

Contextual Suggestion Engine for UCSC Singlish Unicode Converter

W.M. Janaka Bandara

2018



Contextual Suggestion Engine for UCSC Singlish Unicode Converter

**A dissertation submitted for the Degree of Master of
Computer Science**

W.M. Janaka Bandara

University of Colombo School of Computing

2018



Abstract

Contextual suggestion engine for the Sinhala language provides end users word suggestions as they type their sentences or phrases by analysing what domain they are writing their document on and predicting what word the user most likely to type next. Even though this is not a new idea and languages such as English already have these kinds of prediction engines implemented, it is not common to see one for the Sinhala language.

This thesis describes a methodology to analyse user input and predict the next word and check the effectiveness of this methodology. As the prediction model, this thesis presents a hybrid model of Ngram model and Markov model. Ngram model is used to predict the next possible match for the given phrase and Markov model to predict the probability of occurrences. Addition to these models, experiments on how term frequency and inverse document frequency can affect the suggestion probability are included in this thesis.

Analyzing the Sinhala language is different from analysing ASCII languages. In this thesis, it describes how to apply above methodologies for Sinhala language and how to overcome difficulties when analysing phrases or words of Sinhala language.

In the testing that had conducted in this thesis, by training about 170 thousand sentences it gives roughly 50-60 % accuracy on suggesting a relevant word when typing in general context. When the domain of the user's context changes, the accuracy dropped down to 40 – 50%. A reason for the dropped accuracy can be mainly due to the comprehensiveness of the domain-specific dataset. However, overall it is within the acceptable accuracy range.

In conclusion, we can use hybrid of Ngram and Markov models to build a suggestion engine but with tweaked features of generic Ngram and Markov models to analyse the Sinhala language properly.

Acknowledgement

I would first like to thank my thesis advisor Dr M.I.E Wickramasignhe of the Computer Science at the University of Colombo. Dr Wickramasignhe always there to help when I ran into a trouble spot or had a question about my research or writing and steered me in the right the direction whenever he thought I needed it.

I also like to thank all the lecturers at Colombo University who helped me to understand concepts and theories I have used in this thesis. And I would like to thank UCSC Language Technology and Research Learning division for providing me with Sinhala corpus, Surgery.lk for providing me with medicine related articles, and let me use it in this thesis.

Finally, I must express my gratitude to my parents and my friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Table of Content

| | |
|---|------|
| Abstract..... | iv |
| Acknowledgement..... | v |
| Table of Content..... | vi |
| List of figures | viii |
| List of Tables..... | ix |
| 1 Introduction | 1 |
| 2 Background..... | 3 |
| 3 Methodology..... | 9 |
| 3.1 Top level architecture | 9 |
| 3.2 Data preparation for analysis | 10 |
| 3.3 Application of Markov chain..... | 11 |
| 3.4 N-gram model..... | 12 |
| 3.5 N-gram with frequency..... | 14 |
| 3.6 Combined usage of unigram, bigram and trigram | 15 |
| 3.7 Probabilistic Smoothing..... | 15 |
| 3.8 Dataset and training | 16 |
| 3.9 Informal and mix language support | 16 |
| 3.10 Adding context bias..... | 17 |
| 4 Implementation..... | 19 |
| 4.1 Data extraction..... | 19 |
| 4.2 Structuring of data | 21 |
| 4.3 Training..... | 21 |
| 4.4 Analysis | 22 |
| 4.5 Matching Ngrams | 23 |
| 4.5.1 General query analysis..... | 23 |
| 4.5.2 Analyze probability of occurrence | 23 |
| 4.6 Suggestion engine frontend implementation | 24 |

| | | |
|-------|--|----|
| 4.6.1 | Add the suggested word to the end of the phrase..... | 24 |
| 4.6.2 | Complete the currently typing word..... | 25 |
| 5 | Evaluation..... | 26 |
| 5.1 | Quantitative evaluation..... | 26 |
| 5.1.1 | With news related sentences..... | 27 |
| 5.1.2 | With medicine related sentences. | 28 |
| 5.2 | Qualitative evaluation..... | 29 |
| 5.3 | Tools and technologies used in the evaluation. | 31 |
| 6 | Conclusion and future work | 32 |
| 7 | References | 33 |

List of figures

| | |
|---|----|
| Figure 1-1: Internet usage statistics in Sri Lanka [19]..... | 1 |
| Figure 2-1: How the word “ලංකා“ can be written in UCSC Singlish Unicode Converter | 5 |
| Figure 2-2: BBC news Sinhala website search result for the word “ලන්කා”..... | 6 |
| Figure 2-3: BBC news Sinhala website search result for the word “ලංකා”..... | 6 |
| Figure 2-4: Synset table - Sinhala Wordnet Database..... | 7 |
| Figure 3-1: Architecture diagram | 9 |
| Figure 3-2: Markov Chain analysed map | 11 |
| Figure 3-3: Markov Chain analysed map with frequency | 12 |
| Figure 3-4: Trigram map of sentences starting with I [18]..... | 13 |
| Figure 3-5: Common phrase extraction..... | 17 |
| Figure 4-1: Non-standard text file | 19 |
| Figure 4-2: Un-sanitized text..... | 20 |
| Figure 4-3: Analyzer workflow | 22 |
| Figure 4-4: Example for TF/IDF processing | 23 |
| Figure 4-5: Word suggestion UI..... | 24 |
| Figure 4-6: Autocomplete (before adding the new word) | 24 |
| Figure 4-7: Autocomplete (after adding a new word) | 25 |
| Figure 4-8: Autocomplete currently typing the word (before)..... | 25 |
| Figure 4-9: Autocomplete currently typing the word (after)..... | 25 |

List of Tables

| | |
|---|----|
| Table 2-1: Accuracy of the algorithm against the Restricted | 8 |
| Table 4-1: Unigram structure | 21 |
| Table 4-2: Bigram structure..... | 21 |
| Table 4-3: Trigram structure | 21 |
| Table 5-1: Confusion matrix structure | 26 |
| Table 5-2: Hybrid model confusion matrix | 27 |
| Table 5-3: Markov model confusion matrix..... | 27 |
| Table 5-4: Hybrid model confusion matrix (medicine related)..... | 28 |
| Table 5-5: Markov model confusion matrix (medicine related) | 28 |
| Table 5-6: Feedback participation | 29 |
| Table 5-7: Criteria 1 average feedback..... | 29 |
| Table 5-8: Criteria 2 average feedback..... | 30 |
| Table 5-9: Criteria 3 average feedback..... | 30 |
| Table 5-10: Criteria 4 average feedback..... | 30 |
| Table 5-11: Criteria 5 average feedback..... | 30 |

1 Introduction

Word suggestion engines are an essential part of search engines to provide better user experience to the users. Many popular search engines such as Google, Bing, and Yahoo provide this feature. This feature allows users to reduce typos and perform better search queries. Even though this is successfully implemented in main languages such as English, French, German, Tamil in the world, there is an inconsistency when it comes to the Sinhala language. Even in Google search, when user types in Sinhala Unicode, it does not provide search suggestions at the moment.

As seen on Figure 1-1: Internet usage statistics in Sri Lanka, internet usage in Sri Lanka had increased drastically over the last few years; According to digitalmarketer.lk, demand for internet usage in Sri Lanka is rising despite the increased tax rates. Most importantly, the growth rate in the usage of internet in areas such as education, recreation and occupational

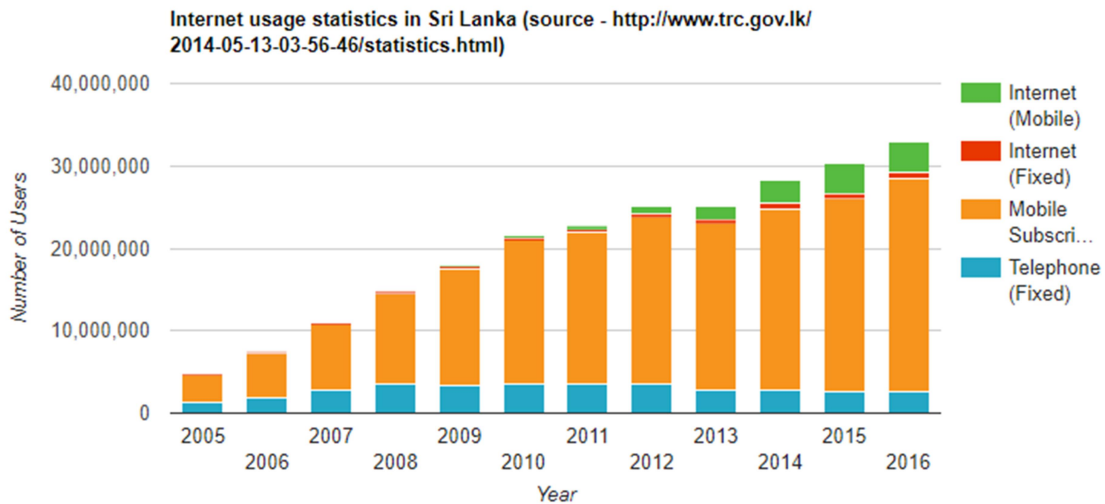


Figure 1-1: Internet usage statistics in Sri Lanka [19]

reasons remain sturdy over the years.

