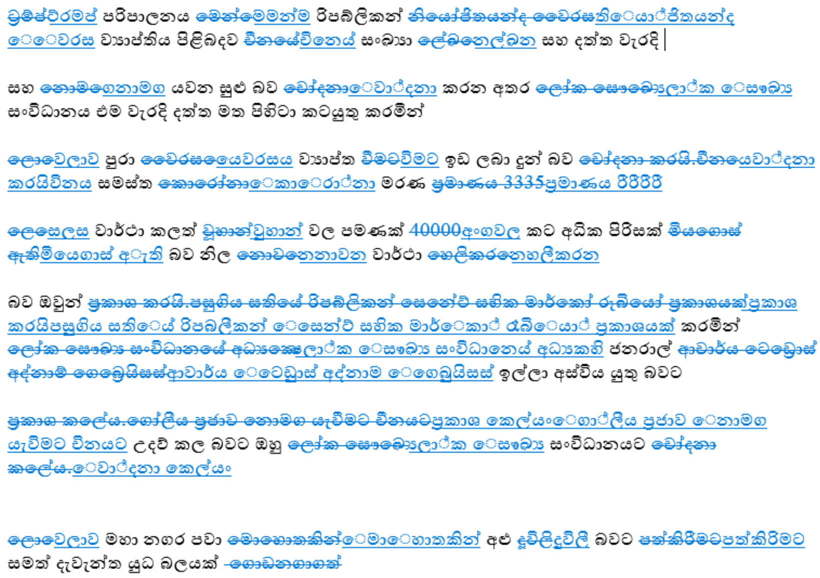


Figure 5.3 Comparison between the original text and the output form the default training data

The following table shows different types of errors identified during this stage.

Table 5.2 Stage 1 output of the different errors

Input Word	Output Word	Explanation
මෙන්ම	ෙමන්ම	Unicode error which needs to be fixed by changing the Unicode sequence of the "කොම්බුව" with the consonant.
වෛරස	ෙෙවරස	Unicode error which needs to be fixed by allocating the correct Unicode sequence and the correct Unicode for "කොම්බු දෙක".
ලෝක	ෙලා්ක	Unicode error and should be corrected by applying the correct Unicode sequence followed by the consonant
සෙනෝට්	ෙසෙන්ට්	Applying the correct Unicode sequnce should resolve the error
රිපබ්ලිකන්	රිපබලීකන්	Error in recognition from tesseract. a confusion pair "ලි" and "ලී" is observed
හෙලිකරන	ෙහලීකරන	Error in Unicode which can be fixed by normalizing but the error in recognition for the confusion pair "ලි" and "ලී" needs to be handled
ආචාර්ය	අාවාර්ය	Error in Unicode. The modifier "ඇල පිල්ල"has been recognized and needs to be replaced by the correct Unicode. The confusion pair "ච" and "ව" is observed
නැත්තාවූත්	නැත්තාවුත්	No Unicode errors, the confusion pair "වු" and "වූ" has resulted in the error
අධ්‍යක්‍ෂ	අධ්‍යකෂි	The joined letter has not been recognized correctly.

5.2 Results from the OCR Engine “පෙළ කැටපත”

The sample input used in the above instance was fed into the OCR Engine “පෙළ කැටපත” [29] developed by the Language Research Training Laboratory of UCSC and the output was extracted.

The output from the above OCR engine yielded the following results.

Total Number of words in the input	= 419
Total Number of words in the output	= 419
No of words in identified correctly	= 233
Misrecognized words	= 186
The word level accuracy (233/419) * 100	= 55.61%

The Figure 5.4 contains a screenshot of the output from the “පෙළ කැටපටත” OCR engine. [29]

ඩ්‍රම්ප් පරිපාලනය මෙන්ම රිපබ්ලිකන් නියෝජිතයන්ද වෛරස ව්‍යාප්තිය පිළිබදව චීනයේ සංඛ්‍යා ලේඛන සහ දත්ත වැරදී

සහ නොමග යවන සුළු බව වෝදනා කරන අතර ලෝක සෞඛ්‍ය සංටිධානය එම වැරදී දත්ත මත පිහිටා කටයුතු කරමින්

ලොව පූරා වෛරසය ව්‍යාප්ත වීමට ඉඩා ලබා දූන් බව වෝදනා කරයි.චීනය සමස්ත කොරෝනා මරණ ප්‍රමාණය 3335

ලෙස වාර්ථා කලත් වූහාන් වල පමණක් 40000 කට අධීක පිරිසක් මියගොස් ඇති බව නිල නොවන වාර්ථා හෙලීකරන

බව ඔවුන් ප්‍රකාශ කරයි.පසුභිය සනියේ රිපබ්ලිකන් සෙනේට් සභික මාර්කෝ රැබියෝ ප්‍රකාශයක් කරමින් ලෝක සෞඛ්‍ය සංටිධානයේ අධ්‍යක්ෂ ජනරාල් ආවාර්ය ටෙඩේරාස් අද්නාම් ගෙබ්‍රෙයිසස් ඉල්ලා අස්විය යුතු බවට ප්‍රකාශ කලේය.ගෝලීය ප්‍රජාව නොමග යැට්මට චීනයට උදව් කල බවට ඔහු ලෝක සෞඛ්‍ය සංටිධානයට

වෝදනා කලේය.

ලොව මහා නගර පවා මොහොතකින් අළු දූවිලී බවට පත්කිරිමට සමන් දැවැන්ත යුධ බලයක් ගොඩානගාගන් මහා බලවන් රටවල් පවා අසරණ භාවයට ඇද දැමූ ඇසට නොපෙනෙන කුඩා කොට්ඩ්-19 වෛරසය ස්වභාවධර්මය

Figure 5.4 output from the “පෙළ කැටපටත” OCR engine

Comparing the results from output from the default training data vs “පෙළ කැටපටත” OCR engine [29].

Table 5.3 Comparison of results from default training data vs “පෙළ කැටපටත” OCR engine

Property	Default training data	පෙළ කැටපත
Total Number of words in the input	419	419
Total Number of words in the output	419	419
No of words in identified correctly	223	233
Misrecognized words	196	186
The word level accuracy	53.22%	55.61%

5.3 Introducing Unicode normalization to the OCR output

To increase the OCR accuracy and to rectify the Unicode errors described under section 4.1, a normalization engine was built to the application. This is an application of post processing in an attempt to determine whether it can increase the accuracy of the output from tesseract OCR engine. Enabling the normalization engine yielded the following results.

Total Number of words in the input = 419

Total Number of words in the output = 419

No of words in identified correctly = 307

Misrecognized words = 112

The word level accuracy (307/419) * 100 = 73.27%

With the introduction of the normalization to the output the word level accuracy increased by 20.05% which is a significant improvement.

Figure 5.5 is a screenshot of the results obtained after enabling normalization rules to the tesseract output with the readily available training data file for Sinhala language.

යලි සිතා බැලිය යුතු බවයිං පිටපිට ජනවාරි මාසයේදී ෆොක්ස් රූපවාහිනී නාලිකාවේ බිස්නස් නිවුස් හි පැනයකට පිළිතුරු දෙමින් එක්සත් ජනපද රාජ්‍ය ලේකම විල්බර් රොස් පවසා සිටියේ චිනයේ මාරාන්තික වෛරසයේ පැතිරිම

එක්සත් ජනපදයට වාසිදායක වන බවයිං ඔහු තවදුරටත් පවසා සිටියේ චිනයෙන් ගිලිහෙන රැකියා අවස්ථා ග්‍රහණය

කර ගැනිමට එක්සත් ජනපදය කටයුතු කල යුතු බවයිං ජනාධිපතිවරණයක් අත ලග ඇති අවස්ථාවේ ප්‍රතිවිරුද්ධ කොන්සර්වෙටිව නියෝජිතයන්ගේ දැඩි විවේචනයට එරමප් ඉලක්ක වි ඇති අතර තමන්ට එල්ල වන චෝදනාවන්ගෙන් නිදහස් විම සඳහා ඔහු ලෝක සෞඛ්‍ය සංවිධානය දෙසට ඇඟිල්ල දිගු කරමින් සිටින බව පැහැදිලියං

Figure 5.5 Tesseract output after applying the normalization rules

Table 5.4 Comparison of results from default training data, "පෙළ කැටපටත" OCR engine and output after normalizing

Property	**Default training data**	පෙළ කැටපත	**Normalized output with default training data**
Total Number of words in the input	419	419	419
Total Number of words in the output	419	419	419
No of words in identified correctly	223	233	307
Misrecognized words	196	186	112
Percentage of errors corrected at this stage	N/A	N/A	42.86%
The word level accuracy	53.22%	55.61%	73.27%

After the application of normalization rules, a significant portion of the Unicode errors were resolved. However, there were some more errors which did not get resolved. Following is an analysis of the resolved and unresolved errors after stage 2.

