A Data-Driven Approach to Checking and Correcting Spelling Errors in Sinhala

Asanka Wasala, Ruvan Weerasinghe, Randil Pushpananda, Chamila Liyanage and Eranga Jayalatharachchi

Abstract—**In this paper we describe the construction of a spell checker for Sinhala, the language spoken by the majority in Sri Lanka. Due to its morphological richness, the language is difficult to enumerate completely in a lexicon. The approach described is based on n-gram statistics and is relatively inexpensive to construct without deep linguistic knowledge. This approach is particularly useful as there are very few linguistic resources available for Sinhala at present. The proposed algorithm has been shown to be able to detect and correct many of the common spelling errors of the language. Results show a promising performance achieving an average accuracy of 82%. This technique can also be applied to construct spell checkers for other phonetic languages whose linguistic resources are scarce or non-existent.**

Index Terms—**spell checking, Sinhala, data driven, n-gram**

I. Introduction

Spell checking deals with detecting misspelled words in a written text and possibly assisting users in correcting them with the use of a dictionary or otherwise. Spell checkers are well-known components of word-processing applications. In addition, spell checkers are widely used in other applications such as Optical Character Recognition (OCR) systems, Automatic Speech Recognition (ASR) systems, Computer Aided Language Learning (CALL) Software, Machine Translation (MT) systems and Text-to-Speech (TTS) systems [1] [2]. The history of automatic spelling correction goes back to the 1960s [3]. Even after decades of extensive research and development, the effectiveness of spell checkers remains a challenge today.

Common spelling mistakes can be classified into two broad categories: 1) non-word errors, where the word itself is invalid (i.e. not present in a valid lexicon) and 2) real-word errors, where the word is valid yet inappropriate in the context [3] [1] [2]. Based on above categorisation, the task of spelling correction can be classified into two approaches: isolated-word correction and context-sensitive error correction. Real-word errors are usually recognized and corrected using non-context-sensitive spelling error correction approaches [3]. Context-sensitive spelling error correction is more complex and requires advanced statistical and Natural Language Processing (NLP) techniques.

In this paper, we focus on detecting and correcting non-word errors, especially to address a prominent issue prevalent in written Sinhala, casually referred to as “na-Na-la-La” dissension. A data-driven algorithm based on n-gram statistics is proposed to solve these spelling problems. In addition, the proposed algorithm is also capable of addressing common spelling errors due to phonetic similarity of letters. At present, there is no published work on Sinhala spell checking. To the best of the our knowledge, this is the first implementation of a spell checker for Sinhala using a data-driven approach.

The rest of this paper is organized as follows: Section II summarizes the related work in this area. Section III gives an overview of the linguistic features related to Sinhala spelling and describes the core spell checking algorithm implemented while Section IV presents an evaluation of the current system. Section V discusses the main findings of the research. Finally, the paper concludes with a summary of the current research and discusses future research directions.

II. Related Work

Spell Checkers for European languages such as English [3] are well developed. Literature concerning spell checkers in Indic languages such as Assamese [2] Bangla [4] [1] Malayalam [5] Marathi [6] and Tamil [7] are less well developed. However, similar research in several other languages, including Sinhala, is underway and need special attention owing to morphological richness.

Several commercial products [18] of Sinhala spell checkers have been announced in recent years. Work on open-source spell checkers has also shown an increase recently. Hunspell (the spell checker of OpenOffice.org, Mozilla Firefox & Thunderbird, Google Chrome, Mac OS X and Opera [19]) has support for Sinhala on OpenOffice.org through extensions [20]. A dictionary-based spell-checker is available for Mozilla Firefox as an add-on [21].

Manuscript received March 30, 2010. Accepted September 12, 2010.
This research was partially supported by the PAN Localization Project, (http://www.PANL10n.net) a grant from the International Development Research Center (IDRC), Ottawa, Canada, administered through the Center for Research in Urdu Language Processing, National University of Computer and Emerging Sciences, Pakistan.

R. A. Wasala was with Language Technology Research Laboratory, University of Colombo School of Computing, 35, Reid Avenue, Colombo 07, Sri Lanka. He is now with the Localisation Research Centre, Department of Computer Science and Information Systems, University of Limerick, Limerick, Ireland. (e-mail: Asanka.Wasala@ul.ie).
A.R. Weerasinghe and D.E. Jayalatharachchi are with the University of Colombo School of Computing, 35, Reid Avenue, Colombo 07, Sri Lanka. (e-mail: arw@ucsc.cmb.ac.lk, dej@ucsc.cmb.ac.lk).
R. Pushpananda and C. Liyanage are with the Language Technology Research Laboratory, University of Colombo School of Computing, 35, Reid Avenue, Colombo 07, Sri Lanka (e-mail: rpn@ucsc.cmb.ac.lk, cml@ucsc.cmb.ac.lk).

Furthermore, there is a Sinhala search-engine [22] that uses a dictionary-based technique to automatically correct spelling mistakes in query strings. Unfortunately, none of these spell-checkers have been systematically assessed or benchmarked.

One of the major issues worth noting in implementing spell checkers in these languages is resource deficiency. Morphological analyzers, tagged corpora and comprehensive lexica are scarce for many languages including Sinhala. Moreover, due to the rich morphological features of these languages, developing entirely rule based systems or integrating with existing open-source spell checkers such as Aspell are arduous tasks. Therefore, the research in Spell checker development in languages such as Sinhala has many unresolved issues.

The problem of spell checking has been addressed using edit distance based approaches, morphology and rule based approaches, dictionary-lookup and reversed dictionary lookup techniques, n-gram analysis, probabilistic methods, and neural nets [3] [9] [1] [4]. Of these, morphology based approaches [5] [6] [7] [2] and reverse dictionary lookup techniques [1] [4] [5] are the most popular ones used in Indic languages, most of which have to deal with rich morphology. Probabilistic or data driven approaches are scarcely reported due to the lack of resources such as corpora, although n-gram based approaches are shown to be effective in addressing spelling errors in other languages [9] [3].

III. Methodology

A rigorous linguistic analysis and a literature survey were carried out to investigate the factors leading to most common non-word spelling errors in Sinhala. Based on this linguistic analysis and a thorough analysis of a text corpus, an algorithm is proposed to detect and correct spelling mistakes typically found in Sinhala writing.

A. *Linguistic Analysis*

Sinhala is diglossic; its spoken form is different from its written form. Sinhala orthography consists of 60 graphemes and an estimated 40 phonemes [10]. The study revealed that most of the non-word spelling errors occur due to three factors: 1) the phonetic similarity of Sinhala characters, 2) irregular correspondence between Sinhala graphemes and phonemes, and, 3) the lack of knowledge of spelling rules.

In Sinhala, non-word spelling mistakes are largely due to the fact that several graphemes correspond to a single phoneme [10]. The most prominent cases are elaborated in the following sections.

1) *The Pronunciation and Orthography of Aspirated and Unaspirated Consonants*

According to Disanayaka [11] the Sinhala writing system contains 10 graphemes for representing aspirated consonants (ඛ /k^h/, ඝ /g^h/, ඡ /tʃ^h/, ඣ /dʒ^h/, ඨ /ʈ^h/,ඪ /ɖ^h/, ථ /t̪/, ධ /d̪^{h}/, ඵ /p^h/, භ /b^h/) and 10 graphemes for representing unaspirated consonants (ක /k/, ග /g/, ච /tʃ/, ජ /dʒ/, ට /ʈ/, ඩ /ɖ/, ත /t̪/, ද /d̪/, ප /p/, බ /b/).

The aspirated consonants occur in words borrowed from *Sanskrit* or *Pali* languages. However, they are generally not pronounced differently from their unaspirated counterparts [11] [10] [12]. This particular gap between the written language and the spoken language has led to some common spelling errors in Sinhala.

Among the letters representing aspirated consonants, the letters ‘ඣ’ /dʒ^h/, ‘ඡ’ /tʃ^h/ and ‘ඵ’ /p^h/ are rarely used, while the rest are frequent. Furthermore, it can be seen that these aspirated letters can appear at the beginning, middle or end of a word. Hence, it is difficult to establish linguistic rules for the proper usage of unaspirated and aspirated letters in Sinhala writing.

2) *Retroflex and Dental Letter Confusion: The ‘na-Na-la-La’ Dissension*

The most common spelling errors in Sinhala are due to the retroflex and dental letter confusion. In spoken Sinhala, several graphemes that represent corresponding retroflex consonants are actually pronounced in an intermediate alveolar-like position. The graphemes ‘ණ’ and ‘ළ’ represent the retroflex nasal /ɳ/ and the retroflex lateral /ɭ/ respectively. But they are pronounced in the same manner as their respective alveolar counterparts ‘න’-/n/ and ‘ල’-/l/ [12] [10]. When pronouncing the above consonants, not much attention is paid to the distinction of the place of articulation (i.e. all of them are pronounced as alveolar sounds), but the distinction of retroflex and dental letters (though pronounced as alveolar consonants) is stressed in the writing system. This confusion inevitably leads to spelling errors.

In the literature, these errors are commonly known as “na-Na-la-La” (/na/-/nə/-/la/-/lə/) Dissention (‘න-ණ - ල-ළ’ errors). Linguists believe that clear guidelines or a mechanism had been present to describe the correct usage of retroflex-dental letters until the end of 13th century [13]. However, due to various reasons, these guidelines no longer exist [13].

By analyzing the language, several rules can be defined to minimize the confusion between retroflex and dental letter usage. Some rules can be defined by considering the phonological transformation rules applied for words derived from other languages. In addition, some more rules can be derived by analyzing the usage and the context of retroflex and dental letters (i.e. ‘න, ණ, ල, ළ’). See Sections 1 - 4 of the Appendix A for linguistic rules concerning the use of the above retroflex and dental letters.