Therefore this subject matter becomes essential in local context due to the tendency in searching content on the internet in Sinhala. More and more search engines are evolving over the years to fill this demand, and lack of word suggestions in searching content in Sinhala has become very inconvenient for the users. Sometimes this leads to wrong search results or not-relevant topics. Since there are few ways to type a Sinhala word in English, the results may vary according to different ways that the user use.

The Sinhala language is a complex and morphologically rich language. Meaning of a word can be varied depending on the context and where it mentioned in a sentence. Further Sinhala is a continuously evolving language. When spoken Sinhala converted directly to another language, the method to get the meaning and provide the relevant suggestions might not yield correct meaning. Understanding informal written Sinhala is a critical part of this process. Some of the rules, practices and word selections in formal written Sinhala might not be applied practically in nonformal written Sinhala. Also in domains such as information technology, it most likely to use the English word as it is in a sentence rather than using Sinhala translation of that word.

There is no straightforward answer or a methodology to address this problem. Even though there are some reasonable solutions available for this problem in other languages such as English, research done on the Sinhala language is quite limited. However, recent successful attempts on building Sinhala Wordnet [1], research on machine learning approach to Sinhala morphological analysis [2], research on English to Sinhala machine translation [3] and research on data-driven approach for Sinhala spell checking [4] are few promising developments to solve this problem.

As the first step to finding a solution for this problem, the system should be trained to identify nouns, verbs, adjectives, compound nouns etc. in a sentence. It is vital to rank which words are more relevant to the sentence. Some stopwords might not add value to the sentence, in which case we can ignore that word at the analysing stage.

In the second step, it is necessary to map and build a word network to identify the correlation between words and possible word occurrences. In order to achieve this, it is required to analyse a large set of sentences to build a relationship between the words.

In the attempt of finding a better approach to the problem, it is more productive to follow a data-driven approach. The reason being the limited resources, time constraint and morphologically rich nature of the Sinhala language. It is highly unlikely that; this approach will provide better results for every word and phrase in the Sinhala language. The primary purpose of this research is to find whether it is possible to provide acceptable word suggestions for selected word set. Even though UCSC¹ Sinhala Corpus consists of 10M words [2] it is practical to select 5000 most frequently occurring words which have selected in Sinhala Wordnet [1].

¹ University of Colombo School of Computing

2 Background

Present day how we interact with computers is very different to how we used to interact with them. People are more connected with electronic devices than ever before. According to a recent survey done in the USA, it shows about 88.5% of the population use the internet [5]. When considering internet experience, “search” plays a massive role. There are over 6 billion search queries performed in a day [6]. Providing user-friendly search experiences for the users is important. Now a day people are expecting a search engine to be smart and efficient rather than searching for the specifics. People tend to use natural language to perform searches than specifying keywords.

Early days of search engines it used to match keywords and provide results, but this proved to be less productive and bit erroneous for a general user. Since users may not type exact keywords or may include typos, this can reduce the effectiveness of the search result. To solve this keyword dilemma, scientist and programmers looking for search engine algorithms which can handle natural languages. The theory is to get the meaning of the sentence and perform searches.

This makes NLP² is a must feature in modern day searching algorithms. The behaviour of the NLP profoundly contributed by the availability of grammatical rules and the complexity and the size of the lexical databases. Larger and complex grammatical rule set with an extensive database can provide more accurate results [7].

Even though NLP based search engines are possible; those are expensive and resource intensive. This lead to suggestion engines for search engines. The main idea is to minimise errors of the user input before performing the query. It is productive to ask the users whether they are entering the query correctly and recommends words for the user before performing guessing in the system after query submission. Google predictive search engine is an excellent example for this [8].

To perform better suggestions, it is recommended to have two subcomponents in the suggestion engine.

- Auto-completer module
- Phrase suggestion module

² Natural Language Processing

Auto-completer module is responsible for checking spelling mistakes of currently typing a word and recommend words. When recommending words, it only considers the word that is currently typing. Algorithms written for English language and other similar languages are generally used Levenshtein's distance [9] theory or phonetic algorithms such as Metaphone [10], Soundex [11] to implement fuzzy search algorithm in the auto-completer module. This also requires a significant enough word corpus to do the cross-checking.

Phrase suggestion module is responsible for suggesting possible upcoming words in the sentence. This module will analyse the whole sentence and cross-reference with the database in order to provide most suitable suggestions. In the phrase suggestion module, the sentence will be analysed in few steps, in a simple phrase suggestion algorithm we can find at least few tokenisers, analysers and stemmers to simplify the search query. As an example: Consider the sentence "The quick brown foxes". If we tokenise the words we get [The, quick, brown, foxes]. Then convert each word to the lower case, which gives [the, quick, brown, foxes]. If we consider the keywords we can identify that stop-word "the" which we can eliminate. It provides us with [quick, brown, foxes]. This we can normalise and reduce to their root forms. The keywords [quick, brown, fox] can be crossed referenced with the database and retrieve similar sentences. This is an example which is not perfect, but this can be modified and optimised to get better suggestions for the user.

How do we cross-reference documents in phrase suggesting algorithm? This is another problem we will be facing in the implementation phase. The current practice of main search implantations such as Elasticsearch [12] is to use the n-gram extract algorithm [13]. N-Gram can be used to predict the likelihood of an upcoming word thus increases the accuracy of the suggestion.

Improving the accuracy of search suggestion is quite important. Introducing context to the suggestion and drastically improve the user experience of the system. Context can be segregated into two main categories.

- User context
- Document context

User context is where search engine tracks past user data, search queries etc. This can provide some idea about what the user tries to search for. Using this for new users is not that effective due to the lack of information of the user [14]. Document context is meta-data about the document. What kind of information stored in the document, what is the source, any social

media relationship etc. [14]. In this research mainly focusing on improving document context thus improves the overall accuracy. This is mainly due to the document-centric behaviour of the system to be built.

Implementing a suggestion algorithm for the Sinhala language is the primary focus of this research. Applying above algorithms as it is for the Sinhala language will not provide accurate results. Sinhala is a morphologically rich language with more complex structure and loose standards. In the paper “Evaluating a Machine Learning Approach to Sinhala Morphological Analysis” [2] to apply Morfessor algorithm, they have defined a Gold Standard for Sinhala as the base case. This is mainly due to the language complexity and lack of standards. Adapting analytical algorithms to fit for the Sinhala language will be the a challenge of this project.

The paper “A Data-Driven Approach to Checking and Correcting Spelling Errors in Sinhala” [4] gives a better insight on how to approach spell checking for the Sinhala language. Spell checking implementations such as Subasa [15] will provide the spell checking feature. This can be utilised to generate a recommendation for the currently typing word by the user.

When considering the practical application of this project, most of the websites in Sinhala do not provide a search option, and the websites that provide search options do not show relevant word suggestions. If a word is not spelt correctly (in Sinhala); the search results might not show relevant results or lead to no content at all. For instance; there are two ways to type the word “ලංකා” in Singlish (lanka or la\nkaa). See Figure 2-1 below.

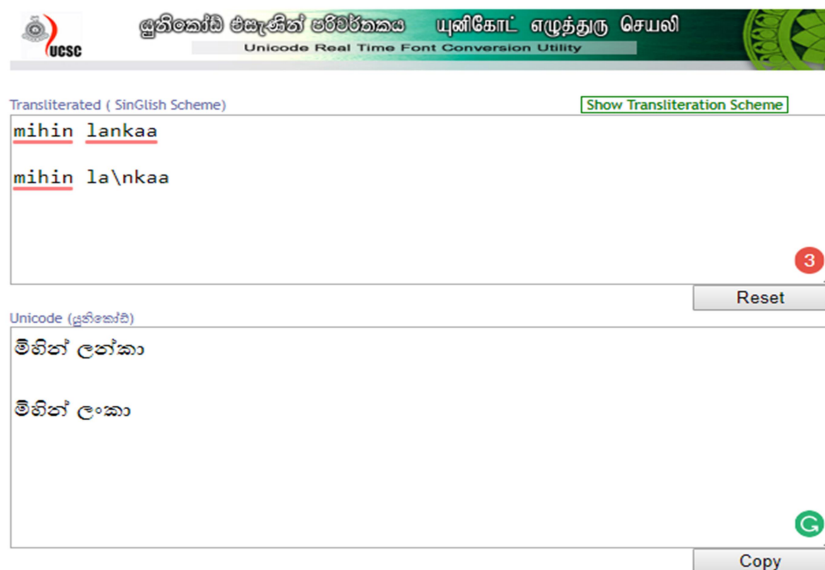


Figure 2-1: How the word “ලංකා” can be written in UCSC Singlish Unicode Converter

In this case, when a user types the most common way (Lanka); the search results will not appear and most users are not familiar with the specific ways (la\nkaa) of typing certain

words. Figure 2-2 and Figure 2-3 show two different ways of typing the same word could lead to an entirely different search results.