Table 5.5 Stage 2 output of different errors

Error from stage1	Output from stage 2	Explanation
ෙමන්ම	මෙන්ම	Resolved with normalization.
ෙෙවරස	වෛරස	Resolved with normalization.
ෙලා්ක	ලෝක	Resolved with normalization.
ෙසෙන්ට	සෙනේට	Resolved with normalization.
රිපබ්ලීකන්	රිපබ්ලිකන්	The confusion pair 'ලි','ලී' exists and makes the word incorrect
ෙහලීකරන	හෙලීකරන	The Unicode error has been resolved through normalization but the error in recognition for the confusion pair "ලි" and "ලී" needs to be handled
අාවාර්ය	ආවාර්ය	The Unicode error has been resolved through normalization the confusion pair "ච" and "ව" needs to be fixed
නැත්තාවුත්	නැත්තාවුත්	The confusion pair "වු" and "වූ" from stage 1 still remains the same.
අධ්‍යකභි	අධ්‍යකභි	The joined letter has not been recognized correctly as observed in stage 1.

Apart from the errors noted above, there are some errors which have been introduced from the recognition phase. These errors are mainly due to the incompleteness of training data. (Missing characters in the training data, punctuations not recognized properly and the training data missing the Arabic numerals)

These errors are described with details in the next analysis after introducing the dictionary correction feature for confusion pairs.

5.4 Dealing with confusion pairs

It was observed that the confusion pairs have a sizable impact on the accuracy of the OCR from the results obtained until now. To rectify these errors and increase the accuracy, a new feature to identify confusion pairs and replace the errored words through a dictionary look up was introduced to the OCR engine.

The word look-up is a complex feature which can be improved in many ways. For example, the word look -up can be introduced to correct word errors in the output by means of N-grams. However, this feature has been introduced to look for a word which contains a confusion pair/pairs in it and by swapping a confusion pair/pairs if a legitimate word can be found in the word lexicon the current word will be replaced by the word from the lexicon.

The word lookup feature is another post processing technique which was used in this project to increase the accuracy of the OCR output. After introducing the correction feature to deal with confusion pairs with a word lookup, the following results were yielded.

Total Number of words in the input	= 419
Total Number of words in the output	= 419
No of words in identified correctly	= 361
Misrecognized words	= 58
The word level accuracy (307/419) * 100	= 86.16%

With the introduction of the feature to correct confusion pairs, the word level accuracy increased from 73.27% to 86.16%. This is a 12.89% increase of accuracy when compared with the results from stage 2 which introduced the Unicode normalization feature.

Figure 5.6 is a screenshot of the results obtained after introducing the correction feature for confusion pairs to the tesseract output with the readily available training data file for Sinhala language. The highlighted text in the figure are some of the corrections which were done during this phase.

ටිරමප් පරිපාලනය මෙන්ම රිපබ්ලිකන් නියෝජිතයන්ද වෛරස ව්‍යාප්තිය පිළිබඳව චීනයේ සංඛ්‍යා ලේඛන සහ දත්ත වැරදි

සහ නොමග යවන සුළු බව චෝදනා කරන අතර ලෝක සෞඛ්‍ය සංවිධානය එම වැරදි දත්ත මත පිහිටා කටයුතු කරමින්

ලොව පුරා වෛරසය ව්‍යාප්ත වීමට ඉඩ ලබා දුන් බව චෝදනා කරයිවිනය සමස්ත කොරෝනා මරණ ප්‍රමාණය ටිටිටිටි

ලෙස වාර්ථා කලත් වුහාන් වල පමණක් අංගවල කට අධික පිරිසක් මියගොස් ඇති බව නිල නොවන වාර්ථා හෙළිකරන

Figure 5.6 Tesseract output after correcting errors from confusion pairs

The following table is a summary of errors corrections done with a comparison of word level accuracy during each stage.

Table 5.6 Comparison of results from default training data, “පෙළ කැටපටත” OCR engine and output after correcting confusion pairs

Property	Default training data	පෙළ කැටපත	Normalized output with default training data	Correcting confusion pairs
Total Number of words in the input	419	419	419	419
Total Number of words in the output	419	419	419	419
No of words in identified correctly	223	233	307	361
Misrecognized words	196	186	112	58
No of words corrected during this phase compared to first phase	N/A	N/A	74	128
Percentage of errors corrected at this stage	N/A	N/A	39.78%	68.82%
The word level accuracy	53.22%	55.61%	73.27%	86.16%

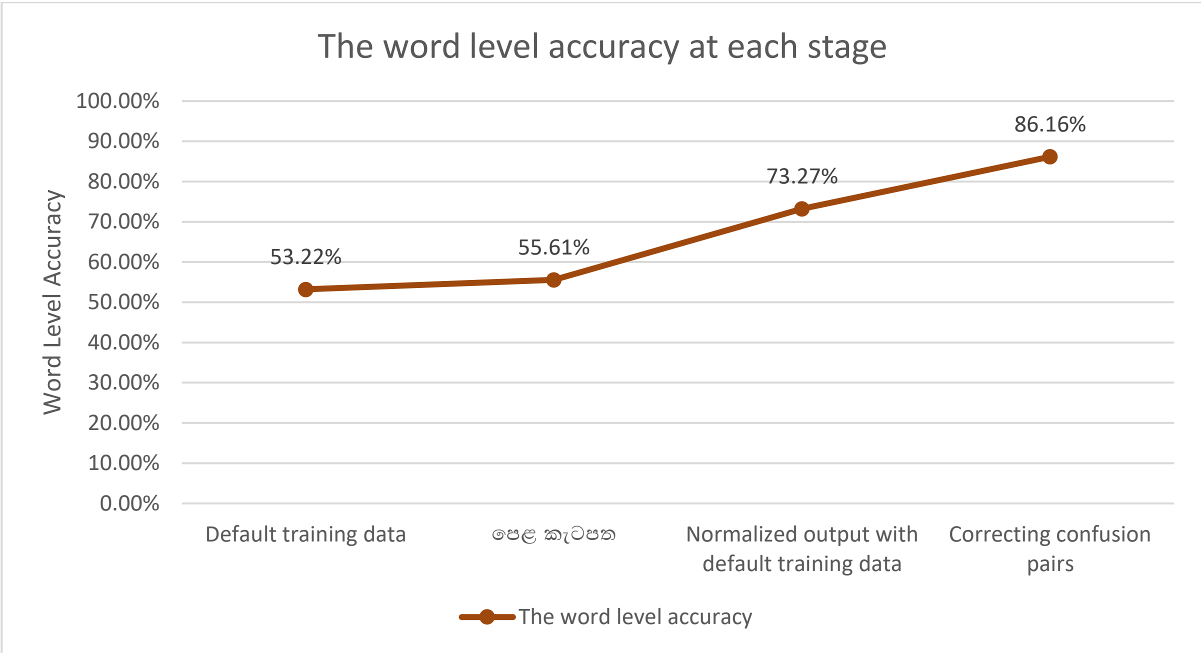
After introducing confusion pair correction for words, a significant portion of the words with errors due to confusion pairs were resolved. However, there were some more errors which did not get resolved. Following is an analysis of the resolved and unresolved errors after stage 3.

