RUHUNA JOURNAL OF SCIENCE
Vol 9(2): 150-159, December 2018
eISSN: 2536-8400
DOI: http://doi.org/10.4038/rjs.v9i2.43

Short paper

Detecting and correcting real-word errors in Tamil sentences

R. Sakuntharaj[1]* and S. Mahesan[2]

[1]*Centre for Information and Communication Technology, Eastern University, Sri Lanka*
[2]*Department of Computer Science, Faculty of Science, University of Jaffna, Sri Lanka*

*Correspondence: *sakuntharaj@esn.ac.lk; ORCID: 0000-0002-5689-6470

Received: 15th March 2018, Revised: 9th November 2018, Accepted: 19th November 2018

Abstract. Spell checkers concern two types of errors namely non-word errors and real-word errors. Non-word errors fall into two sub-categories: First one is that the word itself is invalid; the other is that the word is valid but not present in a valid lexicon. Real-word error means that the word is valid but inappropriate in the context of the sentence. An approach to correcting real-word errors in Tamil language is proposed in this paper. A bigram probabilistic model is constructed to determine appropriateness of the valid word in the context of the sentence using a 3GB volume of corpora of Tamil text. In case of lacking appropriateness, the word is marked as a real-word error and minimum edit distance technique is used to find lexically similar words, and the appropriateness of such words is measured by a word-level n-gram language probabilistic model. A hash table with word-length as the key is used to speed up the search for words to check for the lexical similarity. Words of length differing less than two with the length of the 'inappropriate' word are considered to search in the hash table. Test results show that the suggestions generated by the system are with 98% accuracy as approved by a Scholar in Tamil language.

Keywords. Tamil, real-word error, bigram, minimum edit distance, error correction.

1 Introduction

Tamil language possesses a rich history of more than 2000 years and spoken primarily in Sri Lanka, India, Malaysia and Singapore. Tamil, a Dravidian language, an official language of Sri Lanka, is an agglutinative language having rich morphological structure. Tamil words are formed by lexical roots

followed by one or more suffixes (Navalar 1998, Sangar 2006, Nuhman 2013).

Spell checker is a tool that finds and corrects misspelt words in a text document. Spelling error detection and correction techniques are widely used by text editing systems, search engines, machine translation systems, speech recognition systems, text to speech and speech to text conversion systems, speech recognition systems and optical character recognition systems.

Misspelt words can be classified into two categories of errors namely non-word errors and real-word errors (Kukich 1992, Samanta and Chaudhuri 2013). Non-word errors fall into two sub-categories: First one is that the word itself is invalid (e.g. மளை) and the other is that the word is valid (e.g. அளகு) but not present in a valid lexicon. Real-word error means that the word is valid but inappropriate in the context of the sentence. For example, the sentence மலை நாட்டில் மழை பெய்யும் becomes non-sensical when the two words மலை and மழை are swapped.

In this paper, we focus on detecting and correcting real-word errors in Tamil sentences. We propose a method that uses *bigram probabilistic model* to detect the real-word errors and *minimum edit distance technique* to generate the suggestions for them while making use of a *hash table* to speed up the lookup.

2 Methodology

The input word is first checked with lexicon using a *tree-based lookup algorithm* to see whether the word is valid or invalid. Letter-level and word-level bigrams & trigrams techniques are used to generate suggestions for the invalid words with two different types of hash tables to speed up the search. After correcting all the invalid words in the sentences, real-word errors are detected and corrected. As highlighted in Figure 1, real-word errors are detected by considering the appropriateness of the words in the context of the sentence. The appropriateness of the words is determined by a *bigram probabilistic model* constructed using a pooled corpus of Tamil text (as described in Section 2.1). In case of lacking appropriateness, the word is marked as a *real-word error*, and then *minimum edit distance* (MED) technique is used to find lexically similar words. These words are taken as candidates for suggestions to the word of concern that is marked as a real-word error. The appropriateness of these candidates is measured by a *word-level n-gram language probabilistic model* (described in Section 2.2). Finally, the suggestions are refined by considering the probabilities of bigrams formed by each of these candidates with the word that follows the word of concern in the given sentence. It is interesting to note that the real-word errors are found

to be lexically similar to the word that was intended in the context. It is rather rare or unnatural in Tamil to see real-word errors that are not lexically similar.

