ABSTRACT

A spell checker is a tool that finds and corrects erroneous words and grammatical mistakes in a text document. Spelling error detection and correction techniques are widely used by text editing systems, search engines, text to speech and speech to text conversion systems, machine translation systems and optical character recognition systems. The spell checkers for European languages and some Indic languages are well developed. However, perhaps, owing to Tamil being a morphologically rich and agglutinative language this has been a challenging task. Erroneous words and grammatical mistakes can occur in sentences due to various reasons. Erroneous words can be classified into two categories, namely non-word errors and real-word errors. This work aims to correct non-word errors in Tamil documents by suggesting alternatives. The proposed approach uses letter-level and word-level n-gram, stemming and hash table techniques. Test results show that the suggestions generated by the system are with 95% accuracy.

Keywords: Tamil; spell checker; N-gram; erroneous word.

1. INTRODUCTION

Tamil language possesses a rich history of more than 2000 years and spoken primarily in Sri Lanka, India, Malaysia and Singapore. Tamil words are made up of morphemes (smallest meaningful units) concatenated to one another in a series. More often, such lexical roots may be accompanied by one or more affixes. Tamil verbs inflect with Person, Number and Gender marking, and combine with auxiliaries that indicate aspect, mood, causation, attitude etc.
whereas Tamil nouns inflect with plural markers, case markers, clitics etc. [1,2,3]. The agglutinative nature makes Tamil a complex language to process using computers. Thus, developing a spell checker for Tamil language is a challenging task.

Spell checker is a tool to check the spelling of words in a document, validate them, and list out the possible correct words as suggestions in case of finding error in spelling. A grammar checker is a tool to check each sentence in a text to see whether it conforms to the grammar. In case it finds a structure that conflicts the conformity to the grammar, it would give suggestions for alternatives [4]. The problem of devising algorithms and techniques to automatically correct words in texts has become a research challenge. Spelling error detection and correction techniques are widely used by text editing systems, search engines, text to speech and speech to text conversion systems, machine translation systems, speech recognition systems and optical character recognition systems. Spell checkers for European languages such as English [4] and some Indic languages are well developed. However, only a few efforts have been made to produce spell checker for Tamil language, perhaps, because the fact that Tamil language is morphologically rich and agglutination, making it a challenging task. Moreover, scarcity of digital resources such as Tamil lexicon, tagged corpora, morphological analysers and generators might also have caused a setback for proceeding in such works successfully.

Erroneous words can be classified into two categories, namely non-word errors and real-word errors [5,6,7,8]. Non-word errors fall into two sub-categories: First one is that the word itself is invalid (e.g. மேலெட்டன்) and the other is that the word is valid (e.g. நவாஸ்தாலி) but not present in a valid lexicon. Real-word error (e.g. கூறுகு சந்தாளம் மொழியும்படுத்தல்)) means the word is valid but inappropriate in the context of the sentence. The non-word and real-word errors can occur in sentences when typing a document due to fast typing, switching of fingers on keys, input tools and method, or not knowing the right pronunciation, correct spelling or the meaning of the word. The sentence level should be considered when detecting the real-word errors and providing suggestions for both non-word and real-word errors in sentences.

In this paper, we focus on correcting non-word errors in Tamil language. We propose a method that uses n-gram technique on stemmed form of the words to generate suitable alternatives to misspelt words while making use of a hash table to speed up the lookup. After this introductory section the rest of this paper is organised as follows: Section 2 describes our methodologies. Test results are discussed in Section 3. Finally, Section 4 concludes this paper.