Table 5.7 Stage 3 output of different errors

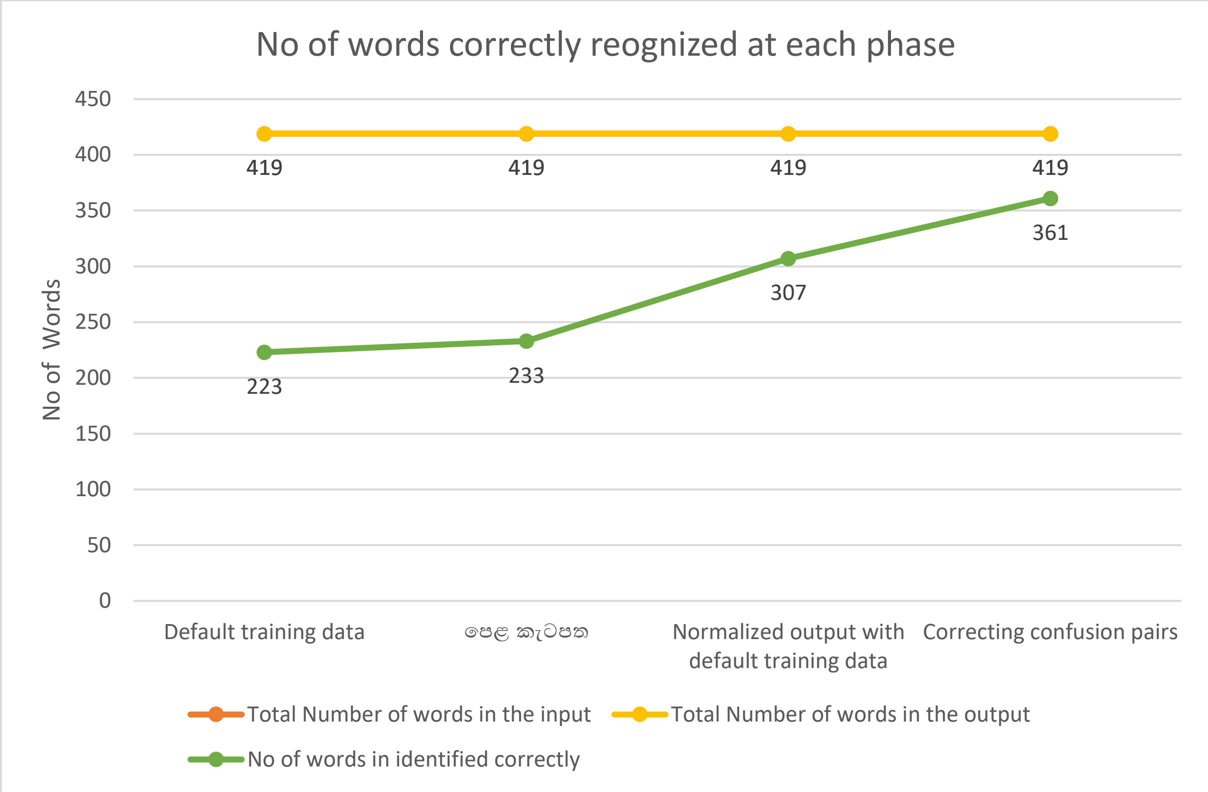
Error from stage2	Output from stage 3	Explanation
රිපබලීකන්	රිපබ්ලිකන්	The confusion pair ‘ලි’,’ලී’ has been resolved
හෙලීකරන	හෙලිකරන	The Unicode error has been resolved through normalization but the error in recognition for the confusion pair "ලි" and "ලී" needs to be handled
ආවාර්ය	ආචාර්ය	The confusion pair "ච" and "ව" has been resolved.
නැත්තාවුත්	නැත්තාවූත්	The confusion pair "වු" and "වූ" has been resolved
අධ්‍යකහි	අධ්‍යකහි	The joined letter has not been recognized correctly as observed in stage 1 and stage 2
කරයිවිනය	කරයිවිනය	The correct word sequence from the input source is කරයි. චීනය. However, the punctuation “full stop” has not been recognized correctly

ඍබියෝ	ඍබියෝ	The character ‘රෘ’ has been mis-recognized as ‘ඍ’. This is a recognition level error which needs to be addressed at the training data level
සමබන්ධයෙන්	සමබන්ධයෙන්	The error is due to the character ‘ම’ is recognized instead of the expected ‘ම්’. This is an issue with the training data and needs to be corrected from the training data.

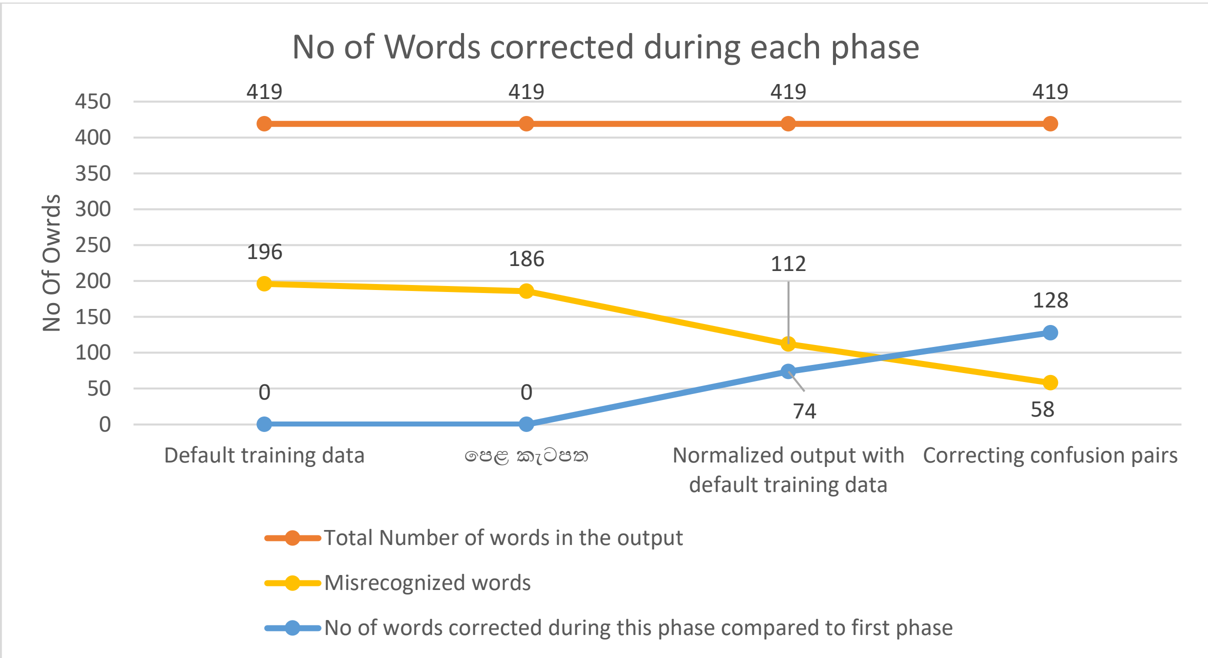
Analyzing the above output, it is evident that the OCR accuracy has improved from stage 2. However, there is room for improvement at the recognition phase by improving the quality of the training data used with tesseract.



Graph 6.1 Word level accuracy at each level



Graph 6.2 No of words correctly recognized at each phase



Graph 6.3 No of words corrected recognized at each phase

CHAPTER 6: CONCLUSION

6.1 Discussion

The default training data for Sinhala language readily available with the Tesseract OCR engine was used during the recognition phase of this project. The post processing features were built to the OCR engine to improve the OCR accuracy from Tesseract.

To compare the results, the 'පෙල කැටපත' OCR software developed by the Language Technology Research Laboratory of UCSC was used as the baseline. The Accuracy measure which was used in phase was word level accuracy. The data set which was used to test OCR accuracy contained a combination of Sinhala characters covering all character classes and most of their permutations. Some of the character classes and permutations used are:

1) Vowels
2) Consonants
3) Conjunct Characters (Eg: ඥ)
4) Special modifiers (Eg:)

The input image format for the proposed system was tif. However, the input for the 'පෙලකැටපත' OCR tool, the input source had to be of type jpg. Hence the dataset which was a 2-page tif image, was converted to 2 jpg images.