Rules of the former type are extremely complex. A layman lacks the requisite linguistic knowledge to apply such rules to decide whether to use retroflex or the dental letter in spelling a given a word.

For example, see the rule described in Appendix A 1.2:

1. Intervocalic Sanskrit and Pali ණ /ɳ/ does not get evolved [14].

   Sanskrit ‘ෂ්ණ’ /ʂɳ/ > Pali ‘ණ්හ’ /ɳh/ > Sinhala ණ /ɳ/ [14]

Example: උෂ්ණ /uʂɳə/ > උණ්හ /uɳhə/ > උණු /uɳu/, උණ /uɳə/

All words used in this example are used in modern Sinhala writing, yet, a layman would not normally know which words are borrowed 'as is' from foreign languages (such as *Sanskrit* or *Pali*) and which words are of the derived kind. Therefore, without this knowledge, one might fail to apply the above rule and use the dental letter 'න' instead of retroflex letter 'ණ' (or vice versa) in certain words causing spelling errors.

3) The Pronunciation and Orthography of the Retroflex and Palatal Sibilants

In Sinhala, the grapheme 'ෂ' that represents the retroflex sibilant /ʂ/, is pronounced as the palatal sibilant 'ශ'-/ʃ/. As with the case of graphemes for representing aspirated sounds, the above two graphemes were also borrowed from Sanskrit. Here too, though the distinction of the place of articulation is not prominent in pronunciation, the correct grapheme has to be used in writing. It is possible to define some linguistic rules on the correct use of above graphemes (See Sections 5 and 6 of the Appendix A). It should be noted, however, that not all cases are covered by such linguistic rules. Hence, it can be seen that the confusion between above two graphemes has lead to some spelling errors.

B. Error Detection and Correction Methodology

The algorithm for spelling error detection and correction is described below. It is based on n-gram statistics computed from the *UCSC Sinhala Corpus* [23]. The computed word unigram frequencies and the syllable bigram and trigram frequencies are effectively utilized in addressing the prominent "na-Na-la-La" dissension as well as other spelling errors described in Section III-A above.

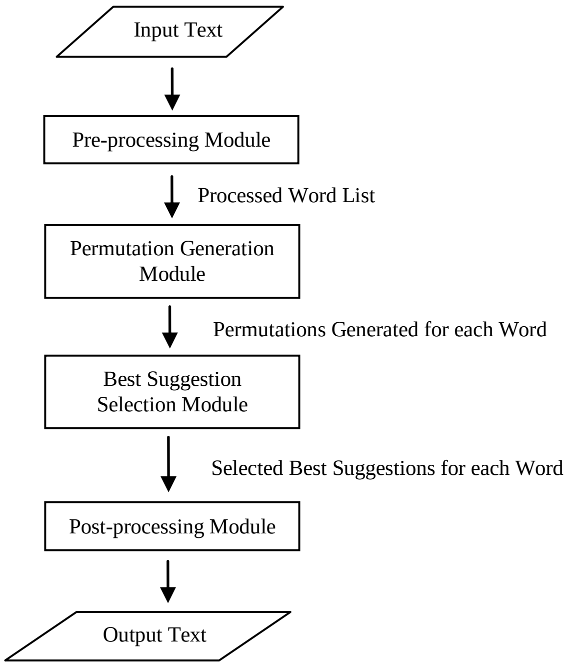

Input Text
Pre-processing Module
Processed Word List
Permutation Generation Module
Permutations Generated for each Word
Best Suggestion Selection Module
Selected Best Suggestions for each Word
Post-processing Module
Output Text

Fig. 1. Core Modules and the Overall Architecture of the Spell Checker

Our algorithm is based on the assumption that the majority of users of the language write using correct spellings. In other words, we assume that the frequency of valid words (i.e. words with correct spelling) appearing in the corpus is higher than the frequency of invalid words. The core modules and the overall architecture of the spell checker are illustrated in Figure 1.

The main algorithm of the Sinhala spell checker is given in Figure 2. Each module has been implemented as a function and the algorithm corresponding to each function is given after the description of each module.

```
ProcessedWordList=PreProcess(InputText)
 for each word w in ProcessedWordList
    PermutationList=GeneratePermutations(w)
    BestSuggestion=SelectBestSuggestion
    (PermutationList, w)
      if BestSuggestion is not equal to w
      then
         SubstitutionList[w]=BestSuggestion
      end if
 end for
OutputText=PostProcess(InputText,
SubstitutionList)
Display(OutputText)
```

Fig. 2. The Main Algorithm of the Spell Checker

1) Pre-Processing Module

The input to the system is Unicode text. In the pre-processing module, the system first tokenizes the text stream and builds a list containing unique Sinhala words found in the text. Each word is then compared with an exception word list. If a word is found to be in the exception word list, it will be removed from the unique word list, hence from further processing. The exception word list contains a list of homophones and valid spelling variants. A total of 1188 words identified mainly from literature [15] are included in this exception list. Several examples for homophones include {කන - /kanə/ - *eat*, කණ - /kanə/- *ear*}, {තන - /t̪anə/ - *breast*, තණ - /t̪anə/- *grass*}, spelling variants include {උළෙල -/ulelə/ - *ceremony*, උලෙළ - /ulelə/ - *ceremony*} and {කුසලතා - /kusələt̪a:/- *skills*, කුශලතා - /kusələt̪a:/ - *skills*}. Homophone disambiguation requires contextual information as well as advanced Natural Language Processing (NLP) techniques and is beyond the scope of this paper. The algorithm in its current form is only capable of processing isolated words. Therefore, homophones and spelling variants are excluded from further processing. Each word in the processed unique list is then passed to the permutation generation module.

Pre-Processing Algorithm

Function `PreProcess` takes the input text as a parameter and returns a list of unique Sinhala words found in the input text but not in the exception word list.

```
PreProcess(InputText)
 TokenizedWordList = Tokenize(InputText)
 for each word w in  TokenizedWordList
    if w is Sinhala and
       w is not in UniqueWordList and
       w is not in ExceptionWordList then
       append w to UniqueWordList
    end if
 end for
 return UniqueWordList
```

Fig. 3. Pre-Processing Algorithm

2) *Permutation Generation Module*

As identified in the Section III-A, phonetic similarity of letters can cause spelling mistakes in a word. Similar sounding groups of letters found in this exercise include {ක,ඛ}=/k/, {ග,ඝ}=/g/, {ච,ඡ}=/tʃ/, {ජ,ඣ}=/dʒ/, {ට,ඨ}=/ʈ/, {ඩ,ඪ}=/ɖ/, {ත,ථ}=/t̪/, {ද,ධ}=/d̪/, {ප,ඵ}=/p/, {බ,භ}=/b/, {න,ණ}=/n/, {ල,ළ}=/l/, {ස,ශ,ෂ}=/s/ or /ʃ/ and {ඥ,ඤ}=/ɲ/.

The permutation-generation module accepts a Sinhala word processed by the pre-processing module and generates permutations by searching the word for above similar sounding letters and substituting them with their corresponding letters that belong to the same group. Among the generated words, there can be words with correct spellings, the given (source) word itself and words with incorrect spellings. For example, given the word ‘සුපතල’ - /supat̪alə/ - *popular*, the module will generate and return a list containing 24 tokens including සුපතල, සුපතළ, සුපථල, සුපථළ, සුඵතල, සුඵතළ, සුඵථල, සුඵථළ, ශුපතල, ශුපතළ, ශුපථල, ශුපථළ, ශුඵතල, ශුඵතළ, ශුඵථල, ශුඵථළ, ෂුපතල, ෂුපතළ, ෂුපථල, ෂුපථළ, ෂුඵතල, ෂුඵතළ, ෂුඵථල and ෂුඵථළ.

Permutation Generation Algorithm

The function `GeneratePermutations` accepts a Sinhala word processed by the `PreProcess` function and returns a list containing all generated permutations. The permutations are generated by searching the word for similar sounding letters and substituting them with corresponding letters that belong to the same group.

```
GeneratePermutations(w)
 SimilarLetterGroups={{ක,ඛ},{ග,ඝ},{ච,ඡ},
 {ජ,ඣ}, {ට,ඨ}, {ඩ,ඪ}, {ත,ථ}, {ද,ධ},
 {ප,ඵ},{බ,භ},{න,ණ},{ල,ළ}, {ස,ශ,ෂ},{ඥ,ඤ}}
 PermutationList=[]
 for each letter l in w
  if l found in  SimilarLetterGroup g
    for each SimilarLetter in g
     w = replace l with SimilarLetter in w
     if w not in PermutationList
       append w to PermutationList
       results = GeneratePermutations(w)
      append results to PermutationList
     end if
    end for
 end if
 return PermutationList
```

Fig. 4. Permutation Generation Algorithm

3) *Best Suggestion Selection Module*

This is the core module of the algorithm. This module involves the detection and correction of spelling errors. The n-gram statistics computed from the UCSC Sinhala Corpus is used in this module. A distinct word list along with word frequencies (word unigram frequencies), syllable trigram frequencies and syllable bigram frequencies have been pre-complied and stored in a database for fast retrieval and efficient processing (See Section V for details of the bigram and trigram counting algorithm).

In the first step, word unigram frequencies obtained from the corpus are used to rank the words generated from the permutation generation module and to choose the best suggestion among the generated words. The word unigram frequency corresponding to each generated word is obtained from the database. The word with the highest frequency is chosen as the best suggestion. If none of the generated words are found in the corpus, i.e. the word unigram frequencies returned zero for all the generated words, syllable trigram and bigram frequencies are used to select the best suggestion in the successive steps. If the generated word consists of more than three syllables, it will be divided into overlapping sequences of three syllables. Then, for each three syllable sequence, the corresponding pre-computed trigram frequencies are obtained from the database and summed up to get an overall score for the generated word. If the summed up trigram frequencies yield zero for a certain word, the word will be divided into repetitive chunks of two syllables and pre-computed syllable bigram frequencies will be summed up to get an overall score for the word. Similarly, if the generated word consists of two syllables, the syllable bigram frequency is used. Generated words are sorted according to the overall score obtained. The word with the highest score is chosen as the best suggestion. Output of this module is the best suggestion for a given word. The functionality of the above module is explained below using examples.

