International Journal of Engineering Sciences & Emerging Technologies, June 2012.
ISSN: 2231 – 6604 Volume 2, Issue 2, pp: 43-52 ©IJESET

A NON-WORD KANNADA SPELL CHECKER USING MORPHOLOGICAL ANALYZER AND DICTIONARY LOOKUP METHOD

Rajashekara Murthy S[1], Vadiraj Madi[2] , Sachin D[3], Ramakanth Kumar P[4]

[1]Assistant Professor & [2,3] Students, Department of CSE,
[4]Professor and Head, Department of ISE,
RV College of Engineering, Bangalore -560059, INDIA
rajashekaramurthys@rvce.edu.in , sachindevarajappa@gmail.com
vadirajrmadi@gmail.com , ramakanthkp@rvce.edu.in

ABSTRACT

A non-word spell checker for Kannada language is designed and implemented. The spell checker developed, by using morphological analyzer and dictionary lookup method, identifies errors in a given block of text and suggests a list of valid words for each of the erroneous word. The suggested list contains words from the dictionary if they differ from the erroneous word by one edit distance length. It is a multi-threaded tool that employs efficient data structures and allows the user to add new words if they are not available in the dictionary. It has a better Graphical User Interface (GUI) and performs better when compared to the commercially available Spell Checkers.

KEYWORDS: *Spell Checker, Kannada, Trie, Non-word Error, Morphological Analyzer*

I. INTRODUCTION

Spell checker is one of the software tools required by machines to process natural languages effectively. It can be used as an independent tool or as a tightly coupled component of a number of systems like Machine Translator, OCR, and Word Processor. Whatever may be the case; a spell checker should handle a word and identify the spelling error in it and should help in correcting them.

Kannada is one of the forty most widely spoken languages in the world and has roughly over 38 million speakers [1]. It is one of the four major literary languages of the Dravidian family. Kannada is one of the oldest languages of the world and the earliest written document in Kannada can be seen in the Halmidi inscription which is dated back to 450 A.D. [2]. Despite of its popularity and its status as a classical language, Kannada language processing is still in its infancy.

A study by Damerau reports that 80% of the misspelled words in English are non-word errors and caused by single error misspellings [3]. We did a simple study similar to B. B. Choudhury [4] before the implementation of this tool. Around 1,12,000 words are obtained from written answer-scripts and dictated notes taken from secondary and higher secondary school students. When analyzed, it was found that 2,431 words were misspelled. Out of this 2004 words were in the category of non-word errors. Though a comprehensive study is required to come to a clear opinion, it was enough to realize that non-word error detection is the first step towards a truly professional spellchecker.

The paper is organized in to the following sections. Section 2 describes the relevance, types of errors and the data structures used. Section 3 discusses the challenges in building a spell checker for Kannada and the work done so far. Design of a dictionary, word stripping methods and process of

error identification is given in section 4. Implementation and results are discussed in Section 5. Finally the paper ends with some concluding remarks and acknowledgements.

II. PRELIMINARIES

A spell checker is a tool that flags words in a document that are misspelled or not in dictionary. A Spell checker consists of two main components - error detection and suggestions prediction. Error detection component identifies the errors. Upon detecting the error, the suggestions prediction component provides suggestions that are closer to the misspelled word by a defined metric. A spell checker should be capable of operating on a block of text or on a single word.

A spell checker can be used as a standalone component or as part of a language processing tool. A better performance can be achieved of a machine translator, if it is supported by a powerful spell checker. The translator provided by Google, translates the Kannada sentence "ನನಗೆ ಮಾವಿನ ಹಣ್ಣು ಇಷ್ಟ" correctly as "I like Mango fruit". But, "ನನಗೆ ಮಾವಿ ಹಣ್ಣು ಇಷ್ಟ" is wrongly translated as "I do not like the fruit of Maui" as shown in the Figure 1.