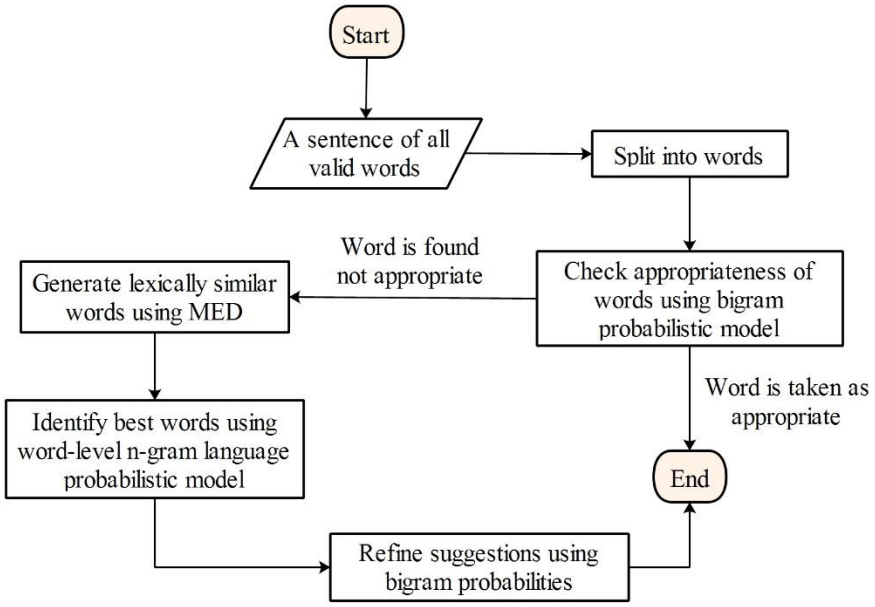

Fig. 1. Flowchart of the real-word error detection and correction steps

2.1 Bigram probabilistic model to determine the appropriateness of words

The bigram probabilistic model constructed to determine the appropriateness of a word in the context of the sentence is described below. Let the sequence of words $(w_1, w_2, \ldots\ldots, w_n)$ denote the sentence in the corpus. Several Tamil corpora collected from various sources including Tamil news websites, Tamil articles, Tamil story books *etc.* were incrementally pooled into a single corpus. While they were collected, validity of words was checked by our system and texts were added to the corpus after correcting the invalid words. Also *new valid* words were added to the lexicon.

Let $\big((\$start, w_1),\ (w_1, w_2),\ (w_2, w_3),\ (w_3, w_4), \ldots\ldots\ldots, (w_{n-1}, w_n),\ (w_n, \$end)\big)$ denote the word-level bigrams of the corpus.

Let pm be the bigram probability matrix that is constructed for the corpus using Equation 1.

$$pm_{ij} = p_r(w_j|w_i) = \frac{count\ of\ bigram(w_i, w_j) + 1}{count\ of\ w_i + n} for\ i, j = 1,2,..n \quad (1)$$

where n = $|V|$ is the size of the vocabulary set V.

The size of the vocabulary set is 1,778,676. To calculate the probability pm_{ij}, the *Laplace smoothing technique* (Jurafsky and Martin 2018) is used to get rid of zero probabilities. As the result of this smoothing, the probability $\frac{C}{N_w}$ of a $P_r(w_i \mid w_{i-1})$ becomes $\frac{C+1}{N_w+n}$. That is, the zero probability now becomes $\frac{1}{N_w+n}$ after smoothing.

To determine the appropriateness of word w_i, the probabilities $P_r(w_i \mid w_{i-1})$ and $P_r(w_{i+1} \mid w_i)$ are considered. When either one of them is found to be too small (that is less than a prescribed threshold), w_i is considered inappropriate in the context of the sentence. In the experiments it found that words with bigram probabilities less than 10^{-6} are not suitable in the context of the sentences, as determined by the Scholar in Tamil language, and thus the value 10^{-6} is chosen to be a good threshold for word to be considered as appropriate or inappropriate.