Example #1: Suggestion of the best word using word unigram frequencies.

Input word: කුලුන - /kulunə/ (*column*)

`PermutationList =` කුලුන, කුලුණ, කුළුන, කුළුණ, ඛුලුන, ඛුලුණ, ඛුළුන, ඛුළුණ

Step 1: Obtaining the corresponding word unigram frequencies from the corpus.

කුලුන	2	කුලුණ	1	කුළුන	0	කුළුණ	43
ඛුලුන	0	ඛුලුණ	0	ඛුළුන	0	ඛුළුණ	0

Step 2: Selecting the best suggestion (word with the highest frequency)

Best suggestion: කුළුණ

Example #2: Suggestion of the best word using syllable trigram frequencies.

Input word = ඵැඛිළෙණවා - (a word with incorrect spellings meaning *falter* – the word with the correct spelling is not included in the word unigram list.)

`PermutationList` = පැකිලෙනවා, පැකිලෙණවා, පැකිළෙනවා, පැකිළෙණවා, පැඛිලෙනවා, පැඛිලෙණවා, පැඛිළෙනවා, පැඛිළෙණවා, ඵැකිලෙනවා, ඵැකිලෙණවා, ඵැකිළෙනවා, ඵැකිළෙණවා, ඵැඛිලෙනවා, ඵැඛිලෙණවා, ඵැඛිළෙනවා, ඵැඛිළෙණවා

Overall word score computation method for the word 'ඵැකිලෙනවා' is given in the following:

Step 1: Decomposition of the word into repetitive three syllable sequences.

ඵැකිලෙනවා = ඵැකිලෙ + කිලෙන + ලෙනවා

Step 2: Obtaining the syllable trigram frequencies for above syllable sequences from the database.

ඵැකිලෙ 0 කිලෙන 27 ලෙනවා 43

Step 3: Adding above frequencies to obtain the overall score for the word

ඵැකිලෙනවා = 0+27+43 = 70

Similarly, the word 'පැකිලෙනවා' yields a score of 95 when computed using syllable trigram frequencies. Overall scores will be computed for the other generated words in the same manner.

Words are then sorted according to the computed scores and the word with the highest score is returned as the best word.

The best word suggested for the above input word for instance is: පැකිලෙනවා.

If summed up syllable trigram frequencies yield zero for all generated words, the word will be passed to the bigram computation component. The best word selection using the bigram computation component operates in a similar manner to the trigram computation method explained above.

Example #3: Suggestion of the best word using syllable bigram frequencies.

Input word = ඛදෝඵැනියා - (a word with incorrect spellings meaning firefly – the word with the correct spelling is not included in the word unigram list.)

`Permutation list` = කදෝපැනියා, කදෝපැණියා, කදෝඵැනියා, කදෝඵැණියා, කධෝපැනියා, කධෝපැණියා, කධෝඵැනියා, කධෝඵැණියා, ඛදෝපැනියා, ඛදෝපැණියා, ඛදෝඵැනියා, ඛදෝඵැණියා, ඛධෝපැනියා, ඛධෝපැණියා, ඛධෝඵැනියා, ඛධෝඵැණියා

Overall word score computation method for the word 'ඛදෝඵැනියා' is given in the following:

Step 1: Decomposition of the word into repetitive two syllable sequences.

ඛදෝඵැනියා = ඛදෝ + දෝඵැ + ඵැනි + නියා

Step 2: Obtaining the syllable bigram frequencies for above letter sequences from the database

ඛදෝ 2 දෝඵැ 0 ඵැනි 0 නියා 2630

Step 3: Adding above frequencies to obtain the overall score for the word

ඛදෝඵැනියා = 2+0+0+2630=2632

Similarly, the word 'කදෝපැනියා' yields the highest score of 2875 when computed using syllable bigram frequencies. Hence it is selected as the best suggestion.

As the output of this module, a list (`SubstitutionList`) containing the original words and their corresponding best words is returned (e.g.

`SubstitutionList`['කුලුන'] = 'කුළුණ',

`SubstitutionList`['ඵැකිලෙනවා'] = 'පැකිලෙනවා'

etc).

Best Suggestion Selection Algorithm

The function `SelectBestSuggestion` accepts a generated permutation list and selects the best suggestion from the list based on word unigram, syllable bigram or syllable trigram frequencies.

```
SelectBestSuggestion(PermutationList,
OriginalWord)
## uni-gram comparison
  HighestUnigramFrequency = 0
  BestWord = OriginalWord
  for each word w in PermutationList
    WordUnigramFrequency =
      GetUnigramCountFromDB(w)
    if WordUnigramFrequency >
      HighestUnigramFrequency then
    HighestUnigramFrequency =
      WordUnigramFrequency
    BestWord = w
   end if
  end for
## tri-gram comparison
  if BestWord is equal to OriginalWord
  then
    HighestTrigramScore=0
    BestWord=OriginalWord
    for each word w in PermutationList
      ThreeSyllableChunks=[]
      if length of w > 3 then
        ThreeSyllableChunks =
          DecomposeWordIntoTrigrams(w)
        WordTrigramScore=0
        for each ThreeSyllableChunk in
        ThreeSyllableChunks
          WordTrigramScore =
            WordTrigramScore +
            GetTrigramCountFromDB
            (ThreeSyllableChunk)
        end for
        if WordTrigramScore >
        HighestTrigramScore then
          HighestTrigramScore =
            WordTrigramScore
          BestWord=w
        end if
      end if
    end for
  end if
```

```
## bi-gram comparison
  if BestWord = OriginalWord
    then
    HighestBigramScore = 0
    BestWord = OriginalWord
    for each word w in PermutationList
      TwoSyllableChunks = []
      if length of w > 2 then
        TwoSyllableChunks =
          DecomposeWordIntoBigrams(w)
        WordBigramScore = 0
        for each TwoSyllableChunk in
          TwosyllableChunks
          WordBigramScore =
            WordBigramScore
            + GetBigramCountFromDB
            (TwoSyllableChunk)
        end for
        if WordBigramScore >
          HighestBigramScore then
          HighestBigramScore =
            WordBigramScore
            BestWord = w
        end if
      end if
    end for
  end if
  return BestWord
```

Fig. 5. Best Suggestion Selection Algorithm

4) *Post Processing Module*

In this module, the input text will be scanned from the beginning and the words that are in the substitution list are replaced with the best suggestions. This methodology preserves the original formatting information of the text, including non-Sinhala words, numerals, punctuations and space among others.

Post Processing Algorithm

The function `PostProcess` accepts the original input text and the substitution list as the parameters. It will scan the input text for the words that are in the substitution list and replace them with corresponding best suggestions. Then, it will return the output text, which will be rendered by the display function.

```
PostProcess(InputText, SubstitutionList)
  OutputText = InputText
  for each word w in OutputText
   if w is in SubstitutionList then
      replace w in OutputText with
        SubstitutionList[w]
   end if
  end for
  return OutputText
```

Fig. 6. Post Processing Algorithm

IV. EVALUATION AND RESULTS

There is neither a standard lexicon for Sinhala spell checker evaluation, nor previous work reported for comparing with. Therefore, in order to evaluate our system, we used 5505 words obtained from a well known printed dictionary of inherently difficult and commonly misspelled words [15] as the baseline.

TABLE I
EVALUATION RESULTS OF SUBASA SPELL CHECKER

Test No.	Total Number of Words	# of correct suggestions	# of incorrect suggestions	Accuracy (%)
1	5505	4728	777	85.89
2	5505	4616	889	83.85
3	5505	4588	917	83.34
4	3304	2501	803	75.70

The first test was straight forward. Each entry was passed to the system and the output of the system was compared with the original entry. The second and third tests were much more stringent. The second test involved programmatically altering the original entries of the test data set so that all the dental letters were replaced by their corresponding retroflex letters. Furthermore, the unaspirated letters were replaced by their aspirated counterparts. These words were then used as the input to our spell checker, and the output was compared with the original unaltered entries. Similarly, in the third test, the aspirated letters in the original entries were altered to the corresponding unaspirated counterparts and dental letters were replaced by the corresponding retroflex letters. These words were then analyzed by our speller. The output was compared with the original unaltered entries. A fourth test was carried out by obtaining 20 randomly chosen blog articles published online [24]. The articles were analyzed for spelling errors by our system. For each of the above tests, the errors detected by our system were manually analyzed by an expert. The analysis revealed that our system has wrongly identified a small number of words as invalid (false negatives). Moreover, a few words which were identified as valid by our system were actually invalid (false positives). The test results are summarized in Table I.