2. METHODOLOGY

The aim of this work is to correct non-word errors in Tamil documents by suggesting alternatives. The error detection process makes use of tree representation of the lexicon efficiently, as explained in [9]. For the words detected as incorrect words, the suggestions are given as a way of correcting non-word errors. For suggestion generation, three methods are experimented and compared with respect to time taken and number of appropriate suggestions. The three methods are

- Method-1: Tree-based lookup algorithm for detection and letter-level bigram technique with stemming for correction
- Method-2: Tree-based lookup algorithm for detection and letter-level bigram technique with stemming and hash table for correction
- Method-3: Tree-based lookup algorithm for detection and letter-level trigram technique with stemming and hash table for correction

After generating the possible suggestions for erroneous words, the appropriate suggestions based on the context are extracted from the possible suggestions as per the ranks by word-level n-gram language probabilistic model.

The techniques and models are described below:

2.1 N-Gram Technique

N-gram is used to create composition of N adjacent letters of a word or N adjacent words of a sentence. An n-gram of size 1 is a unigram (that is, N=1), N=2 would produce bigrams, N=3 would produce trigrams and so on. For example, the letter-level bigrams of the word கூறுகு சந்தாளம் மொழியும்படுத்தல் would comprise of the two letter combinations கூறுகு சந்தாளம் and சந்தாளம் மொழியும்படுத்தல். Similarly, the letter-level trigrams of the word கூறுகு சந்தாளம் மொழியும்படுத்தல் would comprise of the
three letter combinations $\text{ஏம், இர், எல், ஏன், ஏம், ஏல்}$. As n-gram technique can determine the probability of adjacent letters in a word, it can be used for both error detection and correction of words [4,7,10,11]. Using the letter-level n-gram technique, the similarity between input and lexicon words is calculated using Equation 1.

$$\text{Similarity Ratio (SR)} = \frac{C_n}{T_n} \tag{1}$$

where $C_n$ is the number of common n-grams, and $T_n$ is the total number of unique n-grams from each of the pair of words to be compared.

The similarity between input word and lexicon word is calculated using Equation 1. Minimum threshold value for similarity measure considering trigrams is different from that when considering bigrams. For example, the Similarity Ratio (SR) between the correct word ‘படிக்கிண்றான்’ and incorrect word ‘படிக்கி கிண்றான்’ can be measured using bigrams and trigrams as follows:

**Using Letter-level Bigrams:**

Let $S_{2w}$ be set of bigrams of $\text{டிக் க்கிண்றான்}$.

That is, $S_{2w} = \text{டிக், க்கி, க்கு, கிண்றா, கிண்ன்}$

Let $S_{2s}$ be set of bigrams of $\text{டிக் க்கிண்றான்}$.

That is, $S_{2s} = \text{டிக், க்கி, க்கு, கிண்றா, கிண்ன்}$

**Common Bigrams of the two words** $= S_{2w} \cap S_{2s} = \text{டிக், க்கி, க்கு, கிண்றா, கிண்ன்}$

**Unique Bigrams of the two words** $= S_{2w} \cup S_{2s} = \text{டிக், க்கி, க்கு, கிண்றா, கிண்ன்}$

**Similarity Ratio** between these two words, $SR_2 = \frac{S_{2w} \cap S_{2s}}{S_{2w} \cup S_{2s}}$ as per Equation 1.

$$SR_2 = \frac{4}{8} = 0.5$$

**Using Letter-level Trigrams:**

Let $S_{3w}$ be set of trigrams of $\text{டிக் க்கி கிண்றான்}$.

That is, $S_{3w} = \text{டிக், க்கி, க்கு, கிண்றா, கிண்ன்}$

Let $S_{3s}$ be set of trigrams of $\text{டிக் க்கி கிண்றான்}$.

That is, $S_{3s} = \text{டிக், க்கி, க்கு, கிண்றா, கிண்ன்}$

**Common Trigrams of the two words** $= S_{3w} \cap S_{3s} = \text{டிக், க்கி}$

**Unique Trigrams of the two words** $= S_{3w} \cup S_{3s} = \text{டிக், க்கி, க்கு, கிண்றா, கிண்ன்}$

$$SR_3 = \frac{2}{8} = 0.25$$

While a word that has bigram Similarity Ratio greater than 0.1 is a good candidate for suggestion, a word that gives a positive trigram Similarity Ratio seems to be a good candidate for suggestion. This acceptance level is determined by analysing the suggestions with a wider range of thresholds.