- “ලන්කා” spelt in English “Lankaa”

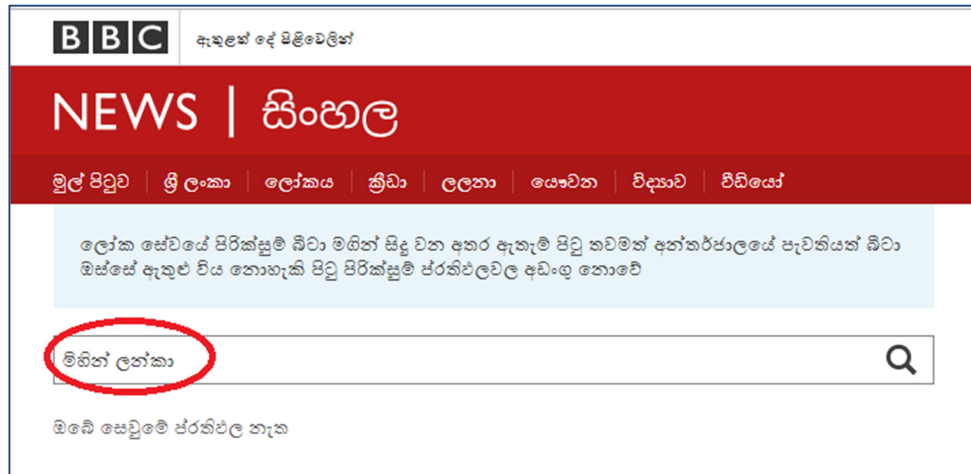


Figure 2-2: BBC news Sinhala website search result for the word “ලන්කා”

- “ලංකා” spelt in Singlish “la\`nkaa”



Figure 2-3: BBC news Sinhala website search result for the word “ලංකා”

Above two examples shows how two different ways of typing the same word give different results. Therefore relevant word suggestions play a crucial role in search engines in order to

maximise the user experience. In the above-mentioned situation, if the word suggestion “ලංකා” appeared after typing “මිහිනි” it would be much easier for the user to get the results that he/she been looking for.

Research and past work on Sinhala phrase suggestion algorithms are currently limited. However, there are related researchers such as Sinhala Wordnet [1], Computational Grammar of Sinhala [16] and Morphological Parser [17]. Main theories in these research papers can be used to support the implementation of the suggestion algorithm.

As for the Sinhala Wordnet research, “Compounding is a very productive morphological process in Sinhala. Both Sinhala nouns and verbs formed by compounding nouns (nouns) and nouns with verbs (e.g. verbs do and be) are extremely productive. As a result of this compounding, the original sense of the constituents of the compound noun is altered, resulting in the derivation of a new sense. The methodology we used to extract the most important senses (as explained in 2.1) does not detect compound words since we used the most frequent single words extracted from the corpus [1]. In order to continue using this methodology, we might have to alter the algorithm to support compound words even though we decide to limit our research for the same selected dataset (1000 words) [1] in Sinhala Wordnet.

And also adding contextual meta-data to the Synset table (refer Figure 2-4) developed in the paper [1] would be a great asset when evaluating the contextual value of a sentence.

| Synset | PWN ID | POS | Gloss |
|----------------------------|----------|-----|---|
| සදහන;සවහන;විස්තරය | 06418196 | n | කෙටි ලිවින සවහන |
| ඉන්නවා | 02703136 | v | තැනක හෝ තත්වයක නැවතී සිටිනවා |
| මූලික | 01922424 | s | මූලික දත්ත හෝ පුනිපත්තිවලට අදාළ වන්නා වූ |
| ජීවිතය | 13777175 | n | ජීවත්වීමේ නෛසර්ගික ආකාරය හෝ රටාව |
| වැඩ;රාජකාරිය;කාර්යය;කටයුත් | 00570312 | n | යමක් කිරීම සඳහා හෝ සෑදීම කෙරේ යොමු වූ ක්‍රිය |
| බෙදර | 08373013 | n | මිඬු ජීවත් වන රට, ප්‍රාන්තය හෝ නගරය |
| අදහස;සිතුවීම;චින්තනය | 05761049 | n | සංකල්පනාව, ප්‍රජානනයේ අන්තර්ගතය |
| පමණ;තරම;ගණන;රාශිය;විපස | 00032028 | n | ප්‍රමාණ කළහැකි යමකින් ඇති ප්‍රමාණය |
| සුරනවා | 00479055 | v | අවසාන කරනවා හෝ අවසානයට පැමිණෙනවා |
| විනාශය | 07233906 | n | යමක් සම්පූර්ණයෙන් සිඳි බිඳ දමන සිද්ධිය (හෝ සිඳි |
| මත;හැඟීම;අදහස | 05841869 | n | යම් විශ්වාසයක් තබනු ලැබූ අනුමාන අදහසක් |
| මූලික | 01051890 | s | ආරම්භයක් ලෙස හෝ පදනමක් ලෙස සේවය කරන |
| නිර්මාණය කරනවා | 01607166 | v | කෘතීම නිෂ්පාදනයක් ඇති කරනවා |
| සාමාන්‍යයෙන් | 00491749 | r | ස්වාභාවික හෝ සාමාන්‍ය අන්දමට |
| නැවත;ආපසු;ආයින්;ආයිමින්;ආ | 00041086 | r | ආයෙමින්, අප්තීන් |
| ජාතික;දෑ | 03070805 | a | ජාතියකට හෝ රටකට අයත් |
| දෙන;වැස්සි;එළදෙන | 01868513 | n | බෙහෙව, විශේෂයෙන් ක්ෂීරපායී සතුන්ගේ ස්ත්‍රී ලිංග |

Figure 2-4: Synset table - Sinhala Wordnet Database

For the purpose of this research; it is sufficient to use the “Restricted List” defined in the paper “Evaluating a Machine Learning Approach to Sinhala Morphological Analysis” [2].

Accuracy level they have managed to gain in that research paper [2] for the restricted list as shown in Table 2-1, is in the acceptable range at the moment. However, in case of modification is required we might have to consider making alterations to the algorithm.

Table 2-1: Accuracy of the algorithm against the Restricted

| POS category | Exact | Stem | # Morph |
|---------------------|--------------|--------------|----------------|
| Nouns | 35.30 | 63.10 | 51.38 |
| Verbs | 29.15 | 37.76 | 56.38 |
| Adjectives | 69.36 | 69.39 | 69.39 |
| Adverbs | 22.93 | 28.66 | 33.44 |
| Function Words | 68.23 | 69.57 | 72.35 |
| Overall | 35.05 | 56.37 | 51.38 |

This research project is heavily dependent on above-mentioned research and work such as Sinhala Wordnet [1]. Due to that, the success of this project highly dependent on the success of the alterations to be done to the existing research areas in digitising the Sinhala Language.

3 Methodology

Word suggestion engine is neither a new concept nor a new idea. The basic concept behind any prediction model is to have an understanding on the probability distribution. For this probabilistic language model, we can use Markov chain with N-gram model.

3.1 Top level architecture

The top-level architecture of the proposed application is as follows (See Figure 3-1).