The results show an overall accuracy of over 82% for the proposed algorithm. Manual analysis of the words that were wrongly suggested by our system as correct revealed that these words are not found in the corpus. Therefore, such words were suggested by the trigram or bigram calculation methods described in Section III-B-3. Prominent observations made by further analyzing such words are given below:

1. අනු- is a commonly used prefix in Sinhala. It has a higher syllable bigram and syllable trigram frequency. Therefore it can be erroneously substituted for අණු- to suggest a word with incorrect spelling.

   e.g. අණුජීවියා ➝ අනුජීවියා

2. The bigram frequency of the letter sequence -රන- shows that it is one of the most frequently used phonemic combination. Therefore, it can be erroneously substituted for -රණ- producing a word with incorrect spelling.

   e.g. කාර්මිකරණය ➝ කාර්මිකරනය

3. The trigram frequency of the suffix -ර්ශන shows that it is one of the most frequently used phonemic combination in the language. Therefore, it can be erroneously substituted for -ර්ෂණ.

e.g. ප්‍රකර්ෂණය ➔ ප්‍රකර්ෂනය

TABLE II
EVALUATION RESULTS OF MICROSOFT OFFICE SPELL CHECKER

Test No.	Total Number of Words	# of correct suggestions	# of incorrect suggestions	Accuracy (%)
1	5505	3453	2052	62.72
2	5505	1668	3837	30.29
3	5505	2416	3089	43.88
4	3304	2131	1171	64.49

TABLE III
EVALUATION RESULTS OF OPEN OFFICE SPELL CHECKER

Test No.	Total Number of Words	# of correct suggestions	# of incorrect suggestions	Accuracy (%)
1	5505	1078	4427	19.58
2	5505	720	4785	13.08
3	5505	1047	4456	19.02
4	3304	2443	861	73.94

4. Similarly, the trigram frequency of the suffix -ංශය is one of the most frequently used phonemic combinations in the language. Therefore, it can be erroneously substituted for -ංසය.

e.g. කඨිනානිසංසය ➔කටිනානිසංශය

To compare the effectiveness of our algorithm against currently available algorithms used by popular applications, we then performed the same tests on Microsoft Office Sinhala Spell Checker and Sinhala Ubiquitous Spell Checker version 3.0.1-beta-1 for OpenOffice.org. In all of these tests, we used the first correction the particular spell checker suggested if it identified an incorrect word. We made the assumption that these spellcheckers suggested corrections based on priorities and relevance and the first suggestion was the most appropriate correction for the relevant error.

Table II summarizes the test results for the Microsoft Spell Checker. It shows an overall accuracy of 50.35%. For naturally occurring text (blog corpus) it reports an accuracy just above 64%, which is not very useful in practice. Table 3 summarizes the test results for the Sinhala Ubiquitous Spell Checker version 3.0.1-beta-1 for OpenOffice.org. These results show an average accuracy of 31.41%, a lower value compared to the Microsoft Office Spell checker. However, it shows a better performance of over 73% for the blog text where spelling mistakes are not deliberate.

From these tests it is evident that our algorithm performs better in naturally occurring text, and much better in extremely bad cases of misspelling such as those simulated for tests 1, 2 and 3.

V. DISCUSSION

The n-gram statistics used in this spell checker were pre-compiled and stored in a database. The current database contains 440022 unique words with their frequency of occurrence in the corpus (word unigrams), 166460 distinct three syllable sequences (syllable trigrams) with their frequency of occurrence and 46878 two syllable sequences (syllable bigrams) with their frequency of occurrence. Our algorithm, combined with these statistics, is capable of processing virtually any given word. The algorithm used to calculate the syllable trigram frequencies is listed below:

```
for each TextFile in the text corpus
  Tokens=Tokenize(TextFile)
for each Token in Tokens
Chunuks
=DivideTokenIntoThreeSyllableChunks()
 for each ThreeSyllableChunk in Chunks
  if ThreeSyllableChunk is in Database
  then
    Occurrence=Occurrence+1
  else if
   InsertIntoDatabase(ThreeSyllableChunk)
    Occurrence=1
  end if
 end for
end for
end for
```

Fig. 7. Syllable Trigram Frequency Algorithm

The syllable bigrams were calculated in a similar manner.

In our algorithm, the complexity and efficiency lie in the permutation generation module. In this paper, we define the term *'complexity'* as the maximum number of words that can be generated for a given word. Using the same distinct word list obtained from the corpus, a few experiments were carried out to find the most complex Sinhala word (Additional details of these experiments are given in Appendix C). The study revealed that a word of local origin, 'පුස්තකාලාධිපතිතුමන්ලා' – librarians, can generate up to 3072 permutations. This word can be further inflected as 'පුස්තකාලාධිපතිතුමන්ලාත්', increasing the number of generated words up to 6144. Moreover, some lengthy borrowed words from *Pali* such as 'පංචුපාදානස්කන්ධයන්ගෙන්' (6144) and 'නෙවසඤ්ඤාණාසඤ්ඤායතනයාගේද' (9213) can generate up to 12288 permutations due to further inflections (e.g. 'පංචුපාදානස්කන්ධයන්ගෙනුත්'). However, such words are not used in everyday writing. Analysis of words with complexity higher than 6144 revealed that most such words are borrowed words that are no longer used in ordinary Sinhala writing. Some other words in the test set were found to be erroneous words (e.g. words with Unicode conversion errors, non-delimited words etc). Though it is safe to declare 6145 as the threshold for the complexity, allowing room for inflections of borrowed words and in order to shield the system from intentional *attacks*, we have set a threshold of 20000 as the maximum complexity that can be handled by the current implementation of our algorithm. The reason for this limitation is to avoid deliberate attempts to break the system by inputting a letter sequence with extremely high complexity. Any input that will result in more than 20000 permutations will be left unprocessed. Such words will be specially marked as '*unchecked'* in the output. All modern computers are capable of handing large amounts of data in a fast and reliable manner due to increased memory capacity and high-speed parallel processing capabilities. Therefore, the generation of 20000 permutations can be completed within a negligible amount of time. Furthermore, the study revealed that the word length does not significantly affect the complexity of a word (See Appendix C for details). The average word length (i.e. the number of Unicode code points) of Sinhala words was found to be 4. The locally originating maximum length word was found to be 'ජ්‍යෝතිශ්ශාස්ත්‍රයින්ගේ' – *astrologists'* in this study. It is interesting to note that there can be extremely lengthy borrowed words (from *Pali* or *Sanskrit*) such as 'නෙත්‍රාකෘෂ්ටටෛරිහාධිව්‍රජමලසලසද්බල්ලවීදොර්ලතාභි' though such words are very rare in modern texts.

VI. Conclusion and Future Work

The implementation of an n-gram based spell checker for Sinhala has been discussed in this paper. By substituting phonetically similar characters in a given word, permutations are generated and sent to the best suggestion selection module. The best suggestion selection module uses three techniques for ranking the generated permutations. The three techniques are based on word unigram frequencies, syllable trigram frequencies and syllable bigram frequencies, which are pre-computed from a raw text corpus. Empirical evaluation of our algorithm using four different test cases has revealed extremely promising results.

A platform independent desktop application with a graphical user interface (GUI) and a web-based version of the same have been developed using the *Python* programming language to demonstrate the functionality of the proposed algorithm. The usage of the applications are described in Appendix B. The accuracy of corrections suggested by the algorithm can be increased by simply adding non-existing words to the distinct word list and by increasing the unigram frequencies of words with correct spellings. It is expected to incorporate a crowd source based automated mechanism for improving the accuracy of the current spell checker.

Further enhancements planned include the optimization of the permutation generation module by storing and processing data using a *Trie* [3] data structure. This will help to effectively prune a large number of generated words to only those that appear in the distinct word list. The current algorithm is only capable of addressing substitution errors. The success of the application of the *Reverse Dictionary Lookup* methodology for other Indic languages [1] [4] [5] has motivated us to attempt the same approach for Sinhala. This will enable the algorithm to capture other types of spelling errors such as insertion, deletion and transposition [3] [4]. Research is underway to investigate the incorporation of the n-gram score computation methodology proposed in [9] for this purpose.

The algorithm applied for Sinhala, can also be used to construct spell checkers for other languages in which linguistic resources are scarce or non-existent. It is of particular relevance to languages which have rich morphology and thus are difficult to completely enumerate in a lexicon. Furthermore, the same algorithm can be utilized for the identification of homographs and common spelling mistakes found in Sinhala. To the best of our knowledge this is the first study and evaluation of a Sinhala spell checker algorithm. This study has opened up new opportunities for further research and will provide a baseline for comparison and evaluation of Sinhala spell checking algorithms in future.

Appendix A

Sinhala Spelling Rules

1. Use of Retroflex ණ /ɳ/ in Sinhala

1. Intervocalic Sanskrit and Pali ණ /ɳ/ does not get evolved [14].

   Example: රෝහණ /ro:həɳə/ > රෝහණ /ro:həɳə/ > රුහුණු /ruhuɳu/

2. Sanskrit 'ෂ්ණ' /ʂɳ/ > Pali 'ණ්හ' /ɳh/ > Sinhala ණ /ɳ/ [14]

   Example: උෂ්ණ /uʂɳə/ > උණ්හ /uɳhə/ > උණු /uɳu/, උණ /uɳə/

3. Retroflex ණ /ɳ/ is used in front of a retroflex consonant. Retroflex consonants are ට /ʈ/, ඨ /ʈ^h/, ඩ /ɖ/, ඪ /ɖ^h/.

   Examples: ඝණ්ටාර /g^{h}aɳʈa:rə/, කාණ්ඩ /ka:ɳɖə/, චණ්ඩාල /ʧaɳɖa:lə/

   Exceptions:

   i. Dental න /n̪/ is used before a retroflex consonant in words borrowed from western languages [16].

      Examples: කවුන්ටරය /kaun̪ʈərəjə/, කැන්ටිම /kæn̪ʈimə/

   ii. Dental න /n̪/ is used without a vowel before the letter /ʈ/ in dative case nouns [16].

      Examples: දරුවන්ට /d̪aruvan̪ʈə/, මිනිසුන්ට /min̪isun̪ʈə/

iii. In Sinhala, the suffix ට /-ʈə/ occurs in infinitives. In that case the /n̪/ in front of it doesn't appear as a retroflex /ɳ/.

Example: ලබන්ට /laban̪ʈə/, කියන්ට /kijan̪ʈə/

4. In certain constructions, germinated ණ /ɳ/ occurs. But those constructions belong to Indian languages.

Examples: උප්පලවණ්ණා /uppələvaɳɳə/, කණ්ණාඩි /kaɳɳa:ɖi/

5. Retroflex ණ /ɳ/ is used after ර /r/ in nouns and adjectives [15].

Examples: වර්ණ /varɳə/, පරිණත /pariɳət̪ə/, තරුණ /t̪aruɳə/

Exceptions :

i. Dental න /n̪/ is used before the letter ර /r/ in nouns.