**2.2 Stemming**

Stemming is a technique used to reduce the inflected words to their stem or root word form. That is, it is a process of removing whole or part of suffixes from inflectional words [12-14]. For example, the words ‘அம்மாண்’, ‘அம்மாண்டு’, ‘அம்மாண்டும்’ and ‘அம்மாண்டுன்’ are stemmed to ‘அம்மா’. A set of 23 Tamil grammar rules have been defined for the stemming process based on:

**Rule 1**: Remove a consonant (ஏ, இ, எ, ஐ, ஒ, ஓ, ஓ or வி) that comes at the end of words

**Rule 2**: Remove plural markers (ஆல், போன்)

**Rule 3**: Remove verb suffixes (ஏர், இந்தர், என், ஏதர், ஏகேன், ஏன், ஏன், ஏன், ஏன், ஏன், ஏன், ஏன், ஏன், ஏன், ஏன், ஏன், ஏன், ஏன், ஏன், ஏன், ஏன், ஏன், ஏன்)

**Rule 4**: Remove question suffixes (ஏய், வே, வே)

**Rule 5**: Remove conjunction suffix (எல்)

**2.3 Hash Table**

The time for generating suggestions for erroneous words and the number of suggestions generated depends on the lexicon size and lookup method and slows down when the size exceeds a few thousand words. Fortunately, the hash table is a good alternative to speed up the lookup. To lookup an input string, one simply finds its hash key and retrieves the word stored at that key in the pre-constructed hash table [15-17]. To reduce the search time for generating suggestions using n-gram technique, a hash table is constructed for the lexicon words using
the first-letter of the lexicon words as key. Fig.1 depicts the hash table with a selected portion. At the time of experiment, the lexicon consists of about 705,000 valid Tamil words. They are stored in the hash table and saved as a Python pickle for easy and quick restoration.

| Keys | Values |
|------|--------|
| அமர், அவ்வாள், அவர், அவுண்ட், அவுர், அவன், அவ்வாள், அவிட்டேன், அவிட்டேன், அவிட்டேன், அவிட்டேன், அவிட்டேன் | குறிக்குரிய வருண்ணவை, குறிக்குரிய வருண்ணவை, குறிக்குரிய வருண்ணவை, குறிக்குரிய வருண்ணவை, குறிக்குரிய வருண்ணவை, குறிக்குரிய வருண்ணவை, குறிக்குரிய வருண்ணவை, குறிக்குரிய வருண்ணவை, குறிக்குரிய வருண்ணவை |
| குறிக்குரிய வருண்ணவை | பரவலாயத்தின் | பரவலாயத்தின் | பரவலாயத்தின், பரவலாயத்தின் |
| பரவலாயத்தின் | பரவலாயத்தின் | பரவலாயத்தின், பரவலாயத்தின் |

Fig. 1. A depiction of entries in hash table with first-letter and length of the lexicon words as key

### 2.4 Word-level N-gram Language Probabilistic Model

Word-level n-gram language probabilistic model determines the most appropriate suggestions for erroneous words based on the context. The model is constructed for the sentences in the constructed corpus as follows:

Let the sequence of words \( w_1, w_2, w_3, \ldots, w_n \) denote the sentence in the corpus, \( () \) denote its unigram sequence \( \langle \text{($start$)}, (w_1), (w_2), (w_3), \ldots, (w_n), (\text{($end$)}) \rangle \), \( \langle (\text{($start$)}, w_1, w_2), (w_2, w_3), (w_3, w_4), \ldots, (w_{n-1}, w_n), (\text{($end$)}) \rangle \) denote its bigram sequence and \( \langle (\text{($start$)}, w_2), (w_1, w_2, w_3), (w_2, w_3, w_4), (w_3, w_4, w_5), \ldots, (w_{n-2}, w_{n-1}, w_n), (\text{($end$)}) \rangle \) denote its trigram sequence [4].