Let us consider an example to see how to detect a real-word error in a sentence: 'அவன் பலங்களைச் சாப்பிட்டான்'. The bigram probabilistic model built for the corpus gives the bigram probabilities for this sentence as follows:

P_r (அவன் | \$start) = $4.3623*10^{-4}$

P_r (பலங்களைச் | அவன்) = $5.3920*10^{-7}$

P_r (சாப்பிட்டான் | பலங்களைச்) = $5.6221*10^{-7}$

P_r (\$end| சாப்பிட்டான்) = $8.7481*10^{-5}$

Of the above values, P_r (பலங்களைச் | அவன்) and P_r (சாப்பிட்டான் | பலங்களைச்) are found to be less than the threshold value of 10^{-6}, indicating the word 'பலங்களைச்' is inappropriate in this sentence making a real-word error.

How to find words lexically similar to the word under consideration, and how to choose the most appropriate one out the lexically similar ones are described below.

2.2 Finding lexically similar appropriate words

As the first step to generate suggestion for alternative words for inappropriate word w_i of length m_i, lexically similar words are found using Minimum Edit Distance (MED) technique. MED is the minimum number of editing operation required to transform one string into another (Damerau 1964, Wagner and Fisher 1974). A lexicon word that gives minimum edit distance with erroneous word less than four is considered as a lexically similar suggestion for the erroneous word. For the erroneous word of length m_i, words of lengths from m_i-1 to m_i+1 are checked for lexical similarity. This range was determined by analysing the words with wider range of length.

After generating the lexically similar words for real-word errors, the appropriateness of the generated words is measured by a *word-level n-gram language probabilistic model*. The model helps choosing suggestions for erroneous words from the constructed corpus. The model is constructed for the sentences in the corpus as follows:

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

Let p_i denote a measure for a word w_i calculated by using Equation 2 as instructed in (Jurafsky and Martin 2018).

$$p_i = \lambda_1 p_i^{(1)} + \lambda_2 p_i^{(2)} + \lambda_3 p_i^{(3)} \tag{2}$$

where λ_js are positive scalars such that $\lambda_1 + \lambda_2 + \lambda_3 = 1$,

$p_i^{(1)}$ denotes the probability $p_r(w_i)$ in the corpus,

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

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

The values of λ_js are tuned by 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 listed below:

$$p_i^{(1)} = p_r(w_i) = \frac{count\ of\ w_i}{Total\ no.of\ words\ in\ the\ corpus} \tag{3}$$

$$p_i^{(2)} = p_r(w_i \mid w_{i-1}) = \frac{count\ of\ bigrams\ (w_{i-1}, w_i)}{count\ of\ w_{i-1}} \tag{4}$$

$$p_i^{(3)} = p_r(w_i \mid w_{i-2}, w_{i-1}) = \frac{count\ of\ trigrams\ (w_{i-2}, w_{i-1}, w_i)}{count\ of\ (w_{i-2}, w_{i-1})} \tag{5}$$

After calculating the measures for all the suggestion words for an 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 under consideration. 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 words is found to be less than five in many cases.

Let $\{\hat{w}_i^j\}_{j=1...k}$, be a set of top k candidates for suggestions for inappropriate word w_i. Let w_{i+1} be the word that comes next to w_i in the sentence under consideration and the candidates $\hat{w}_i^j$ that give the probability $\hat{p}_j = p_r(w_{i+1}|\hat{w}_i^j)$ greater than 10^{-6} are taken as refined suggestions.

Let's go back to our example sentence: 'அவன் பலங்களைச் சாப்பிட்டான்'.

Once the word 'பலங்களைச்' is found to be inappropriate as already described in Section 2.1, the real-word error correction module generates lexically similar words for the word 'பலங்களைச்' using MED: The module finds the words {'பழங்களை', 'பழங்களைச்', 'பழங்களினைச்', 'பாலங்களைக்', 'வைரங்களை', 'கோலங்களைக்', 'சடலங்களை'} from the lexicon as lexically similar to 'பலங்களைச்'. These words are considered as candidates for suggestion.