Examples: තොරන් /t̪oran̪/, රෑන් /ræ:n̪/, රොන් /ron̪/ [16]

ii. Dental න /n̪/ is used after the letter ර /r/ in verbs [15].

Examples: කරනවා /kərən̪əva/, මරනවා /marən̪əva/, තෝරනවා /t̪o:rən̪əva/

iii. In present participles, the suffix න /-n̪ə/ occurs. It is not written as a retroflex ණ /ɳ/ in the vicinity of ර /r/.

Present participle Noun

Examples: මරන /marən̪ə/ මරණ /marəɳə/

iv. Dental න /n̪/ is used in the imperative verb suffix නු /-n̪u/.

Examples: කරනු /kərən̪u/, දරනු /d̪arən̪u/, හරිනු /harin̪u/

6. In certain evolved words, ණ /ɳ/ appear in original words. In certain borrowed words ර /r/ is not fully recorded. But half of it is called *rakaransa*. In the vicinity of Rakaransa retroflex ණ /ɳ/ is written.

Examples: ආමන්ත්‍රණ /a:man̪t̪rəɳə/, පරිත්‍රාණ /parit̪ra:ɳə/, ශ්‍රේණි /ʃre:ɳi/

7. Retroflex ණ /ɳ/ is used after the retroflex ෂ /ʂ/.

Examples: තෘෂ්ණා /t̪ruʂɳa:/, ගවේෂණ /gave:ʂəɳə/, දක්ෂිණ /d̪akʂiɳə/

8. In the honorific suffix ආණ /-a:ɳə/ and its variations always occurs ණ /ɳ/ [14].

Example: ආණ /-a:ɳə/ අණු /−aɳu/ අණි /−aɳi/

පියාණන් /pija:ɳan̪/ තෙරණුවන් /t̪erəɳuvan̪/ දියණි /d̪ijəɳi/

9. Retroflex ණ /ɳ/ is used in suffixes that ණ /−ɳə/ and ණි /−ɳi/ in intransitive past tense verbs.

Examples: ණ /-ɳə/ ණි /−ɳi/
æsiɳə væʈəhiɳi

10. Retroflex ණ /ɳ/ is used in suffixes that ණු /−ɳu/ and ණ /−ɳə/ in ancient intransitive verb particles.

Examples: ණු /-ɳu/ ණ /-ɳə/
ඉදුණු /id̪uɳu/ වැටුණ /væʈuɳə/

2. *Use of Dental න /n̪/ In Sinhala*

1. Sanskrit ර්ණ /rɳ/ > Pali ණ්ණ /ɳɳ/ > Sinhala න /n̪/ [14]

Example: කර්ණ /karɳə/ > කණ්ණ /kaɳɳə/ > කන් /kan̪/

2. Sanskrit ඍණ්හ /rhɳ/ > Pali ණ්හ /ɳh/ > Sinhala න /n̪/ [14]

Example: ගෘහ්ණාති /grhɳa:t̪i/ > ගණ්හාති /gaɳha:t̪i/ > ගනු /gan̪u/

3. Sanskrit ඥ /dʒɲ/ > Pali ඤ /ɲ/, ඤ්ඤ /ɲɲ/ > Sinhala න /n̪/ [14]

Example: ඥාති /dʒɲa:t̪i/ > ඤාති /ɲa:t̪i/ > නෑ /n̪æ:/

4. Sanskrit න්‍ය /n̪j/, ණ්‍ය /ɳj/ > Pali ඤ්ඤ /ɲɲ/ > Sinhala න /n̪/ [14]

Example: පුණ්‍ය /puɳjə/ > පුඤ්ඤ /puɲɲə/ > පින් /pin̪/

5. Sanskrit ශ්න /ʃn̪/, ෂ්ණ /ʂɳ/ > Pali ඤ්හ /ɲh/ > Sinhala න /n̪/ [14]

Example: ප්‍රශ්න /praʃn̪ə/ > පඤ්හ /paɲhə/ > පැන /pæn̪ə/

6. Sanskrit ණ /ɳ/ > Pali න /n̪/ > Sinhala න /n̪/ [14]

Example: නිර්වාණ /n̪irva:ɳə/ > නිබ්බාන /n̪ibba:n̪ə/ > නිවන් /n̪ivan̪/

7. Dental න /n̪/ is used without a vowel in front of a dental consonant. Dental consonant are ත /t̪/, ථ /t̪^{h}/, ද /d̪/, ධ /d̪^{h}/ [16]

   Examples: ත /t̪/ – චින්තන /ʧin̪t̪ən̪ə/, ශාන්ත /ʃa:n̪t̪ə/, කාන්තා /ka:n̪t̪a:/

   ථ/t̪^{h}/ – ග්‍රන්ථ /gran̪t̪^{h}ə/, මන්ථ /man̪t̪^{h}ə/

   ද /d̪/ – කොන්ද /kon̪d̪ə/, මන්දිර /man̪d̪ira/, සින්දු /sin̪d̪u/

   ධ /d̪^{h}/ – අන්ධ /an̪d̪^{h}ə/, සම්බන්ධ /samban̪d̪^{h}ə/, සන්ධි /san̪d̪^{h}i/

8. If a Sinhala word contains a geminated nasal consonant it should be dental න /n̪/.

   Examples: ආසන්න /a:san̪n̪ə/, වන්නම් /van̪n̪am/, වන්නි /van̪n̪i/

9. Dental න /n̪/ is used without a vowel before ස /s/ [16].

   Examples: පන්සල් /pan̪sal/, කාන්සි /ka:n̪si/, වහන්සේ /vahan̪se:/

10. The nasal that occurs at inanimate noun roots which ends in ඉ /i/ or උ /u/ is dental න /n̪/. When singular suffix අ /-ə/ occurs at end position of a word the last consonant doubles.

    Examples: ගිනි /gin̪i/, පිනි /pin̪i/, ඔටුනු /oʈun̪u/, දුනු /d̪un̪u/

11. Dental න /n̪/ is used after ස /s/ or ශ /ʃ/[17].

    Example: ස /s/ – උදැසන /ud̪æ:sən̪ə/, වාසනා /va:sən̪a:/, සේනා /se:n̪a:/

    ශ /ʃ/ – දර්ශන /d̪arʃən̪ə/, ප්‍රකාශන /prəka:ʃən̪ə/, ශූන්‍ය /ʃün̪jə/

12. Dental න /n̪/ without a vowel is used in endings of noun roots.

    Examples: පාවහන් /pa:vahan̪/, වදන් /vad̪an̪/

13. Dental න /n̪/ is used after ර /r/ in compound nouns.

    Examples: වහරනුසරෙන් /vaharan̪usaren̪/, පිරිනිවන් /pirin̪ivan̪/, බණ්ඩාරනායක /baɳɖa:rən̪a:jəkə/

3. *Use of Retroflex ළ /ɭ/ in Sinhala*

   1. Sanskrit and Pali ට /ʈ/, ඨ /ʈ^{h}/, ඩ /ɖ/, ඪ /ɖ^{h}/ > Sinhala ළ /ɭ/ (Jayathilake, 1937)

      Example: කූඨ /ku:ʈ^{h}ə/ > කළ /kaɭə/

   2. Pali ළ /ɭ/ > Sinhala ළ /ɭ/ [17]

      Example: දෘඪ /d̪rɖ^{h}ə/ > දළ්හ /d̪aɭhə/ > දළ /d̪aɭə/

   3. Sanskrit and Pali ණ /ɳ/ > Sinhala ළ /ɭ/ [17]

      Example: වාණිජ්‍යා /va:ɳiʤʤa:/ > වණිජ්ජා /vaɳiʤʤa:/ > වෙළෙඳ /veɭe^{n}d̪ə/

   4. Where both ල /l/ and ළ /ɭ/ obtain as alternatives in Pali Sinhala generally adopts the latter ළ /ɭ/ [14].

      Example: දලිද්ද /d̪aliddə/, දළිද්ද /d̪aɭid̪d̪ə/ > දිළිඳු /d̪iɭi^{n}d̪u/

   5. Retroflex ළ /ɭ/ is used on behalf of ර /r/ in past participles which are composed from verb roots ending in ර /r/.

      Examples: කර /karə/ – කළ /kaɭə/

      මර /marə/ – මළ /maɭə/

   6. Prefix පිළි /piɭi-/, which is derived from a Sanskrit prefix ප්‍රති /prət̪i-/, is used with retroflex ළ /ɭ/.

      Examples: පිළිගන්නවා /piɭigan̪n̪əva/, පිළිතුරු /piɭit̪uru/, පිළිබඳ /piɭiba^{n}d̪ə/

   7. Retroflex ළ /ɭ/ is used excessively before the nasalized consonants ඟ /^{ŋ}g/, ඳ /^{n}d̪/, ඹ /^{m}b/.

      Examples: ළඟ /ɭa^{ŋ}gə/, ළඳ /ɭa^{n}d̪ə/, කොළඹ /koɭə^{m}bə/ [15]

      Exceptions:

      1. Following words use the dental ල /l/ [15].

         Examples: පොලඹ /pola^{m}bə/, සලඹ /salə^{m}bə/

4. *Use of Dental ල /l/ in Sinhala*

   1. Sanskrit and Pali ල /l/, ල්ල /ll/ > Sinhala ල /l/

      Example: මහල්ලක /mahalləkə/ > මහලු /mahalu/ [17]

   2. Sanskrit and Pali ර /r/, න /n̪/ > ල /l/ [14].

      Example: කරුණා /karuɳa:/ > කුලුණු /kaluɳu/

      වන /van̪ə/ > වල් /val/

   3. The *Halant* form ල /l/ occurring at the end position of noun roots is always the dental ල /l/. When such words combine with the vowels, they retain the dental ල /l/.