Let \( p_i \) denote a measure for a word \( w_i \) calculated using Equation 2 as instructed in [18].

\[
P_i = \lambda_1 p_i^{(1)} + \lambda_2 p_i^{(2)} + \lambda_3 p_i^{(3)}
\]  

where \( \lambda_i \)s are positive scalars such that \( \lambda_1 + \lambda_2 + \lambda_3 = 1 \),

\( p_i^{(1)} \) denotes the probability \( p_c(w_i) \) in the corpus,

\( p_i^{(2)} \) denotes the probability \( p_r(w_i|w_{i-1}) \) in the corpus, and

\( p_i^{(3)} \) denotes the probability \( p_r(w_i|w_{i-2}, w_{i-1}) \) in the corpus.

The values of \( \lambda_i \)s are tuned using a set of test sentences, and the respective values 0.05, 0.7 and 0.25 are found to be working well with the constructed corpus.

The probabilities \( p_i^{(1)}, p_i^{(2)} \) and \( p_i^{(3)} \) are calculated using Equations 3, 4 & 5 respectively.

\[
p_i^{(1)} = p_r(w_i) = \frac{\text{count of } w_i}{\text{Total no. of words in the corpus}}
\]  

\[
p_i^{(2)} = p_r(w_i|w_{i-1}) = \frac{\text{count of bigrams } (w_{i-1}, w_i)}{\text{count of } w_{i-1}}
\]  

48
After calculating the measures for all the suggestion words for a erroneous word using Equation 2, the suggestion words are ranked based on their measures and top-ranked words are picked as suggestions for the erroneous word. Up to five words that give \( p \) measure greater than the threshold value \( 10^{-6} \) are considered for suggestions. The value \( 10^{-6} \) is found to be a good threshold as being determined by analysing the words in the constructed corpus. Moreover, it is found that the number of appropriate suggestions is found to be less than five in many cases.

Let us see how to generate suggestions for an erroneous word in a sentence: Consider the sentence ‘விருந்திைர்களின் வருனاكக்காக மக்கள் காத்திருக்கிண்றார்கள்’. The word ‘காத்திருக்கிண்றார்கள்’ is found to be an erroneous word because it does not exist in the lexicon. N-gram module generates possible alternative words {‘காத்திருக்கிறார்கள்’, ‘காத்திருக்கின்றை’, ’காத்திருப்பதாக’, ‘காத்திருக்கின்றார்கள்’, ‘காத்திருக்கின்றறாம்’, ‘காத்துக்கிடக்கின்றார்கள்’, ‘காத்திருந்தார்கள்’, ‘காத்துக்கிடக்கின்றார்கள்’}. The word-level n-gram language probabilistic model ranks these suggestions based on their measures of \( p \) that are calculated using Equation 2 as follows:

\[
p_1 = p(காத்திருக்கிறார்கள்) = 2.5149 \times 10^{-1}
\]
\[
p_2 = p(காத்திருக்கின்றை) = 2.0021 \times 10^{-1}
\]
\[
p_3 = p(காத்திருப்பதாக) = 1.8332 \times 10^{-1}
\]
\[
p_4 = p(காத்திருக்கிறார்கள்) = 8.3427 \times 10^{-2}
\]
\[
p_5 = p(காத்திருக்கின்றார்கள்) = 9.1694 \times 10^{-7}
\]
\[
p_6 = p(காத்துக்கிடக்கின்றார்கள்) = 3.3445 \times 10^{-7}
\]
\[
p_7 = p(காத்திருந்தார்கள்) = 2.5779 \times 10^{-7}
\]

Fig. 2. Non-word error detection and correction steps using Method-1, Method-2 and Method-3
After determining the $p$ measures of these suggestions, up to five words with measure more than a predetermined threshold $10^{-6}$ are picked as suggestions for the erroneous word. In the above example, the words ‘காத்திருக்கின்றார்கள்’, ‘காத்திருக்கிறார்கள்’, ‘காத்திருந்தார்கள்’, ‘காத்துக்கிடக்கின்றார்கள்’, ‘காத்திருக்கிறார்கள்’ would be selected as the appropriate suggestions based on the context for the erroneous word.