The word-level n-gram language probabilistic model ranks these candidates based on their measures of p_is that are calculated using Equation 2 as described above. For the example considered above, p measures for the suggestion candidates are:

$p_1 = p$ (பழங்களைச்) = $2.7173*10^{-4}$
$p_2 = p$ (பழங்களினைச்) =$1.7502*10^{-5}$
$p_3 = p$ (பழங்களை) = $8.0030*10^{-6}$
$p_4 = p$ (பாலங்களைக்) = $8.3220*10^{-6}$
$p_5 = p$ (வைரங்களை) = $9.4058*10^{-6}$
$p_6 = p$ (கோலங்களைப்) = $2.8425*10^{-8}$
$p_7 = p$ (சடலங்களை) = $7.3924*10^{-9}$

After determining the measures of these lexically similar words, up to five words with measures more than a predetermined threshold 10^{-6} would be selected. For the above example, the words 'பழங்களைச்', 'பழங்களினைச்', 'பழங்களை, 'பாலங்களைக்' and 'வைரங்களை' would be selected as their measures are higher than the threshold of 10^{-6}.

For these selected five words $\hat{p}_j = P_r(w_{i+1}|\hat{w}_i^j)_{j=1...5}$ are obtained from the bigram probability matrix as follows:

$\hat{p}_1 = P_r$(சாப்பிட்டான் | பழங்களைச்) = $2.3482 * 10^{-3}$
$\hat{p}_2 = P_r$(சாப்பிட்டான் | பழங்களினைச்) = $2.2393 * 10^{-3}$
$\hat{p}_3 = P_r$(சாப்பிட்டான் | பழங்களை) = $5.6187 * 10^{-7}$

$\hat{p}_4 = P_r$(சாப்பிட்டான் | பாலங்களைக்) = 5.5821 * 10^{-9}
$\hat{p}_5 = P_r$(சாப்பிட்டான் | வைரங்களை) = 5.6219 * 10^{-9}

Of these values, P_r (சாப்பிட்டான் | பழங்களைச்) and P_r (சாப்பிட்டான் | பழங்களினைச்) are greater than the threshold value of 10^{-6} and thus the words {'பழங்களைச்', 'பழங்களினைச்'} are given as refined suggestions for the real-word error 'பலங்களைச்'.
Time taken for generating lexically similar words for w_i is dependent on the lexicon size and lookup method. Fortunately, using a hash table is a good approach to speed up the lookup as an alternative to mere linear search. In order to select the words of given length, the hash table with word-length as the key is built for the lexicon words and stored in a Python pickle for easy and quick restoration, which significantly saves time.

3 Results and Discussion

The programming language Python 3.5 has been used to write the programs and run on Ubuntu 16.04 on a Dell machine with Intel Core i7-6500U processor of 2.5GHz clock speed and with 8GB RAM.

The proposed real-word error detection and correction system corrects following types of real-word errors that occur due to the wrong choice of letters from the confusion sets and due to '*sandhi*' mistakes. Sandhi inflects two words coming in between them. The following groups of letters form confusion sets:

{ழ, ள, ல}: In Tamil, they sound somewhat similar to 'la',

{ந, ண, ன}: these letters sound somewhat similar to 'na' and

{ர, ற}: these letters sound somewhat similar to 'ra'.

Each of these letters in each of these sets has distinct pronunciation, and has to be used appropriately with care.

The error due to wrong choice of letters from any confusion set more likely to occur when a transliteration method is used to type Tamil text or in speech to text conversion. For example, to input மழை one may type 'malai', 'maLai', 'malzi' (as all sounds similar) instead of right choice for letter 'ழை' in the transliteration system. The sandhi mistake occurs due to the wrong suffix letter or missing of it. The right suffix depends on the word that follows it. It should be noted that the suggestions to all these errors are found to be lexically similar to the erroneous words.

During the error detection and correction process, the inappropriate words in the sentences and suggestion words for them are recorded. The following five examples show the output of the real-word error detection and correction.
Example 1: (Correcting contextual spelling mistakes that occur due to the

letters in the confusion set {ழ, ள, ல})
Input sentence: அவன் பலங்களைச் சாப்பிட்டான்.
Output:
Real-word errors: பலங்களைச்
Suggestions: பழங்களைச் / பழங்களினைச்

Example 2: (Correcting contextual spelling mistakes that occur due to the letters in the confusion set {ர, ற})
Input sentence: அவள் கரை படிந்த சட்டை அணிந்திருந்தாள்.
Output:
Real-word errors: கரை or படிந்த
Suggestions: கறை படிந்த or கரை மடிந்த

Example 3: (Correcting contextual spelling mistakes that occur due to the letters in the confusion set {ந, ண, ன})
Input sentence: நாண் நேற்று கோயிலுக்குப் போனேன்.
Output:
Real-word errors: நாண்
Suggestions: நான்