The first set of results from 'පෙලකැටපත' tool and the default training data from the Tesseract yielded the word level accuracy of 55.61% and 53.22% respectively. Comparing the above figures, it was observed that the 'පෙලකැටපත' tool was able to produce slightly better word level accuracy. However, the results from both of the above tools poor. The noticeable difference between the two outputs was that the 'පෙලකැටපත' OCR engine did not have any Unicode character sequence confusions.

However, due to the way that tesseract is trained for the language the tesseract output will have Unicode errors.

<u>For example:</u>

The letter කෞ consists of the following characters

SINHALA VOWEL SIGN KOMBUVA ෙ◌

SINHALA LETTER ALPAPRAANA KAYANNA ක

SINHALA VOWEL SIGN GAYANUKITTA ◌ෟ

In Unicode, the following Unicode values are assigned to each of these glyphs:

0x0DD9 SINHALA VOWEL SIGN KOMBUVA ෙ◌

0x0D9A SINHALA LETTER ALPAPRAANA KAYANNA ක

0x0DDF SINHALA VOWEL SIGN GAYANUKITTA ◌ෟ

However, to generate the character කෞ the glyphs Kombuwa and the Gayanukitta has a single Unicode in the Sinhala Unicode character list.

0x0DDE SINHALA VOWEL SIGN KOMBUVA HAA GAYANUKITTA ෙ◌ෟ

And the order in which the Unicode sequence is assigned is different to that of the visual sequence. That is to generate the character කෞ the proper Unicode sequence would be 0x0D9A(ක)+0x0DDF (◌ෟ). But since tesseract is recognizing the character sequence in the visual order as 3 glyphs in the order 0x0DD9(ෙ◌) + 0x0D9A(ක)+ 0x0DDF (◌ෟ) the final output will be rendered as ෙ◌කෟ.

To address this issue, a Unicode normalization engine was built to the proposed system. With the introduction of the Unicode Normalization engine, the word level accuracy of the output raised to the percentage 73.27%. This was a 20.05% increase from the previous value that was generated from the raw Tesseract output figure 53.22%.

While the input contained 419 words, the output from the tesseract engine contained 419 words. Hence, no words were missed. However, out of the 419 words recognized, there was clear evidence that none of the Arabic numerals or the punctuations were recognized. Furthermore, there were some characters which seemed to be missing in the original tesseract training dataset.

The number of misrecognized words in the output in the raw tesseract output was 196 and that number was brough down to 112 with the introduction of the Unicode normalization feature.

The next step was to identify the confusion pairs which prevented a word from being accurate. A confusion pair is a visually similar characters which is incorrectly identified as it’s incorrect version during the recognition phase. An example of this is the Sinhala letter ව being recognized as ච. Both these characters have visual similarities which the OCR engine might confuse and identify one character incorrectly as the other character.

The resulting word could be a legit word or an illegitimate word. The approach followed during this project to correct confusion pairs is to use a word lexicon along with the confusion pairs.

If an output word after normalization is not available in the word list and if it contains a confusion pair, it is assumed that the current word is incorrect. Based on this assumption, the confusion character is substituted in the word with its associated pair character and then the word lexicon is probed for hit. If a match is found it is assumed that hit is the legit word that fits the current context and is replaced with the word.

Building this feature into the current system yielded positive results and the word level accuracy of the output was further increased up to 86.16%. This is a 12.89% increase when compared to the previous stage and an overall 34.94% from the original figure yielded from the raw tesseract training data output. During this phase the total number of incorrectly recognized were further brough down to 58 from 112 words which was yielded in the previous stage. Out of the words which were recognized incorrectly, 15 words were due to the current training dataset not containing the input characters so that the tesseract engine can recognize the characters correctly. Furthermore, the punctuations and the Arabic numerals which is a part of the input dataset used for OCR has not been recognized by the current training dataset.

The Java wrapper library and the current version of the tesseract OCR engine provides the ability to use multiple training data sets. The current version of the system supports multiple training data files. Hence, the above errors are can be mitigated with the introduction of more training data to the tesseract training dataset.

Considering the word level accuracy at each stage, a clear improvement of the accuracy is observed. So, we can safely conclude that the application of the proposed post processing techniques has improved the OCR accuracy of the output from the Tesseract OCR engine. Hence it can be said that the goals of this research project have been achieved to a satisfactory level and there is room for improvement for the project to reach to a commercial level.

6.2 Future work

The current word level correction done for the confusion pairs is using a word lexicon and has limited capability to correct word errors. Furthermore, comparing for each confusion pair and performing a word look up can be a costly operation depending on the number of confusion pairs which can be identified for a character and the number of such characters found in a word.

The current system does not process a word for multiple hits when looking for confusion pairs. That is the first word found as a hit for an incorrect word in the text will be substituted and the post processing engine will stop substituting confusion pairs further.

This feature can be improved by introducing a probabilistic feature to the pick the best match for confusion characters. The probability of a word appearing in a text can be considered when replacing an incorrect word with confusion character which yields multiple hits when processing. Another approach which could be used with the above feature is to consider the context of the documents scanned. If there is a way to obtain some metadata about the source document (Eg: an article about science, an article about history) depending on the context the wordlists can be used for post processing.

A context-based lexicon can be defined to be used with a source document which has text related to a matching context. This feature could further be improved to correct the errors in the source document itself which would provide a meaningful output from the input document.

Language level features like extensive grammar rule check up and use linguistic features of a word like root sems, adverbs and adjectives to improve the OCR accuracy can also be considered as future work. However, implementing such language agonistic features will need researching and gaining a deeper understanding of the Sinhala language. This can be facilitated by a dictionary look up methodology to increase accuracy once the initial analysis of the language features like identifying the root words and the variants like Adverbs and adjectives is implemented.

REFERENCES

[1] "tesseract-ocr/tesseract", GitHub, 2020. [Online]. Available: https://github.com/tesseract-ocr/tesseract. [Accessed: 22- Jan- 2020].

[2] S. Rice, F. Jenkins and T. Nartker, "The Fifth Annual Test of OCR Accuracy", Information Science Research Institute University of Nevada, Las Vegas, 1996.

[3] Christensson, Per. "OPTICAL CHARATER RECONGNITION Definition." TechTerms. (April 16, 2018). Accessed Oct 25, 2019. https://techterms.com/definition/Optical Character Recognition.

[4] "Optical character recognition (OCR)", En.wikipedia.org, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Optical_character_recognition#Types. [Accessed: 22- Oct- 2019].

[5] Kai Niklas, "Unsupervised Post-Correction of OCR Errors," Diploma thesis, Univ. Hannover, Jun. 2010.

[6] Xiaofan Lin, "DRR Research beyond COTS OCR Software: A Survey," in SPIE Conf. Document Recognition and Retrieval XII, San Joes, CA, Jan. 2005

[7] "tesseract-ocr/tesseract", GitHub, 2019. [Online]. Available: https://github.com/tesseract-ocr/tesseract#about. [Accessed: 23- Oct- 2019].

[8] B. Chaudhuri and U. Pal, "A complete printed Bangla OCR system", Pattern Recognition, vol. 31, no. 5, pp. 531-549, 1998. Available: 10.1016/s0031-3203(97)00078-2.

[9] X. Tong and D. Evans, "A Statistical Approach to Automatic OCR Error Correction in Context", ACL Anthology, 2019. [Online]. Available: https://www.aclweb.org/anthology/W96-0108/. [Accessed: 25- Oct- 2019].

[10] Bansal, Veena & Sinha, R.M.K.. (2000). Integrating knowledge sources in Devanagari text recognition system. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on. 30. 500 - 505. 10.1109/3468.852443.

[11] Sharma, Dharam & Lehal, Gurpreet & Mehta, Sarita. (2009). Shape Encoded Post Processing of Gurmukhi OCR. Proceedings of the International Conference on Document Analysis and Recognition, ICDAR. 788-792. 10.1109/ICDAR.2009.180.