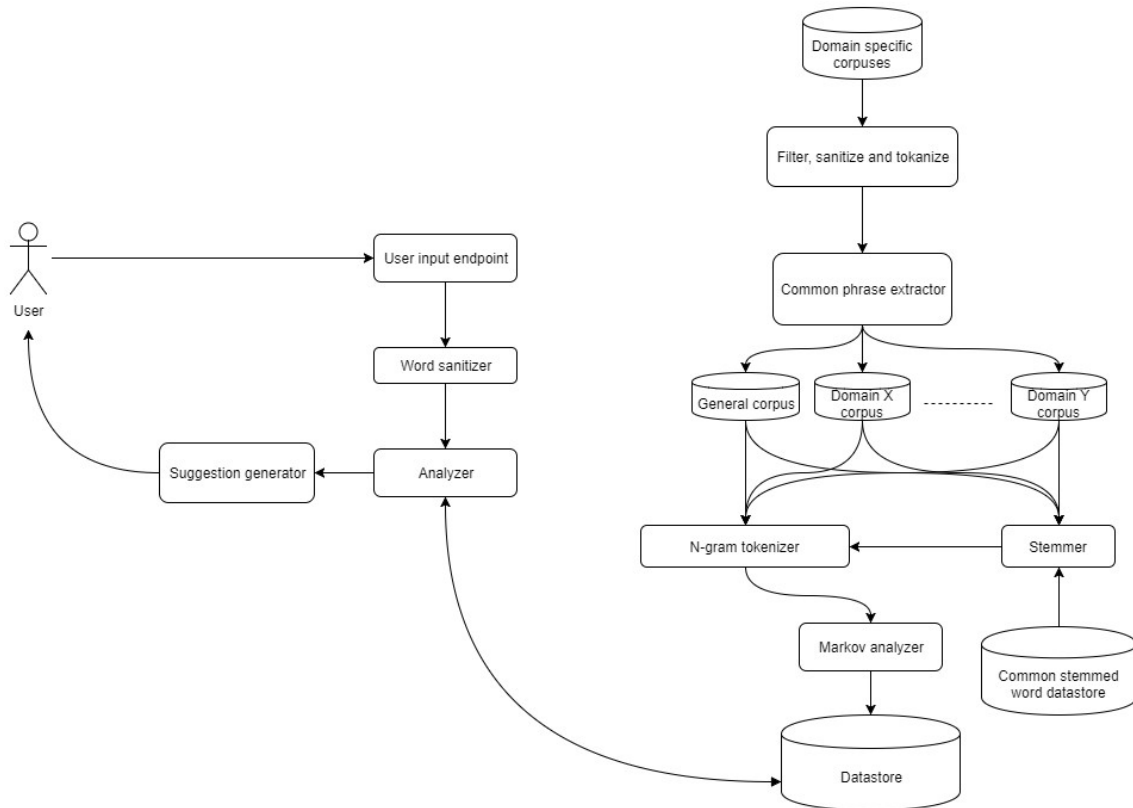


Figure 3-1: Architecture diagram

Main architecture of the program consists of four primary segments,

- Word corpus extraction module.
- Word corpus analysis module.
- Input analysis module.
- Suggestion generation module.

3.2 Data preparation for analysis

Word corpus extraction module is responsible for segregating domain-specific corpora from the general corpus. Extracted word corpus then goes through n-gram tokeniser and analysis process. N-gram generation for per corpus has two types of generation methods.

- Using the words as is without stemming
- Using the stemmed words

Consider the following two sentences.

- සමාජයීය වැදගත් තත්වයක්
- සමාජයේ වැදගත් තත්වයක්

Without stemming above two sentences will provide three bigrams.

(සමාජයීය, වැදගත්) (සමාජයේ, වැදගත්) (වැදගත්, තත්වයක්)

Having this type of bigrams is useful if the corpus have all the variations of the given word set. It also helps to result in more specific suggestions in a sentence. With stemmed n-gram, it derives bigrams as follows.

(සමාජ, වැදගත්) (වැදගත්, තත්ව)

These stemmed n-grams are useful when there are no specific words in the trained dataset. This provides more generalised suggestion for the given word.

Example: Assume a user input the word සමාජයේ but the trained dataset does not contain a bigram start with the word සමාජයේ, instead the trained dataset does contain a bigram with the word සමාජයීය. If the system configured to handle stemmed bigrams, the application will be able to provide a suggestion of the word වැදගත්. These suggestions might not provide a hundred percent accurate suggestion but will provide an acceptable suggestion.

3.3 Application of Markov chain

Markov chain describes how individual states are a sequence and the probability of a given state following another state. This algorithm helps to determine the occurrence of the next word by analysing the preselected dataset.

This will result in a map of words with a probability of next words that can occur after it. Probabilistic map derived from above analysis is used to decide which word will have a high probability of occurring after a given the word. In a Markov chain, the probability distribution of next states for a Markov chain depends only on the current state, and not on how the Markov chain arrived at the current state. This is called the Markov Property. Which means if the current state is X_t , X_{t+1} depends upon X_t and not depend upon $X_{t-1} \dots X_1$

Markov property can be explained as follows,

$$P(X_{t+1} = s \mid X_t = st, X_{t-1} = st-1, \dots, X_0 = s_0) = P(X_{t+1} = s \mid X_t = st)$$

For all $t = 1, 2, 3, \dots$ and for all states $s_0, s_1, s_2 \dots s_t, s$.

Please consider the following sentences as an example of how Markov chain to be used.

අප සමාජයේ ඇති වී තිබෙන ගැටළු

සමන් අප සමඟ පාසැල් ගිය අයෙකි

අප ජන සමාජයේ බහුතරයක් නැකත් විශ්වාස කරයි

After analysing each sentence mentioned above with Markov chain analyser, it derives a word map as illustrated in Figure 3-2.

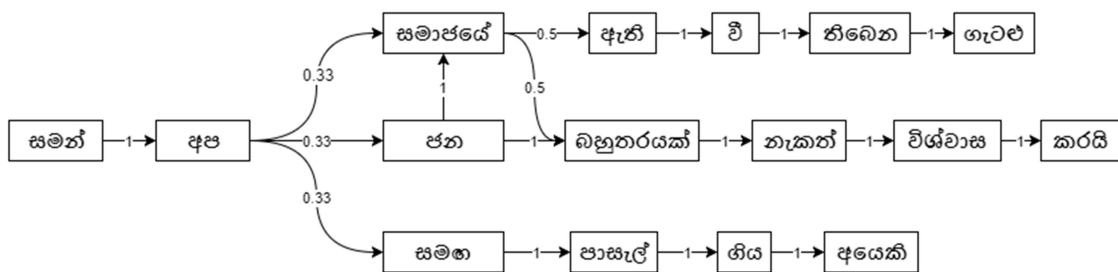


Figure 3-2: Markov Chain analysed map

As for the Figure 3-2, word අප has three possible upcoming words hence the probability of each word with the possibility of 0.33 and word සමාජයේ has 2 words with the possibility of 0.5.

When calculating the probability, it is possible to count the frequency of the same word. For example, by training the dataset with another sentence like “අප සමඟ ඒමට ඔහු එකඟ විය”; the probability of occurrence of the word සමඟ after the word අප can increase (see Figure 3-3).

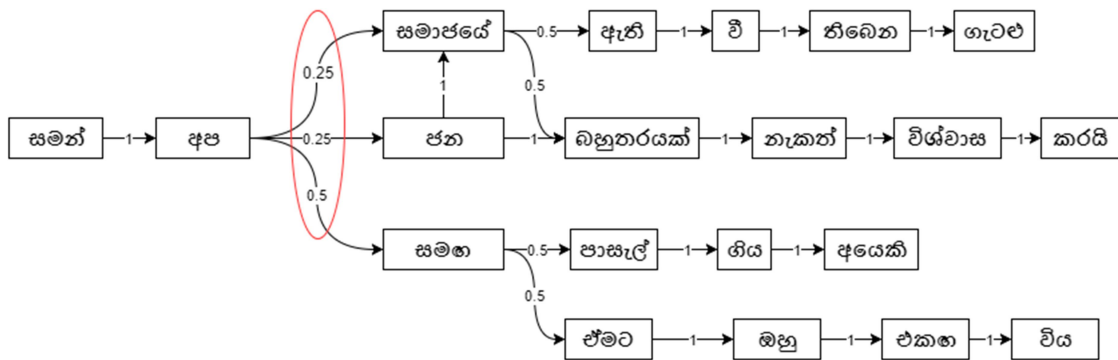


Figure 3-3: Markov Chain analysed map with frequency

Using the frequency on this model can improve the suggestion relevancy in most cases. But in some cases, this might have an adverse effect when applying context bias to the suggestion. In such cases, it is necessary to balance the weight from Markov model and context probability.

3.4 N-gram model

An n-gram is a subsequence of n items from a given sequence. Items can consist of phonemes, syllables, letters, words or any other definition depending on the application. N-gram model helps to define statistical properties of characters and words how they appear next to each other. This helps to make a more sensible suggestion. Even though this model can use with any number of sequence set (unigram, bigram, trigram, 4-gram ... n-gram), widely used models in practical applications are unigram, bigram and trigram. It is essential to identify up to what N value to be used. For most use cases 3-gram model would suffice to get a reasonable knowledge of word possibilities (ex: see Figure 3-4).