Example 4: (Correcting *sandhi* (junctional inflection) mistakes between two words)
Input sentence: நான் அவனைச் கண்டேன்.
Output:
Real-word errors: அவனைச்
Suggestions: அவனைக் / அவளைக் / அவரைக்

Example 5: (Correcting any real-word error in a sentence)
Input sentence: அந்த விதியின் நடுவே குழியொன்று உள்ளது.
Output:
Real-word errors: விதியின்
Suggestions: வீதியின் / விடுதியின் / நதியின்

In Examples 1, 2 & 3, the real-word error detection and correction system detects and corrects the contextual spelling mistakes. In Example 4, there is a 'sandhi' mistake between the words 'அவனைச்' and 'கண்டேன்'. However, the proposed system identifies the word 'அவனைச்' as a real-word error and suggests the word 'அவனைக்' as the most appropriate alternative to 'அவனைச்' that results in correcting the sandhi mistake. In Example 5, the real-word error detection and correction system identifies the word

‘விதியின்’ as a real-word error and provides the words {‘வீதியின்’, ‘விடுதியின்’, ‘நதியின்’} as suggestions.

This experiment is tested with two sets of sentences:

- Set 1 consists of one thousand sentences of random choice (none being taken from the corpus already built). These 1000 input test sentences have 131 real-word errors as confirmed by a Scholar in Tamil language, and the bigram probabilistic model detects all of them.
- Set 2 consists of 100 sentences deliberately made different from those in Set 1 so that each has a real-word error, the bigram probabilistic model detects all of them.

The following table shows the details of

(A) the number of real-word errors existed in the chosen sets of sentences,

(B) the number of real-word errors detected by the system,

(C) the number of real-word errors with at least one suggestion with the bigram probabilistic measure greater than or equal to 10^{-6} as described in Section 2.2,

(D) the number of generated suggestions with the bigram probabilistic measure greater than or equal to 10^{-6} as described in Section 2.2, and

(E) the number of generated suggestions approved by Tamil Scholar.

Test set	(A)	(B)	(C)	(D)	(E)
Set 1	131	131	99*	130	127 (97.7%)
Set 2	100	100	90*	156	154 (98.7%)

* the reason for not getting any suggestion for 32 + 10 words is that the bigram involving the word in the input sentence that follows the suggestion candidate does not exist in the model constructed for the corpus.

The test results show that 98% of the suggestions generated by the system are found to be suitable as approved by a Scholar in Tamil language.

4 Conclusion

In this work, a method has been proposed to detect and correct real-word errors in Tamil sentences. In this regard, a bigram probabilistic model is constructed to detect real-word errors. Minimum edit distance technique is used to generate suggestions. Test results show that the bigram probabilistic

model detects all the real-word errors in input sentences and suggestions generated by the system are with 98% accuracy as approved by a Scholar in Tamil language.

Acknowledgements

Two anonymous reviewers are acknowledged for their comments on the initial manuscript.

References

Damerau F. 1964. A technique for computer detection and correction of spelling errors. *Communications of the Association for Computing Machinery* 7(3): 171–176, doi:10.1145/363958.363994

Jurafsky D, Martin JH. 2018. *Language Modeling with N-grams.* [Online] Available at: https://web.stanford.edu/~jurafsky/slp3/3.pdf

Kukich K. 1992. Techniques for Automatically Correcting Words in Text. *Association for Computing Machinery Computing Survey* 24(4): 377– 439, doi: 10.1145/146370.146380

Navalar A. 1998. *Tamil Grammar Questions and Answers.* No. 366, Kankesanthurai Road, Jaffna: Vannai Santhayarmadam, Jaffna.

Nuhman MA. 2013. *Basic Tamil Grammar.* University of Peradeniya: Readers Association, Kalmunai.

Samanta P, Chaudhuri B. 2013. A simple real-word error detection and correction using local word bigram and trigram. *Proceeding of the 25th Conference on Computational Linguistics and Speech Processing*. Kaohsiung, Taiwan: 211–220.

Sangar V. 2006. *Tamil Grammar.* Puduchcheri, India: Nanmozi Printers.

Wagner RA, Fischer MJ. 1974. The String to String Correction Problem. *Journal of the Association for Computing Machinery* 21(1): 168–173, doi:10.1145/321796.321811