[12] G S Lehal, Chandan Singh, “A post-processor for Gurmukhi OCR,” Sadhana , vol. 27, Part 1, February 2002, pp. 99–111.

[13] Kenneth Ward Church, Patrick Hanks, “Word Association Norms, Mutual Information, and Lexicography,” Computational Linguistics, vol. 16, Mar. 1990

[14] Thien M Ha , H Bunke, “Image Processing Methods For Document Image Analysis,” in Handbook Of Character Recognition And Document Image Analysis, World Scientific Publishing Company, Singapore, May 1997,ch 1, pp 1-47.

[15] Dulip Herath, Nishantha Medagoda, “Research Report on the Preprocessing Engine of the Optical Character Recognition System for Sinhala Scripts,” Language Technology Research Laboratory,Univ. Colombo, Sri Lanka.

[16] "Page segmentation method", GitHub, 2019. [Online]. Available: https://github.com/tesseract-ocr/tesseract/wiki/ImproveQuality#page-segmentation-method. [Accessed: 23- Oct- 2019].

[17] Mohamed Cheriet, Nawwaf Kharma, Cheng-Lin Liu, Hing Y. Suen, Character Recognition Systems - A Guide For Students And Practitioners, A John Wiley and Sons, New Jersey, 2007.

[18] A. Lawrence Spitz, "Shape-Based Word Recognition," Document Analysis and Recognition, Vol. 1, pp178-190, May. 1999.

[19] H. Bunke, “A Fast algorithm for finding the nearest neighbor of a word in a dictionary,” in Proc. 2nd Int. Conf. .Document Analysis and Recognition, Nov. 1993

[20] T.K. Ho, J.J. Hull, and S.N. Srihari, "Word Recognition with Multi-Level Contextual Knowledge,” in Document Analysis and Recognition Int. Conf., Saint Malo, France, 1991.

[21] A. Dengel, R. Hoch, F. Hones, T. Jager, M. Malburg, and A. Weigel. “Techniques for Improving OCR Results,” in Handbook of Character Recognition and Document Image Analysis, World Scientific Publishing Company, Singapore , May 1997, ch 4, pp 227–258.

[22] ජාතික අධ්‍යාපන ආයතනය, සිංහල ලේඛන රීතිය, තෙවන මුද්‍රණය, ජාතික අධ්‍යාපන ආයතනය,කොළඹ, ශ්‍රී ලංකා, 2001.

[23] ජේ.බී. පීරිස්, නූතන සිංහල ලේඛන ව්‍යාකරණය, ප්‍රථම මුද්‍රණය, සීමා සහිත ලේක් හවුස් ඉන්වෙස්මන්ට්ස් සමාගම, කොළඹ, ශ්‍රී ලංකාව,1990

[24]"Sinhala corpus of 10 million words", 2012. [Online]. Available: http://www.ucsc.cmb.ac.lk/ltrl.[Accessed: 23- Oct- 2019].

[25] Lalith Premaratne, E Jarpe, Josef Bigun, "Lexicon and Hidden Markov Model based Optimization of the recognizes Sinhala Script," Pattern Recognition Letters, vol. 27, pp 696-705, Apr. 2006

[26] "OPTICAL CHARACTER RECOGNITION (OCR)", Datamerj, 2020. [Online]. Available: https://datamerj.com/what-is-ocr.html. [Accessed: 24- Aug- 2019].

[27] ජේ.බී. පීරිස්, සිංහල අක්ෂර විචාරය, ප්‍රථම මුද්‍රණය, සුමිත ප්‍රකාශන, ශ්‍රී ලංකා, 2006.

[28] Vietocr.sourceforge.net. 2020. Jtessboxeditor - Tesseract Box Editor & Trainer. [Online] Available at: <http://vietocr.sourceforge.net/training.html> [Accessed 7 April 2020].

[29] Subasa.lk. 2020. Optical Character Recognition. [Online] Available at: <http://subasa.lk/aocr/ocr.php> [Accessed 7 April 2020].

[30] UCSC. 2020. Optical Character Recognition System for Sinhala - UCSC. [Online] Available at: <https://ucsc.cmb.ac.lk/optical-character-recognition-system-sinhala/> [Accessed 7 April 2020].

[31] S. Rice, F. Jenkins and T. Nartker, "The Fifth Annual Test of OCR Accuracy", Information Science Research Institute University of Nevada, Las Vegas, 1996.

[32] Tess4j.sourceforge.net. n.d. Tess4j - JNA Wrapper For Tesseract. [Online] Available at: <http://tess4j.sourceforge.net/> [Accessed 10 April 2020].

APPENDIX A - ANALYSIS OF INPUT SOURCE

Word	Occurance
.	16
ලෝක	9
සෞඛ්‍ය	8
වන	6
ඔහු	6
බව	5
කටයුතු	5
බවට	4
කරමින්	4
එක්සත්	4
සිටියේ	3
සහ	3
සඳහා	3
සංවිධානයේ	3
සංවිධානයට	3
සංවිධානය	3
ලබා	3
යුතු	3
බවයි	3
ප්‍රකාශ	3
පවා	3
දෙමින්	3
ට්‍රම්ප්	3
චෝදනා	3
චීනය	3
කියා	3
කලේය	3
කල	3
ඇති	3
ආචාර්ය	3
අධ්‍යක්ෂ	3
අතර	3
සිටියේය	2
සම්බන්ධයෙන්	2
සමග	2
වෛරසය	2
වෛරස	2
වූහාන්	2
වැරදි	2
වැඩි	2
වාර්ථා	2
වසංගතය	2

වගකීම්	2
ලොව	2
රිපබ්ලිකන්	2
මහා	2
බවත්	2
ප්‍රකාශයක්	2
පිළිතුරු	2
පසුගිය	2
පවසා	2
නොමග	2
දත්ත	2
තම	2
ජනපද	2
චීනයේ	2
චීනයට	2
චීන	2
ගෝලීය	2
කිරීම	2
කරයි	2
කර	2
කපා	2
ඔවුන්	2
එල්ල	2
ඉඩ	2
අරමුදල්	2
හේතුවෙන්	1
හෙළිකරන	1
හිදී	1
හි	1
හැරියා	1
හාන්ස්	1
හරින	1
හමුවකදී	1
ෆොක්ස්	1
ස්වභාවධර්මය	1
සෙනේට්	1
සුළු	1
සිතා	1
සිටිමු	1
සිටින	1
සහයෝගයෙන්	1
සලකුණක්	1
සමස්ත	1
සමත්	1
සභික	1
සපයන	1

සතියේ	1
සටනක්	1
සටන	1
සංඛ්‍යා	1
ව්‍යාප්තිය	1
ව්‍යාප්ත	1
වෛරසයේ	1
වෙමින්	1
වෙත	1
වූත්	1
වූ	1
වීමට	1
වීම	1
වී	1
විසින්	1
විශේෂඥයින්	1
විවේචනයට	1
විල්බර්	1
වාසිදායක	1
වසංගතයේ	1
වසංගතයට	1
වල	1
වනදා	1
වටයකින්	1
ලෝකය	1
ලේඛන	1
ලේකම්	1
ලෙස	1
ලඟ	1
රොස්	1
රුබියෝ	1
රූපවාහිනී	1
රැකියා	1
රාජ්‍ය	1
රටවල්	1
යුරෝපීය	1
යුධ	1
යුද්ධය	1
යැවීමට	1
යැයි	1
යවන	1
යලි	1
යයි	1
මොහොතකින්	1
මෙය	1
මෙම	1

මෙන්ම	1
මුල්	1
මියගොස්	1
මාසයේදී	1
මාසයේ	1
මාස	1
මාර්කෝ	1
මාරාන්තික	1
මාධ්‍ය	1
මරණ	1
මන්දිරයේ	1
මත	1
භාවයට	1
බෘස්	1
බිස්නස්	1
බැලිය	1
බලවත්	1
බලයක්	1
බලධාරීන්	1
බල	1
ප්‍රමාණය	1
ප්‍රතිවිරුද්ධ	1
ප්‍රතිචාර	1
ප්‍රජාව	1
ප්‍රකාශයට	1
පූර්ණ	1
පුරාවට	1
පුරා	1
පිළිබඳව	1
පිහිටා	1
පිරිසක්	1
පිටත්කල	1
පැහැර	1
පැහැදිලිය	1
පැවැත්මේ	1
පැවැති	1
පැනයකට	1
පැතිරීම	1
පැතිරයාමට	1
පැතිරයාම	1
පාලනය	1
පවතී	1
පලකලේය	1
පරිපාලනය	1
පමණක්	1
පත්කිරීමට	1