      Examples: කකුල /kakulə/ කකුල් /kakul/, ගඩොල /gaɖolə/ ගඩොල් /gaɖol/, කරල /karələ/ කරල් /karal/

   4. When doubling the word-end consonant in the inflection of noun roots ending in ඉ /i/ or උ /u/, the dental ල /l/ is retained.

      Examples: ඇඟිලි /æ^{ŋ}gili/ – ඇඟිල්ල /æ^{ŋ}gillə/

      මහලු /mahalu/ – මහල්ලා /mahalla: /

5. *Use of Retroflex ෂ /ʂ/ in Sinhala*
    1. Retroflex ෂ /ʂ/ is used after ක /k/ without a vowel.

       Examples: අක්ෂර /akʂərə/, දක්ෂ /d̪akʂə/, භික්ෂු /b^{h}ikʂu/
    2. Retroflex ෂ /ʂ/ is used without a vowel before the letters of ට /ʈ/ and ඨ /ʈ^{h}/.

       Examples: අධිෂ්ඨාන /ad̪^{h}iʂʈ^{h}a:n̪ə/, ජ්‍යෝෂ්ඨ /ʤje:ʂʈ^{h}ə/, ධර්මිෂ්ඨ /d̪^{h}armiʂʈ^{h}ə/ දිෂ්ටිය /d̪iʂʈijə/, දුෂ්ට /d̪uʂʈə/, ශිෂ්ට /ʃiʂʈə/

6. *Use of Palatal ශ /ʃ/ in Sinhala*
    1. Palatal ශ /ʃ/ is used before dental න /n̪/.

       Examples: දර්ශනය /d̪arʃən̪əjə/, දේශනය /d̪e:ʃən̪əjə/, ප්‍රශ්න /praʃn̪ə/

APPENDIX B

Implementation of the algorithm

A platform-independent desktop application with a graphical user interface (GUI) and a web-based version of the same has been developed using *Python* programming language to demonstrate the functionality of the algorithm proposed in this paper.

The desktop version of the implementation (Figure B.1) automatically corrects the spellings for the input text. The left pane contains the input text and the right pane contains the corrected text.

The web-based version (Figure B.2) highlights the incorrect words and provides suggest corrections as a right-click context menu. The user can replace the text there in. In addition, correct words are flagged differently to provide improved visual feedback to the user. The suggestions provide additional information as to from which n-gram statistic (word unigram, syllable trigram or syllable bigram) the suggestion was made.

The web-based version has been further developed to facilitate user submissions of corrections to the system for improving the quality of the spellchecking functionality. This is available at http://www.subasa.net/.

Example: පාඨසාලාචාරිනිය ‘ළදරු මරණ අනුපාතය ඉහල යැම’ පිළිබඳ දේශණයක් අලුත්ගම කණිෂ්ඨ විද්‍යාලයීය ශ්‍රවණාගාරයේදී පැවැත්වීය.

APPENDIX C

A few experiments were performed to investigate the relationship between complexity, word length and corpus word frequency. We computed complexity and length of all unique words (440,022) found in the 10 million (10,132,451) word *UCSC Sinhala Corpus*.

A. Complexity Vs. Length

The first experiment investigated the relationship between word length and the complexity. Having removed any duplicates for word length and complexity pairs, ln (complexity) vs. word length graph was plotted. (See Figure `C.1). This graph shows that there can be words with the same length but different complexity values. The graph was of sawtooth type and when the word length is above 30, complexities for a particular word length decreased; at word length 60, complexity was 32. This showed that there is no proportionality between complexity and word length.

B. Complexity Vs. Frequency

To observe the relationship between Frequency and Complexity, ln(frequency) vs. ln(complexity) graph was drawn (see Figure C.2), where frequency for a particular complexity meant the summation of all frequencies corresponding to the words having that same complexity. This also was a sawtooth type graph and generally when the complexity increased the frequency decreased. But there was no regular pattern of decrease and we can safely say that there is no strong relationship between the frequency and the complexity.

C. Frequency Vs. Length

The third graph analysed the relationship of frequency to the word length to get a general idea about the distribution of words in the corpus. This graph plotted ln(frequency) against the word length (see Figure C.3). Up to word length of 4, the graph shows a drastic increase. At word length 4, the graph reaches the maximum frequency of 1647009. Further increase of word length shows a downward trend of frequency. It was also observed that the average Sinhala word length is 4. For word lengths beyond 25 it showed a sawtooth type behaviour and as the word-length goes over 43, it showed only a very low frequency most of the times. These extremely lengthy words were found to be erroneous words (i.e. corpus noise: typos, cleaning errors, Unicode conversion errors) in the raw corpus.

From all the above analyses, we can clearly say there is no relationship between the word length and the complexity.

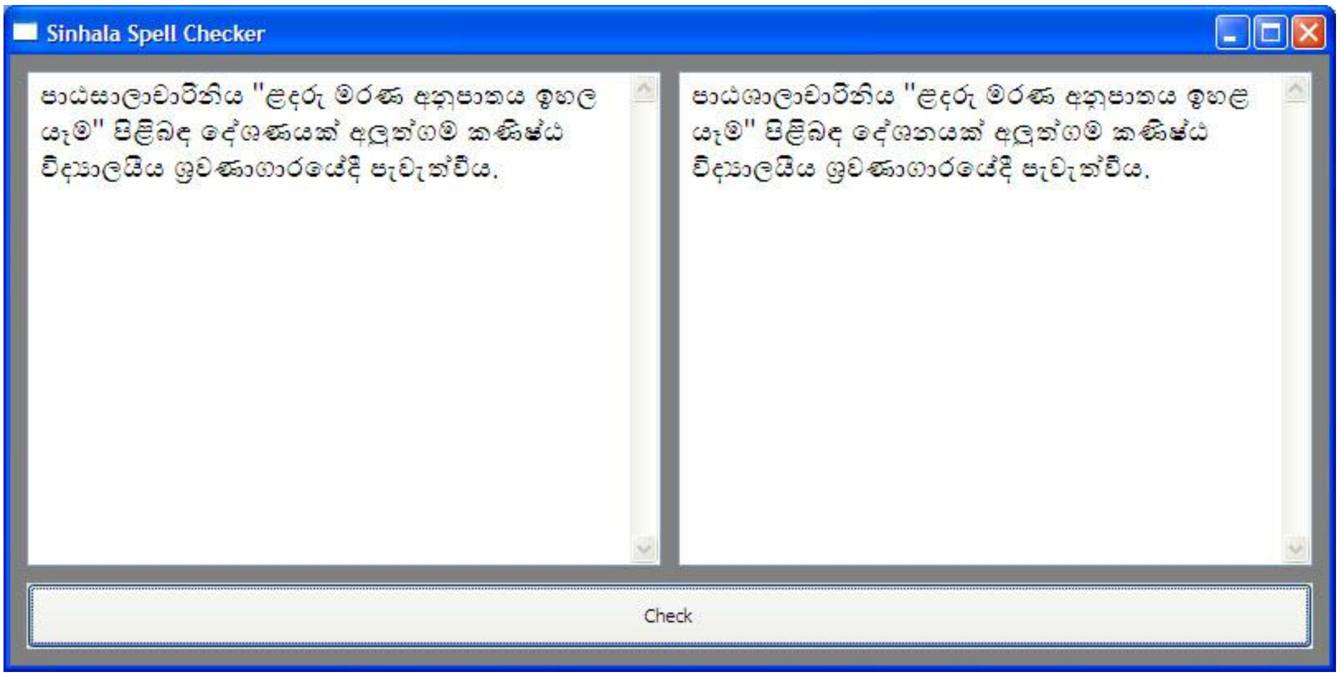
ACKNOWLEDGMENT

We are immensely grateful to Professor *Tissa Jayawardenan* for reviewing the linguistic rules. We also thank our colleagues *Vincent Halahakone*, *Namal Udalamatta* and *Jeewanthi Liyanapathirana* who provided insight and expertise that greatly assisted this research. We would also like to thank the Localisation Research Centre, University of Limerick, Ireland; especially Reinhard Schäler and Karl Kelly for their invaluable support.

REFERENCES

[1] B. B. Chaudhuri, "Towards Indian language spell-checker design," *Language Engineering Conference (LEC'02)*, 2002, p. 139.