Fig. 2 shows the flow diagram as a whole including Method-1, Method-2 and Method-3.

3. RESULTS AND DISCUSSION

During the detection and correction processes, the erroneous words, suggestion words and the time taken to detect erroneous words and to generate suggestions for them by each of the generator modules are recorded. Each of these methods is experimented with input texts of 10000, 20000, 30000, 40000, 50000, 60000, 70000, 80000, 90000 and 100000 words forming ten experiments. The input words were picked randomly from various resources, including Tamil news websites, online Tamil articles etc. These words are obtained from sources that are not part of the sources used to build up the lexicon and corpus.

Table 1 shows the details of

| (A) | (B) | (C) | (D) | (E) (minutes) | (F) (minutes) | (G) (minutes) |
|-----|-----|-----|-----|--------------|--------------|--------------|
| 10000 | 736 | 209 | 527 | 1.57×60 | 10.49 | 10.11 |
| 20000 | 2499 | 237 | 2262 | 5.29×60 | 34.47 | 32.47 |
| 30000 | 2641 | 275 | 2366 | 5.66×60 | 36.44 | 30.66 |
| 40000 | 3194 | 307 | 2887 | 6.42×60 | 39.32 | 29.74 |
| 50000 | 3534 | 323 | 3211 | 6.69×60 | 43.09 | 35.53 |
| 60000 | 4281 | 374 | 3907 | 8.25×60 | 52.48 | 43.79 |
| 70000 | 4682 | 401 | 4281 | 9.89×60 | 57.67 | 50.84 |
| 80000 | 5305 | 412 | 4893 | 11.01×60 | 1.1×60 | 59.01 |
| 90000 | 5975 | 434 | 5541 | 12.75×60 | 1.34×60 | 1.13×60 |
| 100000 | 6768 | 471 | 6297 | 15.07×60 | 1.44×60 | 1.25×60 |

Table 1. Time taken to detect erroneous words and generate suggestions for them by each method

Time taken to detect and correct erroneous words using Method-1

Time taken to detect and correct erroneous words using Method-2

Time taken to detect and correct erroneous words using Method-3.

Times taken by all these three methods are shown as graphs in Fig.3 that shows the time taken by letter-level bigram technique with stemming in hours, and in Fig.4 that shows the time taken by letter-level bigram technique with stemming and a hash table and letter-level trigram technique with stemming and a hash table in minutes.

When considering the time taken, Method-1 takes more time than Method-2 and Method-3 in all the experiments, and Method-2 takes more time than Method-3 in the same experiments. For example, as shown in Table 1, for the input set of 100,000 words, 6768 input words are detected as incorrect, Method-1 takes 15.07 hours, Method-2 takes 1.44 hours and Method-3 takes 1.25 hours. In terms of time, Method-3 seems to be better than Method-2, and Method-1 will be the least preferred among these three methods.

On the other hand, when considering the number of appropriate suggestions based on the context, Method-2 gives more number of appropriate suggestions for erroneous words of length up to five whereas Method-3 gives more number of appropriate suggestions for erroneous words of length greater than five. The following results of the experiments yield this conclusion.
Table 2 shows 20 randomly selected erroneous words, the length of erroneous words and the number of appropriate suggestions based on the context selected from up to five top ranked suggestions that are generated by each of the methods. The up to five ranked suggestions are identified by using word-level n-gram language probabilistic model. In this table, column 1 shows the length of erroneous words and columns 3, 4, & 5 show the number of appropriate suggestions based on the context generated by Method-1, Method-2 and Method-3 respectively for the randomly selected erroneous words from our experiments. The same output is depicted as column chart in Fig. 5.
Table 2. Performance comparison of bigram and trigram for suggestion generation with length of erroneous words