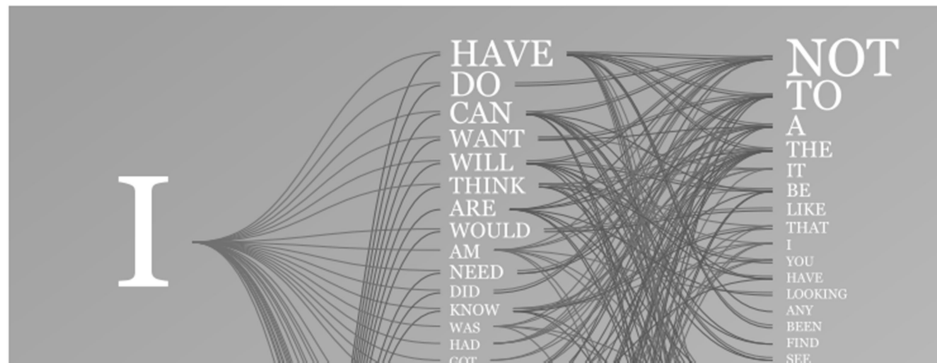


Figure 3-4: Trigram map of sentences starting with I [18]

Derivation of n-gram of a sentence can be visualised as follows.

අප සමාජයේ ඇති වී තිබෙන ගැටළු

Unigram derivation,

(අප) (සමාජයේ) (ඇති) (වී) (තිබෙන) (ගැටළු)

Bigram derivation,

(අප, සමාජයේ) (සමාජයේ, ඇති) (ඇති, වී) (වී, තිබෙන) (තිබෙන, ගැටළු)

Trigram derivation,

(අප, සමාජයේ, ඇති) (සමාජයේ, ඇති, වී) (ඇති, වී, තිබෙන) (වී, තිබෙන, ගැටළු)

In n-gram model, assume a language has T word sets in its lexicon, it predicts how likely is word x to follow the word y. This happens in two stages.

- Estimate the likelihood of x occurring at the start of a sentence based on its general frequency of occurrence estimated from the trained dataset.
- Condition the likelihood of x occurring in the context of the previous words.

In bigram, a trained model can suggest the next word by looking at the previous word and search for word sets starts with the previous word then suggest possible next word in each resulted word list.

Example:

- Step 1: type the word, අප.
- Step 2: search for word sets starts with the word, අප.
- Step 3: search results the set, (අප, සමාජයේ).
- Step 4: extract possible word suggestions, සමාජයේ.

In trigram, it looks for word sets with two words start with the previous two words in the n-gram window of the sentence. The third word of the resulting words sets are the next possible words.

Example:

- Step 1: type the word, අප සමාජයේ.
- Step 2: search for word sets starts with the words, අප සමාජයේ.
- Step 3: search results the set, (අප, සමාජයේ, ඇති).
- Step 4: extract possible word suggestions, ඇති.

As for above example, higher the n-gram word set length; higher the relevancy of the next possible word. These three models can be used independently, or as a combination of two or more, it depends on the application of the model.

3.5 N-gram with frequency

Multiple occurrences of the same word set is a good indication that it has more likelihood of occurring in a sentence. In order to incorporate frequency of the occurrences of a word set, a word set can include the number of occurrence along with words.

Example: if the phrase අප සමාජයේ appeared twice in the corpus, then the trained word set should look as follows

(අප, සමාජයේ, 2)

Hence the trigram probability of a word w_{i-1} followed by the word w_{i-2} followed by the word w_i is as followed.

$$P(w_i|w_{i-1} w_{i-2}) = \text{count}(w_i, w_{i-1}, w_{i-2}) / \text{count}(w_{i-1}, w_{i-2})$$

As well as the bigram probability of a word w_{i-1} followed by the word w_i is as followed.

$$P(w_i|w_{i-1}) = \text{count}(w_{i-1}, w_i) / \text{count}(w_{i-1})$$

3.6 Combined usage of unigram, bigram and trigram

Combination of all unigram, bigram and trigram models is achieved by an interpolation technique. This introduces a new probability as follows,

Assumption $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$

$$P(w_i|w_{i-1}, w_{i-2}) = \lambda_3 P_{ML}(w_i|w_{i-1}, w_{i-2}) + \lambda_2 P_{ML}(w_i|w_{i-1}) + \lambda_1 P_{ML}(w_i)$$

With a large enough corpus, above calculation introduce too much time complexity when comparing to the model's objectives. To address this issue, it is ideal to use "Stupid Backoff" technique.

If a sentence does not have any words, it is impossible to predict words using trigram or bigram models. In such cases, unigram can be used to suggest a word which is highly likely to occur at the beginning of a sentence. To achieve that it is required to flag unigrams which can occur at the beginning of a new sentence. Bigrams can be applied in a situation where trigram fails to provide any suggestions.

Example: Assume that the following sentence was trained with both bigram and trigram models.

අප ජන සමාජයේ බහුතරයක් නැකත් විශ්වාස කරයි

Then if the input phrase is අප බහුතරයක්, trigram model fails to suggest a word. But if this phrase analyzed with bigram model, it is possible to suggest the word නැකත්. Even though bigram suggestions might have lower relevancy than trigram suggestions, it is useful to have some suggestions than no suggestions.

3.7 Probabilistic Smoothing

Probabilistic smoothing is used when the analyser encounters a word that is not in the trained model. In such scenario, the model will assign the probability calculated from defined smoothing algorithm.

“Kneser-Ney (K-N) Smoothing” algorithm is a viable algorithm since it is mostly used in interpolated forms.

K-N smoothing algorithm for bigram model can be defined as follows,

Where $P_{con}(w_i)$ is the continuation probability of w_i , θ is a normalizing constant and δ is a discount weight.

$$P_{kn}(w_i|w_{i-1}) = [\max(\text{count}(w_{i-1}, w_i) - \delta, 0) / \text{count}(w_{i-1})] + \theta(w_{i-1}) * P_{con}(w_i)$$

Applying smoothing will balance the occurrence of unseen phrases with high-frequency words in the corpus. This means it reduces the probability of occurrence of high-frequency words in every situation where unknown word or phrase is found.

3.8 Dataset and training

To improve the general Sinhala suggestion, it is necessary to provide large enough dataset as the training data.

UCSC Sinhala News Corpus provided by University of Colombo Language Technology Research division with around 500,000 words is used as the primary corpus. From this 20% of the corpus is used as the testing data and 80% of the corpus is used as the training data.

In the preparation of this dataset, it was necessary to filter and sanitise the corpus. There were some documents does not have correct encoding, some documents with HTML/XML tags and some documents with mistyped words that had to be filtered before train the models.

Since this project uses all unigram, bigram and trigram models, it was required to train all three models before testing the model.

3.9 Informal and mix language support

The Sinhala language in day to day use is different from the formal use of Sinhala language. It is important to include not only written Sinhala language but general Sinhala sentences collected from Twitter, blogs etc. in the word corpus. This includes some English words that could be used in the middle of the sentences (sentences use IT sector, medical sector etc.). But in this project, these kinds of words were not trained.

3.10 Adding context bias

Once the sample language model is built, it is required to tweak the model to enhance the awareness of the context when predicting words. Example: If the user starts to type: රඹේ පැතිරයන යන ___ the next most suitable word is depends on what context user is typing, if it is medical or news related most likely to be ලෙඩ රෝග rather than the word කටකතා.

Context biasing can be applied in two situations.

- When the model can be configured to a particular domain (medical, news, general)
- When the model has enough usage data of the user.

The first option is used when this model is applied as a search module for a certain website or application where the domain is known. The second option is used in situations where the model can collect historical data about user's writing patterns. For example, if this model is applied to a word processor, then this model can detect what kind of articles a user is generally writing. Implementing that sort of a model is out of scope in this project.

When suggesting words in any domain, there are words which are typical for general writing. Hence it is acceptable to extract those words or phrases into a separate domain.

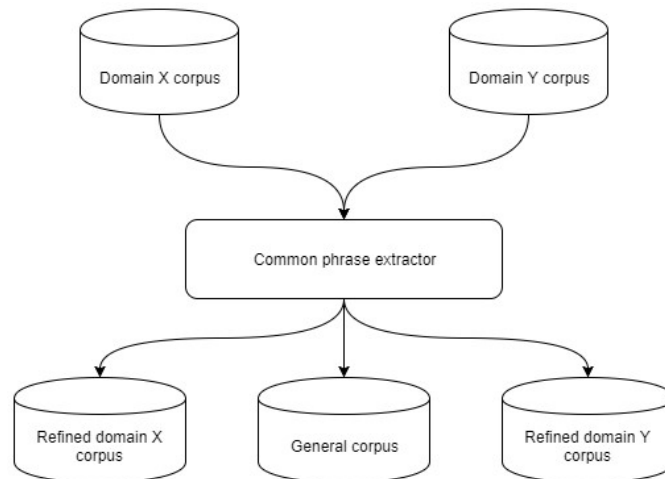


Figure 3-5: Common phrase extraction

Common phrase extractor module cross-reference domain corpuses provided and extract generalised word corpus. How each module is coupled has illustrated in Figure 3-5 above.