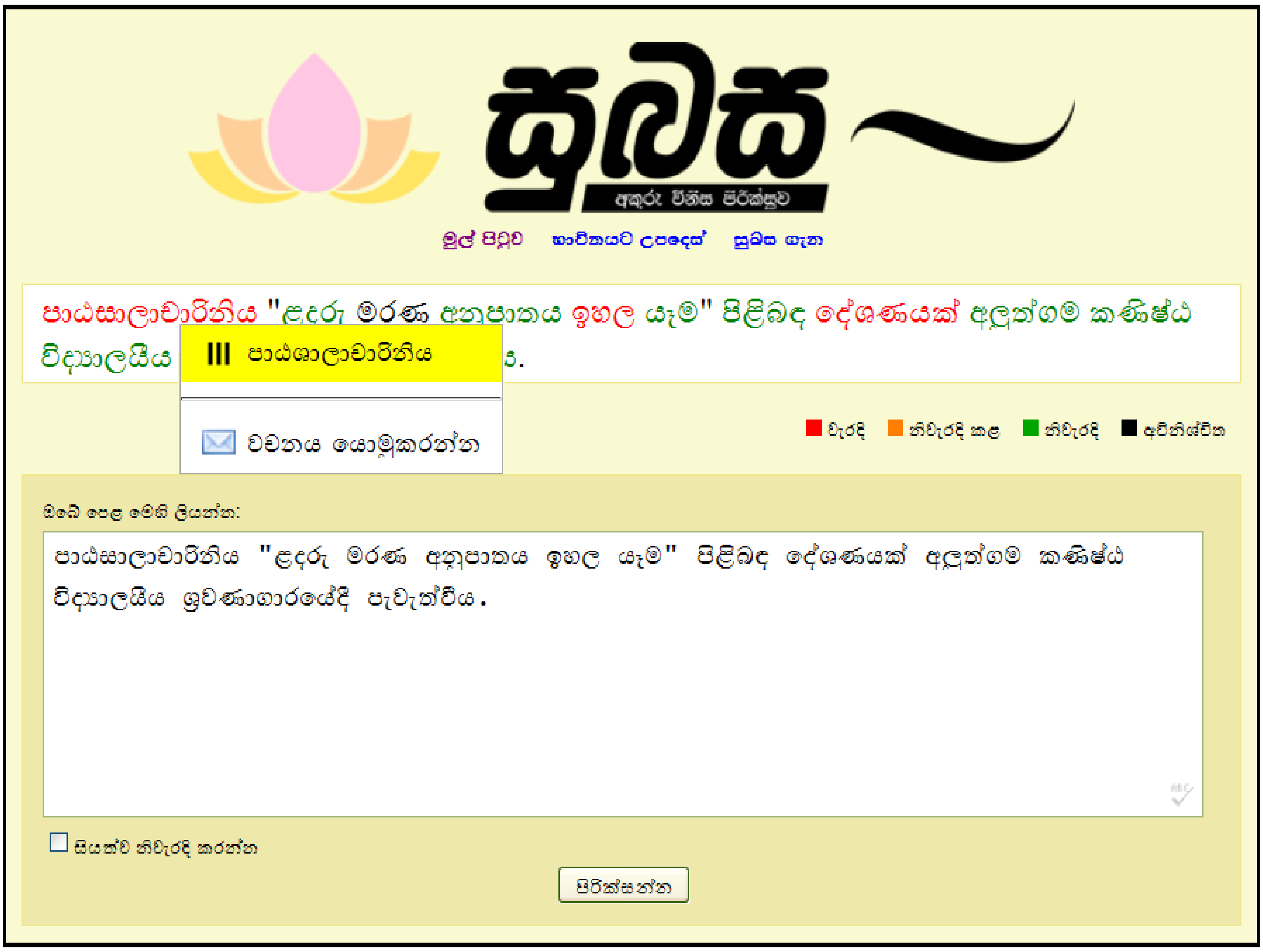
[2] M. Das, S. Borgohain, J. Gogoi, S. B. Nair, "Design and Implementation of a Spell Checker for Assamese," *Language Engineering Conference (LEC'02)*, 2002, p. 156.



පාඨසාලාචාරිනිය "ළදරු මරණ අනුපාතය ඉහල යෑම" පිළිබඳ දේශණයක් අලුත්ගම කණිෂ්ඨ විද්‍යාලයීය ශ්‍රවණාගාරයේදී පැවැත්වීය.

පාඨශාලාචාරිනිය "ළදරු මරණ අනුපාතය ඉහළ යෑම" පිළිබඳ දේශනයක් අලුත්ගම කණිෂ්ඨ විද්‍යාලයීය ශ්‍රවණාගාරයේදී පැවැත්වීය.

Fig B.1. The Desktop Application



පාඨසාලාචාරිනිය "ළදරු මරණ අනුපාතය ඉහල යෑම" පිළිබඳ දේශණයක් අලුත්ගම කණිෂ්ඨ විද්‍යාලයීය ය.

ඔබේ පෙළ මෙහි ලියන්න:

පාඨසාලාචාරිනිය "ළදරු මරණ අනුපාතය ඉහල යෑම" පිළිබඳ දේශණයක් අලුත්ගම කණිෂ්ඨ විද්‍යාලයීය ශ්‍රවණාගාරයේදී පැවැත්වීය.

Fig B.2. The Web-Based Application

[3] K. Kukich, Techniques for automatically correcting words in text. *in ACM Computing Surveys*, vol. 24, no. 4, 1992, pp. 377-439.

[4] B. B. Chaudhuri, "Reversed Word Dictionary and Phonetically Similar Word Grouping Based Spell Checker to Bangla Text," *in Proceedings of the LESAL Workshop*, Mumbai, India, 2001.

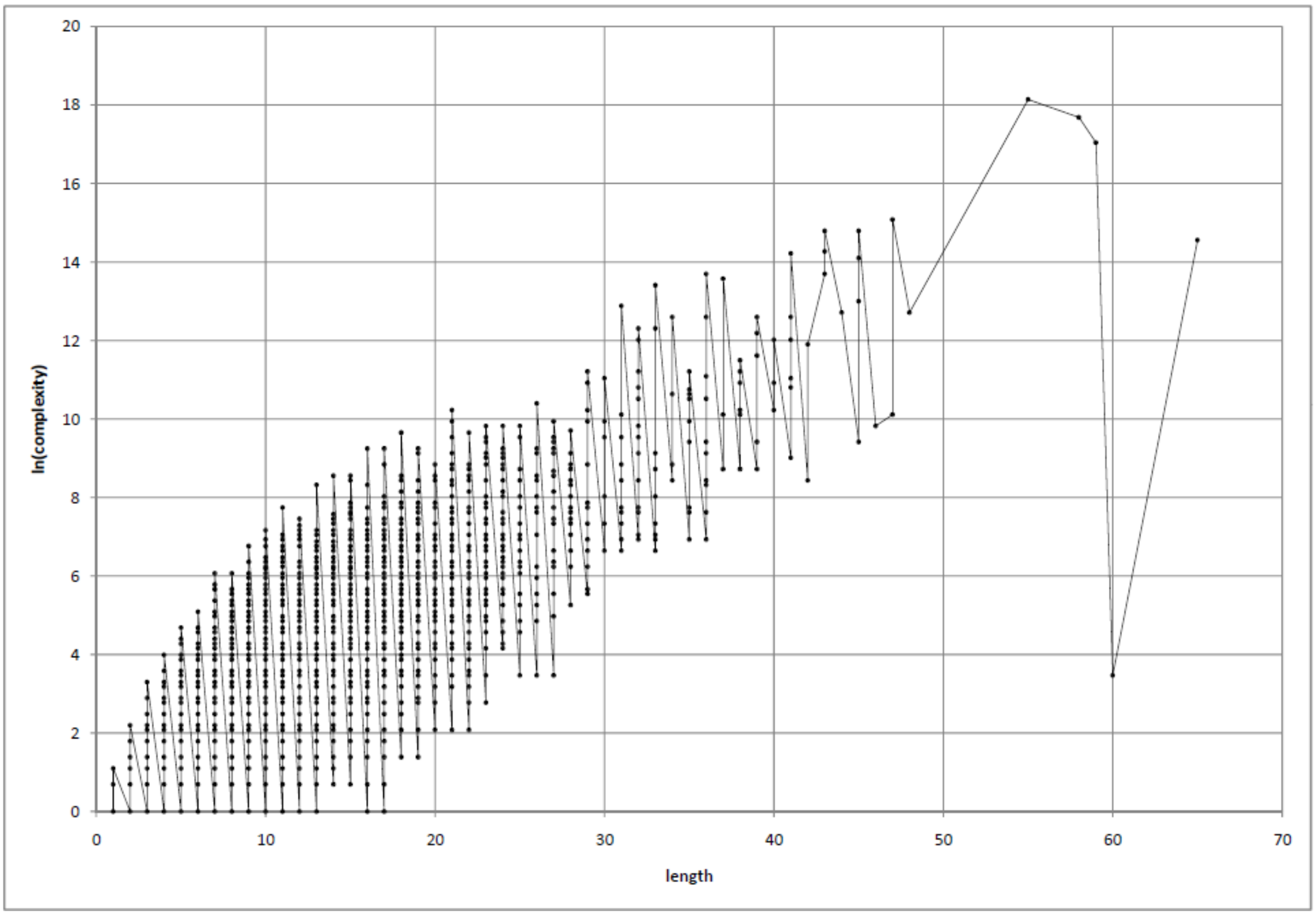


Fig C.1. The Graph of ln(Complexity) vs. Word Length

[5] T. Santhosh, K. G. Varghese, R. Sulochana, and R. Kumar, "Malayalam Spell Checker," *in Proceedings of the International Conference on Universal Knowledge and Language - 2002*, Goa, India, 2002.

[6] V. Dixit, S. Dethe, and R. K. Joshi, "Design and Implementation of a Morphology-based Spellchecker for Marathi, an Indian Language," *Archives of Control Sciences*, vol. 15, no. 3, (n.d.), pp. 301-308.

[7] T. Dhanabalan, R. Parthasarathi, and T. V. Geetha, "Tamil Spell Checker", *In Proceedings of 6th Tamil Internet 2003 Conference*, Chennai, Tamilnadu, India, 2003.

[8] S. Hussain, N. Durrani, and S. Gul. (2005). Survey of Language Computing in Asia, Center for Research in Urdu Language Processing, National University of Computer and Emerging Sciences. [Online]. Available: http://www.panl10n.net/english/outputs/Survey/Sinhala.pdf.

[9] F. Ahmed, E. W. D. Luca, and A. Nürnberger, "MultiSpell: an N-gram based language-independent spell checker," *In poster session of 8th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2007)*, Mexico City, Mexico, 1997.

[10] A. Wasala, R. Weerasinghe, and K. Gamage, "'Sinhala grapheme-to-phoneme conversion and rules for schwa epenthesis,". *In Proceedings of the COLING/ACL Main Conference Poster Sessions*, 2006, pp. 890-897.

[11] J. B. Disanayaka, *Sinhala Akshara Vicharaya (Sinhala Graphology)*, Sumitha Publishers, 2006, ISBN: 955-1146-44-1.