පක්ෂපාතීව	1
නොවේ	1
නොවන	1
නොපෙනෙන	1
නොපමාව	1
නිවුස්	1
නිල	1
නියෝජිතයන්ද	1
නියෝජිතයන්ගේ	1
නිදහස්	1
නැත්තාවූත්	1
නාලිකාවේ	1
නව	1
නගර	1
ධවල	1
දෙසැම්බර්	1
දෙසට	1
දූවිලි	1
දුරටත්	1
දුන්	1
දියුණුව	1
දිගු	1
දැවැන්ත	1
දැමූ	1
දැමීමට	1
දැන්	1
දැඩි	1
දා	1
දර්ශන	1
දක්වමින්	1
තුනකට	1
තීරණාත්මක	1
තිරයෙන්	1
තිබේ	1
තිබෙන්නා	1
තාක්ෂණික	1
තවමත්	1
තවදුරටත්	1
තවත්	1
තර්ජනය	1
තමන්ට	1
ඩොනල්ඩ්	1
ට්‍රම්ප්ගේ	1
ටෙඩ්‍රොස්	1
ජොෂ්ඨ	1
ජනාධිපතිවරණයක්	1

ජනාධිපති	1
ජනවාරි	1
ජනරාල්වරයාගේ	1
ජනරාල්	1
ජනපදයට	1
ජනපදය	1
චෝදනාවන්ට	1
චෝදනාවන්ගෙන්	1
චීනයෙන්	1
ග්‍රහණය	1
ගොඩනගාගත්	1
ගෙබ්‍රෙයිසස්	1
ගෙනවිත්	1
ගිලිහෙන	1
ගැනීමට	1
ගැනීම	1
ගැන	1
ක්ලූජ්	1
ක්‍රෝධයෙන්	1
ක්‍රියාත්මක	1
කොවිඩ්-19	1
කොවිඩ්	1
කොරෝනා	1
කොන්සර්වේටිව්	1
කුඩා	1
කිරීමට	1
කාලයේ	1
කාලයක්	1
කාලය	1
කලේ	1
කලාපයේ	1
කලත්	1
කරන	1
කණ්ඩායමක්	1
කට	1
එහි	1
එසේම	1
එරෙහිව	1
එරෙහි	1
එම	1
එනම්	1
එක්	1
උපදේශක	1
උදව්	1
උග්‍ර	1
ඉල්ලා	1

ඉලක්ක	1
ඉදිරියට	1
ඉටු	1
ඇසට	1
ඇමෙරිකාව	1
ඇමෙරිකා	1
ඇදීයමින්	1
ඇද	1
ඇගිල්ල	1
ආරම්භ	1
ආධිපත්‍ය	1
අළු	1
අස්විය	1
අසරණ	1
අවස්ථාවේ	1
අවස්ථාවක	1
අවස්ථා	1
අවසානයක්	1
අවබෝධ	1
අවධියක	1
අරගලයයි	1
අයිලවර්ඬ	1
අප්‍රේල්	1
අපි	1
අධික	1
අද්නාම්	1
අතරම	1
අත	1
40000	1
3335	1
2020	1
19	1
8	1
7	1

APPENDIX B – CONFUSION GROUPS

්	ි	ඹ	ඹ
ු	ූ	ශ්‍ර	ෂ්‍ර
ය	ස	හ	ග
ප	ෂ	ග	ශ
ස	ඝ	හ	භ
ට	ම	ඩ	ධ
ව	ච	උ	ල
න	ත	ඉ	ඊ
ක	න	එ	ළ
ක	ත	එ	ඵ
රු	ළු	ඵ	ළ
රූ	ළූ	බ	ඛ
ඹ	ඔ	බි	බී
ම	ඹ	ජ	ඡ

APPENDIX C – SINHALA UNICODE CHART

Position	Decimal	Name	Appearance
0x0D82	3458	SINHALA SIGN ANUSVARAYA	◌ං
0x0D83	3459	SINHALA SIGN VISARGAYA	◌ඃ
0x0D85	3461	SINHALA LETTER AYANNA	අ
0x0D86	3462	SINHALA LETTER AAYANNA	ආ
0x0D87	3463	SINHALA LETTER AEYANNA	ඇ
0x0D88	3464	SINHALA LETTER AEEYANNA	ඈ
0x0D89	3465	SINHALA LETTER IYANNA	ඉ
0x0D8A	3466	SINHALA LETTER IIYANNA	ඊ
0x0D8B	3467	SINHALA LETTER UYANNA	උ
0x0D8C	3468	SINHALA LETTER UUYANNA	ඌ
0x0D8D	3469	SINHALA LETTER IRUYANNA	ඍ
0x0D8E	3470	SINHALA LETTER IRUUYANNA	ඎ
0x0D8F	3471	SINHALA LETTER ILUYANNA	ඏ
0x0D90	3472	SINHALA LETTER ILUUYANNA	ඐ
0x0D91	3473	SINHALA LETTER EYANNA	එ
0x0D92	3474	SINHALA LETTER EEYANNA	ඒ
0x0D93	3475	SINHALA LETTER AIYANNA	ඓ
0x0D94	3476	SINHALA LETTER OYANNA	ඔ
0x0D95	3477	SINHALA LETTER OOYANNA	ඕ

0x0D96	3478	SINHALA LETTER AUYANNA	ඖ
0x0D9A	3482	SINHALA LETTER ALPAPRAANA KAYANNA	ක
0x0D9B	3483	SINHALA LETTER MAHAAPRAANA KAYANNA	ඛ
0x0D9C	3484	SINHALA LETTER ALPAPRAANA GAYANNA	ග
0x0D9D	3485	SINHALA LETTER MAHAAPRAANA GAYANNA	ඝ
0x0D9E	3486	SINHALA LETTER KANTAJA NAASIKYAYA	ඞ
0x0D9F	3487	SINHALA LETTER SANYAKA GAYANNA	ඟ
0x0DA0	3488	SINHALA LETTER ALPAPRAANA CAYANNA	ච
0x0DA1	3489	SINHALA LETTER MAHAAPRAANA CAYANNA	ඡ
0x0DA2	3490	SINHALA LETTER ALPAPRAANA JAYANNA	ජ
0x0DA3	3491	SINHALA LETTER MAHAAPRAANA JAYANNA	ඣ
0x0DA4	3492	SINHALA LETTER TAALUJA NAASIKYAYA	ඤ
0x0DA5	3493	SINHALA LETTER TAALUJA SANYOOGA NAAKSIKYAYA	ඥ
0x0DA6	3494	SINHALA LETTER SANYAKA JAYANNA	ඦ
0x0DA7	3495	SINHALA LETTER ALPAPRAANA TTAYANNA	ට
0x0DA8	3496	SINHALA LETTER MAHAAPRAANA TTAYANNA	ඨ
0x0DA9	3497	SINHALA LETTER ALPAPRAANA DDAYANNA	ඩ
0x0DAA	3498	SINHALA LETTER MAHAAPRAANA DDAYANNA	ඪ
0x0DAB	3499	SINHALA LETTER MUURDHAJA NAYANNA	ණ
0x0DAC	3500	SINHALA LETTER SANYAKA DDAYANNA	ඬ
0x0DAD	3501	SINHALA LETTER ALPAPRAANA TAYANNA	ත