When the analyser performing suggestions first it queries a specific domain and then queries the general domain. This provides a bias towards the selected domain while maintaining a general aspect of the suggestion.

Since the main word corpus is consists of news articles, it is hard to demonstrate the actual implementation of the context biasing with only using the main corpus. Two main corpora are used to address this issue.

- Word corpus with news articles.
- Word corpus with medical articles.

However, since the medical article corpus is significantly smaller (5k words) than the news articles, only a selected number of sentences can be used in the evaluation process.

4 Implementation

In the implementation of this project, as the programming language Python is used. Reason being when it comes to natural language processing, Python has high verticality and flexibility as well as a number of libraries available for natural language processing and machine learning. As for the data stores, three database types are used.

- MongoDB
- SQLite 3
- Redis (in memory)
- Elasticsearch

Redis is used as an in-memory database for caching meanwhile SQLite 3 used as the main database. MongoDB is used as a document store for intermediate text processing. Third party libraries such as NLTK is used to manipulate data and used as helper functions in various algorithms.

4.1 Data extraction

Data extraction happens in three stages.

- Read raw .txt files of a given corpus and extract raw text.
- Sanitize sentences of extracted raw text.
- Perform common phrase extraction on the sanitised text.

Raw text file manipulation was challenging due to different types of encodings and non-standard format. It was required to implement exceptions for non-standard text extractions. See Figure 4-1 for a non-standard sample raw text file.

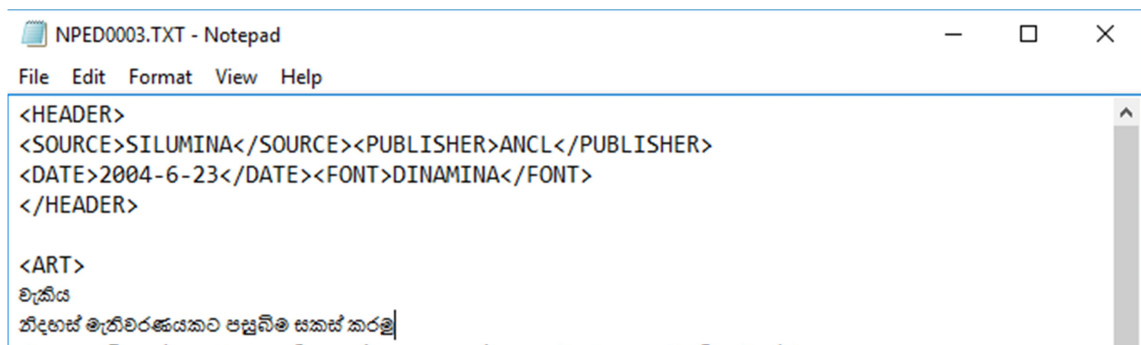


Figure 4-1: Non-standard text file

In this process, each file gets read and sanitised by a filtering process in order to remove any mistyped or formatted words or phrases. See Figure 4-2 for an un-sanitized text example.

සහිතව පැවතීම ඉතා අවදානම් සහිත බවත් කලක් ගත වන විට අත දිගේ වේදනාව ආදිය ඇතිවන බවත් ඇය පවසන්නී ය. මෙවැනි රෝගීන් දේශීය ඖෂධ ප්‍රතිකාර යොදාගෙන වරු තුනකින් සුවකරනු ලැබිය හැකි බව ද වෛද්‍යවරිය පවසයි. මෙම කශේරුකා පද්ධතියේ හටගන්නා ස්කෝලියෝසිස් <FRWD>(SCOLIOSIS)</FRWD> නැත්නම් කොළ ඇට පෙළ ඇදවීමේ රෝගයට නිසි ප්‍රතිකාර වෛද්‍යවරිය විසින් ම සොයා ගෙන තිබේ. අර්ධාවකුඹුළු ලෙස ද මෙම රෝගය හඳුන්වනු ලැබේ. මෙම රෝගය පිළිබඳ කාර්යක් නිසිසේ සිදු කරන ලද අධ්‍යයනයන්ගෙන් පසුව ඒ සඳහා සුදුසු ආයුර්වේද ප්‍රතිකාර ඇය විසින්

Figure 4-2: Un-sanitized text

These sanitised sentences are then stored in the document based database (MongoDB) with the tag of the domain for further processing.

When extracting bigrams or trigrams edge phrases are ignored. For example, if two sentences, “ජනනා මනාපය ලැබිණ. දැන් ඔහුට ජනනා ජීර්ණ අසන්නට ලැබෙන්නේ අර නිලධාරියා ගෙනි.”, bigrams like “ලැබිණ. දැන්” and trigrams like “මනාපය ලැබිණ. දැන්” are ignored because those are belongs to two separate sentences.

Names with initials and numbers are also problematic when processing Ngrams. Names with initials can confuse the extractor when breaking the sentences. For example, “මහාචාර්ය විමල් ජී. බලගල්ලේ”, to avoid extractor breaking the phrase “මහාචාර්ය විමල් ජී.” as a sentence, it is necessary to explicitly remove such words. To achieve this following regex is used. “(\s\p{Sinhala}{1,2}\s)+”. But above regex will also match legitimate single word line endings such as “ඬි.”. This confusion can be avoided by appending a special character to the legit single word line endings and revert it back once illegal line endings are removed.

After extracting corpora, generalisation process starts. In this process, each bigram and trigram goes through a cross-referencing process to identify common bigrams and trigrams. Those identified bigrams and trigrams then stored in a general corpus and excluded from other domain corpora.

4.2 Structuring of data

Data is used in unigram, bigram and trigram models. For each model, it is required to store the dataset in easy to access structure with Meta information such as domain. Database table structures for unigram, bigram and trigram are shown in Table 4-1, Table 4-2 and Table 4-3 respectively.

Unigram

Table 4-1: Unigram structure

| | | | |
|------|-----------|--------|----------------|
| Gram | Frequency | Domain | Is start point |
|------|-----------|--------|----------------|

Bigram

Table 4-2: Bigram structure

| | | | | | |
|-----------|------------|-------------|--------|-----------|--------|
| Gram hash | First gram | Second gram | Weight | Frequency | Domain |
|-----------|------------|-------------|--------|-----------|--------|

Trigram

Table 4-3: Trigram structure

| | | | | | | |
|-----------|------------|-------------|------------|--------|-----------|--------|
| Gram hash | First gram | Second gram | Third gram | Weight | Frequency | Domain |
|-----------|------------|-------------|------------|--------|-----------|--------|

Trigrams, bigrams are stored separately but when the application query for string matching, the system analyses both databases for matches. To speed up the search query, system stores most frequently queried phrases in the Redis cache.

4.3 Training

In this phase, each domain gets trained using above mentioned models to populate the knowledge base. All punctuations are ignored, but a full stop is analysed as an exception because it provides information on words that can start a sentence. These words are flagged as “Is start point”.

4.4 Analysis

A combined version of n-gram models (unigram, bigram, trigram) is used to implement layers of the analyser module as described in the methodology chapter. An abstract diagram of the analyzing workflow has shown in the Figure 4-3.