[12] W. S. Karunatillake, *An Introduction to Spoken Sinhala (3rd Edn.)*, M. D. Gunasena & Co. Ltd., 217, Olcott Mawatha, Colombo 11, Sri Lanka, 2004, ISBN 955- 21-0878-0.

[13] J. W. Gair, and W. S. Karunatillake, *The Sinhala Writing System, A Guide to Transliteration*, Sinhamedia, P.O. Box 1027, Trumansburg, NY 14886, 2006.

[14] J. Lanerolle, *The Uses of න-/n/,ණ-/ɳ/ and ල-/l/, ළ -/ɭ/ in Sinhalese Orthography*, The Times of Ceylon Company Limited, Colombo, 1934.

[15] S. Koparahewa. *Dictionary of Sinhala Spelling*, S. Godage and Brothers, Colombo 10, Sri Lanka, 2006, ISBN 955-20-8266-8.

[16] J. B. Disanayaka, *The Usage of Dental and Cerebral Nasals*, Sumitha Publishers, 2007, ISBN: 978-955-1146-66-5.

[17] D. B. Jayathilake, *Sinhala Shabdakoshaya (Sinhala Dictionary), Prathama Bhagaya (Vol 1)*, Sri Lankan branch of Royal Asian Society, 1937.

[18] http://www.mysinhala.com/features.htm, http://www.microimage.com/press/MicroimageDirectOctober2005.htm and http://www.scienceland.lk/spell-checker.html

[19] http://hunspell.sourceforge.net/

[20] http://wiki.services.openoffice.org/wiki/Dictionaries#Sinhala_.28Sri_Lanka.29

[21] https://addons.mozilla.org/en-US/firefox/addon/13981/

[22] http://www.sasrutha.com/ and http://www.facebook.com/note.php?note_id=176602427414

[23] The *UCSC Sinhala Corpus* is a ten million word raw corpus compiled from various sources. http://www.ucsc.cmb.ac.lk/ltrl/?page=panl10n_p1&lang=en&style=default#corpus

[24] The blog posts were collected from http://blogs.sinhalabloggers.com/ (a popular Sinhala Unicode blog syndicator) on 1st of September, 2009.

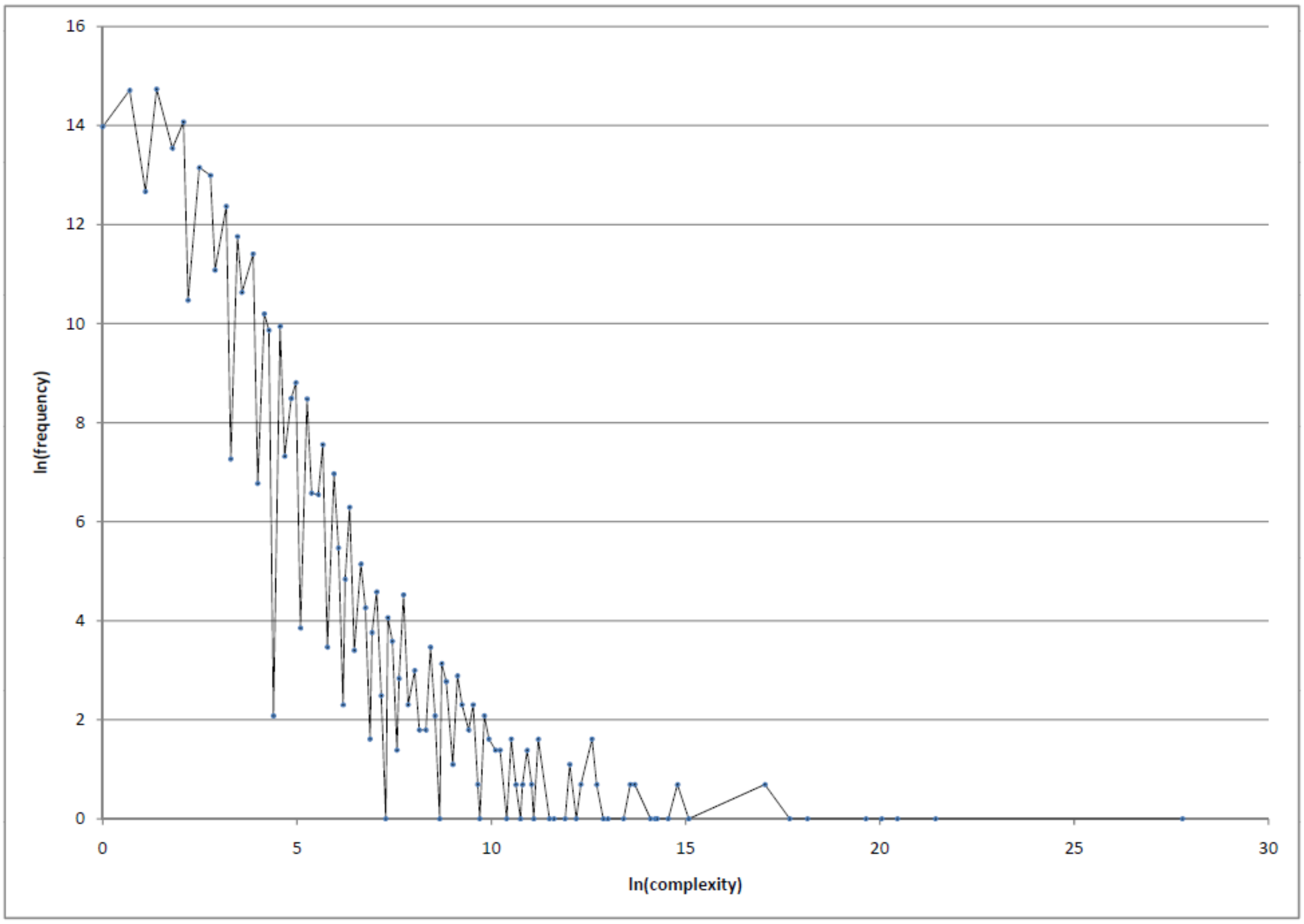


Fig C.2. The Graph of ln(Frequency) vs. ln(Complexity)

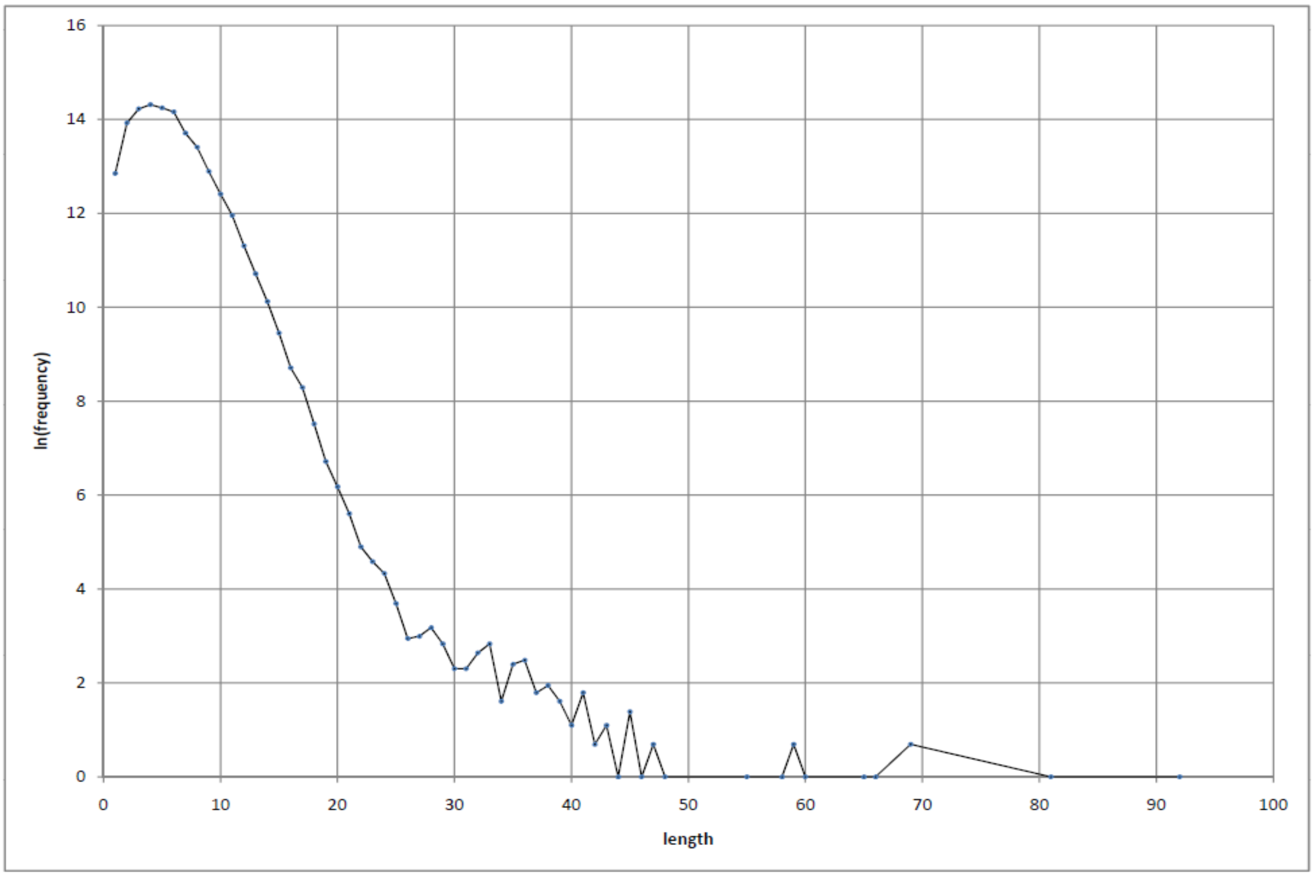


Fig C.3. The Graph of ln(Frequency) vs. Word Length