0x0DAE	3502	SINHALA LETTER MAHAAPRAANA TAYANNA	ථ
0x0DAF	3503	SINHALA LETTER ALPAPRAANA DAYANNA	ද
0x0DB0	3504	SINHALA LETTER MAHAAPRAANA DAYANNA	ධ
0x0DB1	3505	SINHALA LETTER DANTAJA NAYANNA	න
0x0DB3	3507	SINHALA LETTER SANYAKA DAYANNA	ඳ
0x0DB4	3508	SINHALA LETTER ALPAPRAANA PAYANNA	ප
0x0DB5	3509	SINHALA LETTER MAHAAPRAANA PAYANNA	ඵ
0x0DB6	3510	SINHALA LETTER ALPAPRAANA BAYANNA	බ
0x0DB7	3511	SINHALA LETTER MAHAAPRAANA BAYANNA	භ
0x0DB8	3512	SINHALA LETTER MAYANNA	ම
0x0DB9	3513	SINHALA LETTER AMBA BAYANNA	ඹ
0x0DBA	3514	SINHALA LETTER YAYANNA	ය
0x0DBB	3515	SINHALA LETTER RAYANNA	ර
0x0DBD	3517	SINHALA LETTER DANTAJA LAYANNA	ල
0x0DC0	3520	SINHALA LETTER VAYANNA	ව
0x0DC1	3521	SINHALA LETTER TAALUJA SAYANNA	ශ
0x0DC2	3522	SINHALA LETTER MUURDHAJA SAYANNA	ෂ
0x0DC3	3523	SINHALA LETTER DANTAJA SAYANNA	ස
0x0DC4	3524	SINHALA LETTER HAYANNA	හ
0x0DC5	3525	SINHALA LETTER MUURDHAJA LAYANNA	ළ
0x0DC6	3526	SINHALA LETTER FAYANNA	ෆ

0x0DCA	3530	SINHALA SIGN AL-LAKUNA	◌්
0x0DCF	3535	SINHALA VOWEL SIGN AELA-PILLA	◌ා
0x0DD0	3536	SINHALA VOWEL SIGN KETTI AEDA-PILLA	◌ැ
0x0DD1	3537	SINHALA VOWEL SIGN DIGA AEDA-PILLA	◌ෑ
0x0DD2	3538	SINHALA VOWEL SIGN KETTI IS-PILLA	◌ි
0x0DD3	3539	SINHALA VOWEL SIGN DIGA IS-PILLA	◌ී
0x0DD4	3540	SINHALA VOWEL SIGN KETTI PAA-PILLA	◌ු
0x0DD6	3542	SINHALA VOWEL SIGN DIGA PAA-PILLA	◌ූ
0x0DD8	3544	SINHALA VOWEL SIGN GAETTA-PILLA	◌ෘ
0x0DD9	3545	SINHALA VOWEL SIGN KOMBUVA	ෙ◌
0x0DDA	3546	SINHALA VOWEL SIGN DIGA KOMBUVA	ෙ◌්
0x0DDB	3547	SINHALA VOWEL SIGN KOMBU DEKA	ෙෙ◌
0x0DDC	3548	SINHALA VOWEL SIGN KOMBUVA HAA AELA-PILLA	ෙ◌ා
0x0DDD	3549	SINHALA VOWEL SIGN KOMBUVA HAA DIGA AELA-PILLA	ෙ◌ා්
0x0DDE	3550	SINHALA VOWEL SIGN KOMBUVA HAA GAYANUKITTA	ෙ◌ෟ
0x0DDF	3551	SINHALA VOWEL SIGN GAYANUKITTA	◌ෟ
0x0DF2	3570	SINHALA VOWEL SIGN DIGA GAETTA-PILLA	◌ෲ
0x0DF3	3571	SINHALA VOWEL SIGN DIGA GAYANUKITTA	◌ෳ
0x0DF4	3572	SINHALA PUNCTUATION KUNDDALIYA	෴

APPENDIX D – SOURCE CODE

Normalization Engine

```
public String applyVowelNormalizationRules(String wordString) {


  String modifiedWordString = wordString;

  /*
   Sinhala Code point range in decimal 3456-3583 *
   */
  // Start Replace the Vowels with modifies to the proper character
  if (wordString.charAt(0) == 3461 && wordString.charAt(1) == 3535) { //        SINHALA LETTER
AAYANNA

      modifiedWordString = wordString.replace(Character.toString(wordString.charAt(0)),
Character.toString((char) 3462));
      StringBuilder tempWordString1 = new StringBuilder(modifiedWordString);
      tempWordString1.deleteCharAt(1);
      modifiedWordString = tempWordString1.toString();

  } else if (wordString.charAt(0) == 3461 && wordString.charAt(1) == 3536) { //    SINHALA LETTER
AEYANNA

      modifiedWordString = wordString.replace(Character.toString(wordString.charAt(0)),
Character.toString((char) 3463));
      StringBuilder tempWordString2 = new StringBuilder(modifiedWordString);
      tempWordString2.deleteCharAt(1);
      modifiedWordString = tempWordString2.toString();

  } else if (wordString.charAt(0) == 3461 && wordString.charAt(1) == 3537) { // SINHALA LETTER
AEEYANNA

      modifiedWordString = wordString.replace(Character.toString(wordString.charAt(0)),
Character.toString((char) 3464));
      StringBuilder tempWordString3 = new StringBuilder(modifiedWordString);
      tempWordString3.deleteCharAt(1);
      modifiedWordString = tempWordString3.toString();

  } else if (wordString.charAt(0) == 3467 && wordString.charAt(1) == 3551) { //  SINHALA LETTER
UUYANNA

      modifiedWordString = wordString.replace(Character.toString(wordString.charAt(0)),
Character.toString((char) 3468));
      StringBuilder tempWordString4 = new StringBuilder(modifiedWordString);
      tempWordString4.deleteCharAt(1);
      modifiedWordString = tempWordString4.toString();

  } else if (wordString.charAt(0) == 3545 && wordString.charAt(1) == 3473) { // SINHALA LETTER
AIYANNA

      modifiedWordString = wordString.replace(Character.toString(wordString.charAt(1)),
Character.toString((char) 3475));
      StringBuilder tempWordString5 = new StringBuilder(modifiedWordString);
      tempWordString5.deleteCharAt(0);
      modifiedWordString = tempWordString5.toString();

  } else if (wordString.charAt(0) == 3476 && wordString.charAt(1) == 3551) { // SINHALA LETTER
```

```
        modifiedWordString = wordString.replace(Character.toString(wordString.charAt(0)),
Character.toString((char) 3478));
        StringBuilder tempWordString6 = new StringBuilder(modifiedWordString);
        tempWordString6.deleteCharAt(1);
        modifiedWordString = tempWordString6.toString();
    }


    return modifiedWordString;
}




public String applyConsonantNormalizationRules(String innerText) {

    // TODO : Add rule to correct kroo
    int lengthOfString = innerText.length();

    for (int currentPos = 0; currentPos < lengthOfString;) {

        if (innerText.charAt(currentPos) == 3545) { // SINHALA VOWEL SIGN KOMBUVA

            if (currentPos + 3 <= lengthOfString) { // String of 4 chars starting from kombuwa
                if ((innerText.charAt(currentPos + 1) >= 3482 && innerText.charAt(currentPos + 1) <= 3526)
                        && innerText.charAt(currentPos + 2) == 3535 && innerText.charAt(currentPos + 3) == 3530)
{ // kombuwa Consonant alapilla hal kireema

                    innerText = insertCharAt(innerText, (char) 3549, currentPos + 3);
                    innerText = deleteCharAt(innerText, currentPos);
                    innerText = deleteCharAt(innerText, currentPos + 1);
                    innerText = deleteCharAt(innerText, currentPos + 2);

                    lengthOfString = innerText.length();
                    currentPos += 2;

                } else if (currentPos + 2 <= lengthOfString) { // string of 3 chars starting from kombuwa

                    if ((innerText.charAt(currentPos + 1) >= 3482 && innerText.charAt(currentPos + 1) <= 3526)
                            && innerText.charAt(currentPos + 2) == 3535) { // kombuwa consonant and adapilla

                        innerText = insertCharAt(innerText, (char) 3548, currentPos + 2);
                        innerText = deleteCharAt(innerText, currentPos);
                        innerText = deleteCharAt(innerText, currentPos + 2);

                        lengthOfString = innerText.length();
                        currentPos += 2;

                    } else if ((innerText.charAt(currentPos + 1) >= 3482 && innerText.charAt(currentPos + 1) <=
3526)
                            && (innerText.charAt(currentPos + 2) == 3551 || innerText.charAt(currentPos + 2) ==
3571)) { // kombuwa consonant and gayanu kiththa

                        innerText = insertCharAt(innerText, (char) 3550, currentPos + 2);
                        innerText = deleteCharAt(innerText, currentPos);
                        innerText = deleteCharAt(innerText, currentPos + 2);

                        lengthOfString = innerText.length();
```

```
                currentPos += 2;

            } else if (innerText.charAt(currentPos + 1) == 3545
                && (innerText.charAt(currentPos + 2) >= 3482 && innerText.charAt(currentPos + 2) <=
3526)) { // kombuwa combuwa and consonant

                innerText = insertCharAt(innerText, (char) 3547, currentPos + 3);
                innerText = deleteCharAt(innerText, currentPos);
                innerText = deleteCharAt(innerText, currentPos);

                lengthOfString = innerText.length();
                currentPos += 2;

            } else if ((innerText.charAt(currentPos + 1) >= 3482 && innerText.charAt(currentPos + 1) <=
3526)
                && (innerText.charAt(currentPos + 2) == 3551 || innerText.charAt(currentPos + 2) ==
3530)) { // kombuwa consonant and hal kireema

                innerText = insertCharAt(innerText, (char) 3546, currentPos + 2);
                innerText = deleteCharAt(innerText, currentPos);
                innerText = deleteCharAt(innerText, currentPos + 2);

                lengthOfString = innerText.length();
                currentPos += 2;

            } else if (innerText.charAt(currentPos + 1) >= 3482 && innerText.charAt(currentPos + 1) <=
3526) { // kombuwa and consonant

                innerText = swapCharacters(innerText, currentPos, currentPos + 1);
                currentPos += 2;

            } else {

                currentPos++;

            }

        } else if (currentPos + 1 <= lengthOfString) { // string of 2 chars tarting from kombuwa

            if (innerText.charAt(currentPos + 1) >= 3482 && innerText.charAt(currentPos + 1) <= 3526) {
// kombuwa and consonant

                innerText = swapCharacters(innerText, currentPos, currentPos + 1);
                currentPos += 2;

            } else {
                currentPos++;
            }

        } else {

            currentPos++; // TODO implement Later
        }

    } else if (currentPos + 1 <= lengthOfString) { //kombuwa and consonant at the end of a word

        if (innerText.charAt(currentPos + 1) >= 3482 && innerText.charAt(currentPos + 1) <= 3526) { //
kombuwa and consonant

            innerText = swapCharacters(innerText, currentPos, currentPos + 1);
            currentPos += 2;
```

```
            } else {
                currentPos++;
            }

        } else {

            currentPos++;
        }

    } else {

        currentPos++;
    }

}

return innerText;
}

public String applySpecialConsonantRules(String innerText) {

    int lengthOfString = innerText.length();
    char[] charSet = {3482, 3484, 3495, 3497, 3501, 3508, 3510};

    for (int currentPos = 0; currentPos < lengthOfString;) {
        if (currentPos + 5 <= lengthOfString) { // string of 6 chars starting from a consonant
            if (containsChar(innerText.charAt(currentPos), charSet)) { // starting character is a consonant from
the charSet
                if (innerText.charAt(currentPos + 1) == 3546 && innerText.charAt(currentPos + 2) == 8205
                        && innerText.charAt(currentPos + 3) == 3515 && innerText.charAt(currentPos + 4) == 3535
                        && innerText.charAt(currentPos + 5) == 3530) { // Sinhala Char Kroo
                    innerText = swapCharacters(innerText, currentPos + 1, currentPos + 5);
                    innerText = replaceCharAt(innerText, currentPos + 4, 3549);
                    innerText = deleteCharAt(innerText, currentPos + 5);
                    lengthOfString = innerText.length();
                    currentPos = currentPos + 4;

                } else {
                    currentPos++;
                }
            } else {
                currentPos++;
            }

        } else {
            currentPos++;
        }

    }
    return innerText;
}
```