| Erroneous Length of erroneous words | No. of appropriate suggestions based on the context |
|------------------------------------|---------------------------------------------------|
|                                    | Letter-level bigram technique with stemming | Letter-level bigram technique with stemming and a hashtable | Letter-level trigram technique with stemming and a hashtable |
| Word 1 15 | 2 | 2 | 4 |
| Word 2 11 | 1 | 3 | 3 |
| Word 3 11 | 2 | 3 | 4 |
| Word 4 10 | 1 | 1 | 3 |
| Word 5 10 | 3 | 4 | 4 |
| Word 6 9  | 1 | 3 | 3 |
| Word 7 9  | 1 | 2 | 3 |
| Word 8 8  | 2 | 2 | 3 |
| Word 9 8  | 2 | 3 | 4 |
| Word 10 8 | 1 | 1 | 3 |
| Word 11 7 | 2 | 2 | 4 |
| Word 12 7 | 1 | 1 | 3 |
| Word 13 7 | 2 | 3 | 4 |
| Word 14 6 | 2 | 2 | 5 |
| Word 15 6 | 3 | 3 | 3 |
| Word 16 6 | 2 | 3 | 3 |
| Word 17 5 | 1 | 2 | * |
| Word 18 5 | 2 | 3 | * |
| Word 19 4 | 3 | 3 | * |
| Word 20 4 | 1 | 3 | * |

Note: * It is not sensible to use trigram for words of length less than 6 as the number of common trigrams will be of maximum two

Fig. 5. Performance analysis for suggestion generation with length of the erroneous words (numbers in the label of words indicate length of the words)
Table 3. Total no. of appropriate suggestions for 250 randomly selected erroneous words of length from 3 to 15

| Length of erroneous words | No. of erroneous words | Total number of appropriate suggestions based on the context |
|--------------------------|-----------------------|----------------------------------------------------------|
|                          |                       | Method-2 | Method-4 | Method-5 |
| 15                       | 1                     | 1        | 2        | 4        |
| 12                       | 3                     | 3        | 5        | 7        |
| 11                       | 10                    | 14       | 21       | 25       |
| 10                       | 20                    | 30       | 36       | 49       |
| 9                        | 32                    | 34       | 51       | 57       |
| 8                        | 35                    | 38       | 54       | 63       |
| 7                        | 53                    | 59       | 77       | 90       |
| 6                        | 41                    | 50       | 65       | 75       |
| 5                        | 39                    | 47       | 65       | 23       |
| 4                        | 15                    | 19       | 26       | 8        |
| 3                        | 1                     | 1        | 1        | 0        |

Table 2 suggests that Method-3 gives more appropriate suggestions based on the context than the other two methods for erroneous words of length greater than five. For words of length up to five, Method-2 gives sensible suggestions than Method-3. Moreover, results of another set of 250 randomly selected erroneous words of length 3 to 15 are shown in Table 3. In this table, column 1 shows the length of erroneous words, column 2 shows the number of erroneous words with specified length and columns 3, 4 & 5 show the total number of appropriate suggestions based on the context that are generated by Method-1, Method-2 and Method-3 respectively.

When considering the number of appropriate suggestions based on the context generated by Method-1 and Method-2, Method-1 gives less number of appropriate suggestions based on the context for many erroneous words than Method-2 in our experiment. When comparing Method-2 and Method-3 with respect to number of appropriate suggestions based on the context, Method-2 gives more number of appropriate suggestions based on the context for erroneous words of length up to five than Method-3. This is because the number of trigrams for short words would be up to only three which is not sufficient to get common trigrams with other candidate words for suggestions. Further, Method-3 gives more number of appropriate suggestions based on the context for erroneous words of length greater than five as shown in Table 3. Overall, Method-1 seems to be the least preferred method out of these three methods because it takes much more time than Method-2 and Method-3, and gives less number of appropriate suggestions based on the context. The use of hash table gives the time-gain for Method-2 and Method-3. Of these two methods, Method-3 is the most preferred method for detecting and correcting erroneous words of length greater than five and Method-2 is suitable for words of length up to five.