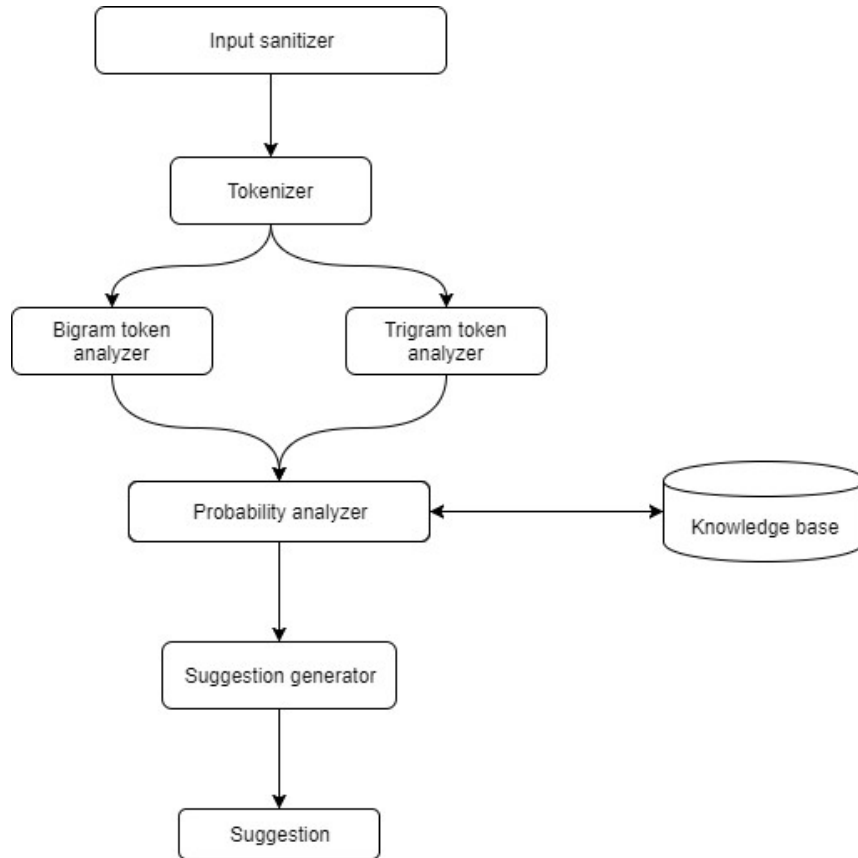


Figure 4-3: Analyzer workflow

In each analysis, first, the query string is tokenised. Query string then gets sanitised to remove any miscellaneous code fragments. Once the query string is cleaned, it passed to both bigram and trigram analysers. Above analysers generate respected bigrams and trigrams and then pass the Ngrams to probability analyser.

Probability analysis happens in two steps.

- Matching Ngram query
- Analyze probability of occurrence

4.5 Matching Ngrams

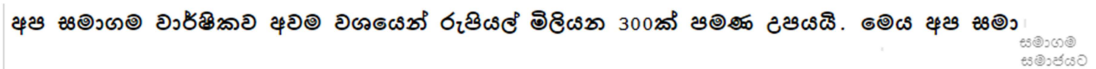
This phrase is to match bigrams or trigrams in the database. In this matching process, trigrams will have a higher weight than bigrams since a matching trigram implies a better match than a bigram. If the session holds enough information to determine the domain or the context of the user query, then the higher weight is given to the corresponding matches. If not Ngram matching considers the query as a general query.

4.5.1 General query analysis

When calculating the relevancy of multiple matches, this system uses term frequency/inverse document frequency (TF/IDF) methodology. Not every term in a match has the same weight. Some are more important than others. Relevancy of a match depends on two factors

- Term frequency: How often does a term appear in the user paragraph, more often means more relevant.
- Inverse document frequency: How often the term indexed, more often means less relevant.

For example, as for the document set trained for this example, for the bigram “මෙය අප” usually system suggests the word “සමාජය”. But since the word “සමාගම” has a higher frequency in the user document, the system gives more weight to the word “සමාගම” in the suggestion section (see Figure 4-4).



අප සමාගම වාර්ෂිකව අවම වශයෙන් රුපියල් මිලියන 300ක් පමණ උපයයි. මෙය අප සමා

Figure 4-4: Example for TF/IDF processing

4.5.2 Analyze probability of occurrence

Once there is a match, system cross-references the number of occurrences of each domain databases. This data is stored in a session data table to calculate the context of the writing later on search queries from the same session.

4.6 Suggestion engine frontend implementation

To capture the user input, it is necessary to have an easy to use UI for the end user. For an illustration of the front-end UI, see Figure 4-5.

Contextual Sug

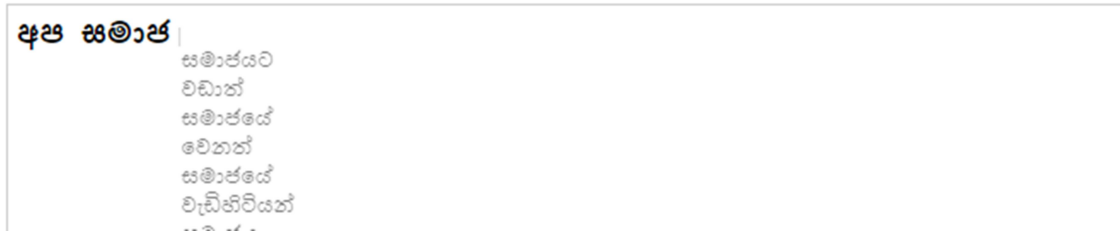


Figure 4-5: Word suggestion UI

This UI gives users suggestion as they type in the text box. More relevant suggestions will be shown in a drop-down menu where the user can select by pressing the tab key to select the most top suggestion or navigating using arrow keys and then tab to select a word from the drop-down.

Once the user press tab, auto-completion can happen in two different ways.

1. Add the suggested word to the end of the phrase
2. Complete the currently typing word

4.6.1 Add the suggested word to the end of the phrase

When the user gets suggestions, the user wants to append a suggested word as the next word. Refer Figure 4-6: Autocomplete (before adding the new word) and Figure 4-7: Autocomplete (after adding a new word).

CoI



Figure 4-6: Autocomplete (before adding the new word)

අප සමාජයට වඩාත් අවශ්‍ය |
වි
අවශ්‍ය

Figure 4-7: Autocomplete (after adding a new word)

4.6.2 Complete the currently typing word

There are situations where the query is a prefix phrase of a suggestion. In these cases when the user press tab, user expect to complete the word with that suggested word rather than append that word to the end of the phrase the user is currently typing. The application should identify whether the suggested word is a completion of a word that the user is currently typing or not and complete it appropriately. Refer Figure 4-8 and Figure 4-9.

C,

අප සමාජ |
සමාජයට
වඩාත්
සමාජයේ

Figure 4-8: Autocomplete currently typing the word (before)

Co

අප සමාජයට |
වඩාත්
ඉසිවර

Figure 4-9: Autocomplete currently typing the word (after)

5 Evaluation

Evaluation methods for this suggestion engine are experiment and interview based.

5.1 Quantitative evaluation

The problem can be treated as a binary classification problem. In this approach, Total of 176929 sentences with 2311816 trigrams in the USCS Sinhala news corpus and medical related articles from surgery.lk are used as the main dataset. 20 percent of the main dataset sentences are randomly selected as the test dataset. The test result will be evaluated by using a confusion matrix.

If the expected word exists in the prediction and the actual class, then it is considered as true positive. If the expected word is not in both prediction and the actual class, then it is considered as true negative. If the expected word exists in the prediction but not in the actual class, it is considered as false positive. If the expected word does not exist in prediction but exists in the actual class, then it is considered as false negative (refer Table 5-1: Confusion matrix structure).

Table 5-1: Confusion matrix structure

| | Actual | | |
|------------|----------|---------------------|---------------------|
| Prediction | | Positive | Negative |
| | Positive | True Positive (TP) | False Positive (FP) |
| | Negative | False Negative (FN) | True Negative (TN) |

The accuracy of the modal is obtainable by applying following formula.

$$Accuracy = \frac{\text{Total occurrences of True Positive (TP)} + \text{Total occurrences of True Negative(TN)}}{\text{All the occurances (TP + FP + FN + TN)}}$$

$$Precision = \frac{\text{Total occurrences of True Positive (TP)}}{\text{Total occurances of prediction positive (TP + FP)}}$$

Above evaluation will be tested against the Tri-gram prediction model which is used in this suggestion engine. Then the same dataset will be tested against Markov chain prediction model.

5.1.1 With news related sentences.

As for the test, by analysing the dataset with Ngram – Markov hybrid model, 35510 sentences with 426122 trigrams, (see Table 5-2: Hybrid model confusion matrix)

Table 5-2: Hybrid model confusion matrix

| | Actual | | |
|------------|----------|----------|----------|
| Prediction | | Positive | Negative |
| | Positive | 159454 | 68842 |
| | Negative | 106210 | 91616 |

$$Accuracy = \frac{251070}{426122} = 0.58919$$

$$Precision = \frac{159454}{228296} = 0.69845$$

By analysing the dataset with Markov model, 35510 sentences with 426122 trigrams, (see Table 5-3: Markov model confusion matrix)

Table 5-3: Markov model confusion matrix

| | Actual | | |
|------------|----------|----------|----------|
| Prediction | | Positive | Negative |
| | Positive | 152636 | 205049 |
| | Negative | 15982 | 52455 |

$$Accuracy = \frac{205091}{426122} = 0.48129$$

$$Precision = \frac{152636}{357685} = 0.42673$$

As for the results, Ngram-Markov hybrid model has a slightly higher accuracy than Markov model approach. However, there is an apparent gap between two models when it comes to the precision. Ngram – Markov hybrid model tend to provide more relevant results than the Markov model.