Dictionary Match

```java
public List<String> readFromWordLexicon() {

    File textFile = new File(".\\word_list.txt");
    List<String> wordList = new ArrayList<>();

    try {
      BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(textFile),
"UTF-8"));
      String st;
      while ((st = br.readLine()) != null) {
        wordList.add(st);
//          log.info(st);
      }
      br.close();
    } catch (FileNotFoundException ex) {
      log.error(ex);
    } catch (IOException ex) {
      log.error(ex);
    }

    return wordList;
  }


  public boolean findDictionaryMatch(String word, List<String> wordList) {
    return wordList.contains(word);

  }
```

Confusion Pairs

```java
  public String applyConfusionRules(String word) {

    String tempWord;
    if (this.findDictionaryMatch(word, wordList)) {
      return word;
    } else {
      for (int i = 0; i < confusionRuleArray.length; i++) {
        if (word.contains(confusionRuleArray[i][0])) { //matching the confusion rule R->L
          tempWord = word.replaceFirst(confusionRuleArray[i][0], confusionRuleArray[i][1]);
          if (findDictionaryMatch(tempWord, wordList)) {
            return tempWord;
          } else {
            return word;
          }

//            log.info("confusion rule found" + confusionRuleArray[i][0] + " " + confusionRuleArray[i][1] + "  "
+ word);
//            log.info("replaced WOrd : " + word.replaceFirst(confusionRuleArray[i][0],
confusionRuleArray[i][1]));
        } else if (word.contains(confusionRuleArray[i][1])) {
          tempWord = word.replaceFirst(confusionRuleArray[i][1], confusionRuleArray[i][0]);
          if (findDictionaryMatch(tempWord, wordList)) {
            return tempWord;
```

```java
            } else {
                return word;
            }
        }
    }
//          return word;
    }

    return word;
}
```

Invoking the Tesseract OCR engine to obtain Output

```java
public String performOcr(String filePath) {

    String hocrOutput = null;
    File imageFile = new File(filePath);

    Tesseract hocrInstance = new Tesseract();// JNA Interface Mapping
    hocrInstance.setLanguage("sin");
    hocrInstance.setHocr(true);
    hocrInstance.setDatapath(".");

    try {
        hocrOutput = hocrInstance.doOCR(imageFile);

    } catch (TesseractException e) {
        System.err.println(e.getMessage());
    }

    return hocrOutput;
}
```

Invoking the corrections during post-processing

```java
public void runOcrErrorCorrectionEngine(File ocrOutputString) {

        String innerSpanContent;
        String innerText;
        String normalizedInnerText;

        try {
            Document inputHtmlDoc = Jsoup.parse(ocrOutputString, "UTF-8");
            PrintWriter writer = new PrintWriter(ocrOutputString, "UTF-8");
            wordList = this.readFromWordLexicon();
            confusionRuleArray = this.readConfusionPairs();

            //Choose each word in the output
            for (Element span : inputHtmlDoc.select("span.ocrx_word")) {

                innerSpanContent = span.html();
                innerText = span.text();

                normalizedInnerText = applyVowelNormalizationRules(innerText); // Apply Vowel Normalization
rules
                normalizedInnerText = applyConsonantNormalizationRules(normalizedInnerText); // Apply
Consonant Normalization rules
```

```
            normalizedInnerText = applySpecialConsonantRules(normalizedInnerText);
            log.info("before confusion :" + normalizedInnerText);
            normalizedInnerText = applyConfusionRules(normalizedInnerText); //Apply confusion rules
            log.info("after confusion :" + normalizedInnerText);
            innerSpanContent = innerSpanContent.replace(innerText, normalizedInnerText);

//              log.info(innerText + " : " + this.findDictionaryMatch(normalizedInnerText, wordList));
            span.html(innerSpanContent);

          }

          writer.write(inputHtmlDoc.html());
          writer.flush();
          writer.close();

        } catch (IOException ex) {
          log.error(ex.getMessage(), ex);
        }

    }
```