4. CONCLUSION

In this work, we have compared the performance of error correction modules for Tamil language. In this regard, the bigram, trigram, stemming and hash table techniques have been used and implemented, and evaluated under different sets of test words. The results indicate that the suggestions generation by the letter-level trigram technique is faster than that by the bigram technique. Moreover, the letter-level trigram technique with stemming using the hash table is better than bigram to generate suggestions for misspelt words of length greater than 5, and the letter-level bigram technique with stemming using the hash table is found to be more appropriate for short length words to generate suggestions as a way of error correction. Moreover, test results show that the suggestions generated by the system are with 95% accuracy.

COMPETING INTERESTS

Author has declared that no competing interests exist.

REFERENCES

1. Navalar A. Tamil grammar questions and answers, No. 366, Kankesanthurai Road, Jaffna: Vannai Santhayarmadam; 1998.
2. Sangar V. Tamil Grammar, Puduchcheri, India: Nanmozi Printers; 2006.
3. Nuhman MA. Basic Tamil Grammar, Kalmunai: Department of Tamil, University of Peradeniya: Readers Association; 2013.
4. Kukich K. Techniques for Automatically Correcting Words in Text. ACM Computing Survey. 1992;24(4):377-439.
5. Samanta P, Chaudhuri B. A simple real-word error detection and correction using local word bigram and trigram. Association for Computational Linguistics and Chinese Language Processing (ACLCLP), Taiwan; 2013.
6. Sakuntharaj R, Mahesan S. A novel hybrid approach to detect and correct spelling in Tamil text. 2016 IEEE International Conference on Information and Automation for Sustainability (ICIAIS), University of Ruhuna, Sri Lanka; 2016.
7. Sakuntharaj R, Mahesan S. Use of a novel hash-table for speeding-up suggestions for misspelt Tamil words. 2017 IEEE International Conference on Industrial and Information Systems (ICIIS), University of Peradeniya, Sri Lanka; 2017.
8. Sakuntharaj R, Mahesan S. Detecting and Correcting Real-word Errors in Tamil Sentences. Ruhuna Journal of Science. 2018;9(2).
9. Sakuntharaj R, Mahesan S. A novel hybrid approach to detect and correct spelling in Tamil text. 2016 IEEE International Conference on Information and Automation for Sustainability (ICIAIS), Sri Lanka; 2016.
10. Ahmed F, Luca EWD, Nurnberger A. Revised N-gram based automatic spelling correction tool to improve retrieval effectiveness. Computer Science and Computer Engineering with Applications (Polibits). 2009;40:39-48.
11. Gupta N, Mathur P. Spell checking techniques in NLP: A survey. International Journal of Advance Research in Computer Science and Software Engineering. 2012; 02(12).
12. Lovins JB. Development of a stemming algorithm. Mechanical Translation and Computational Linguistics. 1968;11:22-31.
13. Hull D. Stemming algorithms: A case study for detailed evaluation. JASIS. 1996;47 (01):70-84.
14. Kasthuri M, Kumar BR. An improved rule based iterative affix stripping stemmer for tamil language using K-mean clustering. International Journal of Computer Applications. 2014;94(13):36-41.
15. Knuth and Donald. The art of computer programming, (2nd Ed.) Sorting and Searching, Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc. 1998;3.
16. Askitis N. Fast and compact hash tables for integer keys. Thirty-Second Australasian Computer Science Conference (ACSC 2009), Wellington, New Zealand; 2009.
17. Liu X, He J, Lang B. Reciprocal hash tables for nearest neighbor search. Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, Bellevue, Washington; 2013.
18. Jurafsky D, Martin J. Language modeling with NGRAMS. [Online]. Available:https://web.stanford.edu/~jurafsky/slp3/3.pdf