5.1.2 With medicine related sentences.

As for the test, by analysing the dataset with Ngram – Markov hybrid model, 12300 sentences with 158357 trigrams, [see Table 5-4: Hybrid model confusion matrix (medicine related)]

Table 5-4: Hybrid model confusion matrix (medicine related)

| | Actual | | |
|------------|----------|----------|----------|
| Prediction | | Positive | Negative |
| | Positive | 42587 | 37373 |
| | Negative | 52890 | 25507 |

$$Accuracy = \frac{68094}{158357} = 0.43$$

$$Precision = \frac{42587}{79960} = 0.53260$$

By analysing the dataset with Markov model, 12300 sentences with 158357 trigrams, [see Table 5-5: Markov model confusion matrix (medicine related)]

Table 5-5: Markov model confusion matrix (medicine related)

| | Actual | | |
|------------|----------|----------|----------|
| Prediction | | Positive | Negative |
| | Positive | 43896 | 61853 |
| | Negative | 37930 | 14678 |

$$Accuracy = \frac{58574}{158357} = 0.36988$$

$$Precision = \frac{43896}{105749} = 0.4150$$

When using medicine related sentences, both accuracy and precision are dropped in in both hybrid and Markov models. So this drop could be a result of having proportionally small dataset of medical articles relative to news articles.

5.2 Qualitative evaluation

As a qualitative assessment of this suggestion engine, a web application which is using the suggestion will be presented to 30 - 40 random candidates who have computer literacy with random backgrounds. Evaluation will have two parts.

- Present the suggestion web application to users and ask them to type Sinhala sentences.
- Present a questioner and collect their feedback on their experience. Each question in the questioner will have a 1 to 5 rating which users can select rating according to their experience with the system.

Questioner consisted of 5 rating criteria as follows,

- How often you see the word you wanted to type in the suggested list?
- How often you see the word you wanted to type at the top of the list?
- How relevant the suggestions you saw in the list the type of document you were typing?
- How useful you find these suggestions are to type a document faster?
- How do you rate your overall user experience with this tool?

Following results had gathered after receiving 31 feedbacks out of 40 requests sent to candidates from various fields of interests (see Table 5-6: Feedback participation).

Table 5-6: Feedback participation

| Field | Number of participants |
|----------------------|-------------------------------|
| News and journalism | 12 |
| Medical | 10 |
| Other (general user) | 9 |

Criteria 1: How often you see the word you wanted to type in the suggested list? (For results refer Table 5-7: Criteria 1 average feedback)

Table 5-7: Criteria 1 average feedback

| Field | Average rating (out of 5) |
|----------------------|----------------------------------|
| News and journalism | 4.26 |
| Medical | 3.17 |
| Other (general user) | 4.63 |

Criteria 2: How often you see the word you wanted to type at the top of the list? (For results refer Table 5-8: Criteria 2 average feedback)

Table 5-8: Criteria 2 average feedback

| Field | Average rating (out of 5) |
|----------------------|----------------------------------|
| News and journalism | 3.12 |
| Medical | 2.18 |
| Other (general user) | 3.81 |

Criteria 3: How relevant the suggestions you saw in the list the type of document you were typing? (For results refer Table 5-9: Criteria 3 average feedback)

Table 5-9: Criteria 3 average feedback

| Field | Average rating (out of 5) |
|----------------------|----------------------------------|
| News and journalism | 3.65 |
| Medical | 3.31 |
| Other (general user) | 4.17 |

Criteria 4: How useful you find these suggestions are to type a document faster? (For results refer Table 5-10: Criteria 4 average feedback)

Table 5-10: Criteria 4 average feedback

| Field | Average rating (out of 5) |
|----------------------|----------------------------------|
| News and journalism | 2.89 |
| Medical | 3.92 |
| Other (general user) | 4.57 |

Criteria 5: How do you rate your overall user experience with this tool? (For results refer Table 5-11: Criteria 5 average feedback)

Table 5-11: Criteria 5 average feedback

| Field | Average rating (out of 5) |
|----------------------|----------------------------------|
| News and journalism | 4.36 |
| Medical | 4.11 |
| Other (general user) | 4.66 |

5.3 Tools and technologies used in the evaluation.

As special tools for this evaluation following will be used,

- USCS Singlish Unicode converter will be used to make Sinhala sentence typing easier for users.
- Web-based interface to collect user feedback
- Web sockets and socket IO to communicate between frontend UI with backend services.

6 Conclusion and future work

In summary, most of the conventional techniques used in languages such as English to predict the next possible word can be used in the Sinhala language as well. However, it is necessary to change the features of these model such as how the model extract the words and analyse, in a way that suitable for the Sinhala language.

Both quantitative and qualitative attributes of the project is at a satisfactory level, but it requires more comprehensive and more extensive dataset to conclude whether this can be applied in real-world scenario.

Most noticeable downfall of this suggestion engine is that it does not provide auto-correction or suggestions for incorrectly spelt words. So the spelling detection and correction model integration for this model can increase the usability and the practicality of this application. Another improvement for this suggestion engine is to provide fuzzy search when analysing the user query.

7 References

- [1] Viraj Welgama, Dulip Lakmal Herath, Chamila Liyanage, Namal Udalamatta, and Ruwan Weerasinghe. (2011, May) <http://www.cle.org.pk>. [Online]. <http://www.cle.org.pk/hltd/pdf/HLTD201107.pdf>
- [2] Viraj Welgama, Ruwan Weerasinghe, and Mahesan Niranjana. <http://ltrc.iiit.ac.in>. [Online]. <http://ltrc.iiit.ac.in/icon/2013/proceedings/File23-paper42.pdf>
- [3] B Hettige. (2010, December) <http://staffweb.sjp.ac.lk>. [Online]. <http://staffweb.sjp.ac.lk/sites/default/files/budditha/files/thesis.pdf>
- [4] Asanka Wasala, Ruwan Weerasinghe, Randil Pushpananda, Chamila Liyanage, and Eranga Jayalatharachchi. (2010, December) <https://icter.sljol.info>. [Online]. <https://icter.sljol.info/articles/abstract/10.4038/icter.v3i1.2844/>
- [5] Internet Live Stats. (2017, June) Internet Live Stats. [Online]. <http://www.internetlivestats.com/internet-users/us/>
- [6] Allen Robert. (2017, July) www.smartinsights.com. [Online]. <http://www.smartinsights.com/search-engine-marketing/search-engine-statistics/>
- [7] Takeshi Matsumoto, David M. W. Powers, and Geoff Jarrad. www.aclweb.org. [Online]. <http://www.aclweb.org/anthology/U03-1003>
- [8] Danny Sullivan. searchengineland.com. [Online]. <http://searchengineland.com/how-google-instant-autocomplete-suggestions-work-62592>
- [9] Wikipedia. Wikipedia. [Online]. https://en.wikipedia.org/wiki/Levenshtein_distance
- [10] Wikipedia. Wikipedia. [Online]. <https://en.wikipedia.org/wiki/Metaphone>
- [11] Wikipedia. Wikipedia. [Online]. <https://en.wikipedia.org/wiki/Soundex>
- [12] Elastic.co. Elastic.co. [Online]. <https://www.elastic.co/products/elasticsearch>
- [13] Alexander Vlasblom. (2015, July) Text Prediction Using N-Grams. [Online]. https://rstudio-pubs-static.s3.amazonaws.com/96252_bd61a0777ad44d04b619ce95ca44219c.html

- [14] Gabriella Pasi. mumia-network.eu. [Online]. http://www.mumia-network.eu/index.php/digital-library/doc_download/134-gabriella-pasi-contextual-search-and-contextual-factors-aggregation
- [15] Asanka Wasala, Ruvan Weerasinghe, Randil Pushpananda, Chamila Liyanage, and Eranga Jayalatharachchi. <http://subasa.ambitiouslemon.com/spellerweb.py>. [Online]. <http://subasa.ambitiouslemon.com/spellerweb.py>
- [16] B Hettige. (2010, December) <http://staffweb.sjp.ac.lk>. [Online]. <http://staffweb.sjp.ac.lk/sites/default/files/budditha/files/thesis.pdf>
- [17] Ruvan Weerasinghe. (2016, September) www.researchgate.net. [Online]. https://www.researchgate.net/publication/268257661_A_Morphological_Parser_for_Sinhala_Verbs
- [18] Chris Harrison. <http://www.chrisharrison.net>. [Online]. <http://www.chrisharrison.net/index.php/Visualizations/WebTrigrams>
- [19] (2018, Feb) [digitalmarketer.lk](http://www.digitalmarketer.lk). [Online]. <http://www.digitalmarketer.lk/internet-usage-statistics-in-sri-lanka-2016-updated.html>