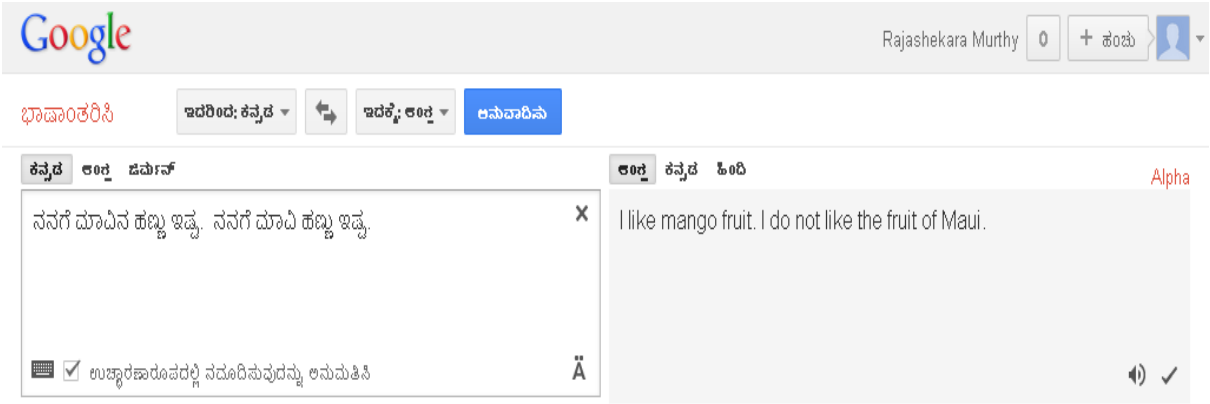

Figure 1: Snapshot from Google Translator

Instead of detecting and reporting "ಮಾವಿ" as a misspelled word, the translator gives wrong translation. Had there been a spell checker, that detects the spelling error and allow the user to correct it, the translator would have translated the sentence properly. A spell checker facilitates improvement in performance of a machine translator. As of now a few Optical Character Recognition (OCR) software are available for digitizing the printed Kannada documents. However the accuracy of these OCRs is unacceptable. One of the popular OCR developed for Kannada provides a maximum accuracy of 91% at character level [5]. In a large text document these errors add up and become unacceptable. So it is necessary to use a spell checker at some intermediate stage to improve the performance of an OCR. Apart from these, a spell checker is a vital tool for word processor. It eases the task of creating error-free Kannada documents.

Errors can be categorized into non-word error and real-word error. Non-word error deals with misspelled words that are formed by common typographical errors. An instance of a non-word error is when "ರಾಮ" is typed as "ರಿಮ". A real-word error occurs when a word is misspelled as another valid word, which is not proper for the context. An instance of a real-word error is when the sentence "ರಾಮನು ಮನೆ ಮಾರಿದನು" is typed as "ರಾಮನು ಮನ ಮಾರಿದನು". Here the valid Kannada word "ಮನೆ " is misspelled as another valid word " ಮನ ". Though the sentence is syntactically correct, it is semantically wrong. Detecting such errors require knowledge of the context. Some authors hence use the term 'context-sensitive word error' for real-word errors. The above categories of errors occur due to one or more of the following reasons.

1. Addition of an extra character results in an Insertion error. For Example, consider the misspelled word ಅಣ್ಣು that results when the glyph 'ು' is inserted to the valid word ಅಣ್ಣ.
2. Absence of a character results in a deletion error as in the misspelled word ರಾಣ, where the character 'ವ' is missing from the valid word ರಾವಣ.
3. Replacement of a character results in a Substitution error as in the case where the valid word ಅಣ್ಣ is misspelled as ಮಣ್ಣ by replacing 'ಅ' by 'ಮ'.
4. A Transposition error occurs due to exchange of characters. For example, consider the valid word ಮರ which is misspelled as ರಮ where 'ರ' and 'ಮ' are interchanged.
5. A split word error occurs when the needless space or gap inserted in between the characters of a word. For example, consider the word "ಅಪಘಾತ", where an unnecessary space between the letters 'ಪ' and 'ಘಾ' results in invalid words "ಅಪ" and "ಘಾತ".
6. An absence of space between two or more consecutive words results in run-on error. For Example, the invalid word "ಅವನುಮತ್ತು" is resulted due to missing of space, the delimiter, between the words "ಅವನು" and "ಮತ್ತು".

As per our study and one of the first general findings [6] 80% of all misspelled non-word errors were due to the first four reasons mentioned above. Hence the tool developed focuses on handling non-word errors resulted by these four reasons. In order to build a non-word boo-boo spell checker for Kannada, one would require a powerful dictionary for reference in the word error detection and suggestions prediction phases. Creating a dictionary having all the words of the language is a laborious task considering the large variation of affix combinations in Kannada. To handle this problem, we use a morphological analyzer and a dictionary of only the root words. For example, only the root word ಕಣ್ಣು is stored in the dictionary instead of storing all its inflections ಕಣ್ಣನ್ನು, ಕಣ್ಣಿನಿಂದ, ಕಣ್ಣಿಗೆ etc. Further the dictionary is divided into twenty seven sub-dictionaries each having related words and mapping on to a particular paradigm table. The division or grouping is done based on a set of rules elaborated in section 4. A paradigm table is a set of suffixes that creates inflections for all the words of a sub-dictionary of root words. There are twenty seven paradigm tables having one-to-one correspondence with sub-dictionaries of the roots. Initially the input word is searched in the root trie, if the input word does not exist in the dictionary as a root word, then the algorithm will employ a morphological analyzer to strip all the suffixes and get the root word. The root word is searched in the selected dictionaries once again. If the search is unsuccessful then the error is reported. To improve the searching efficiency trie is used.

The application makes use of trie data structure to build the dictionary and Levenshtein edit distance algorithm to find suggestions for misspelled word.

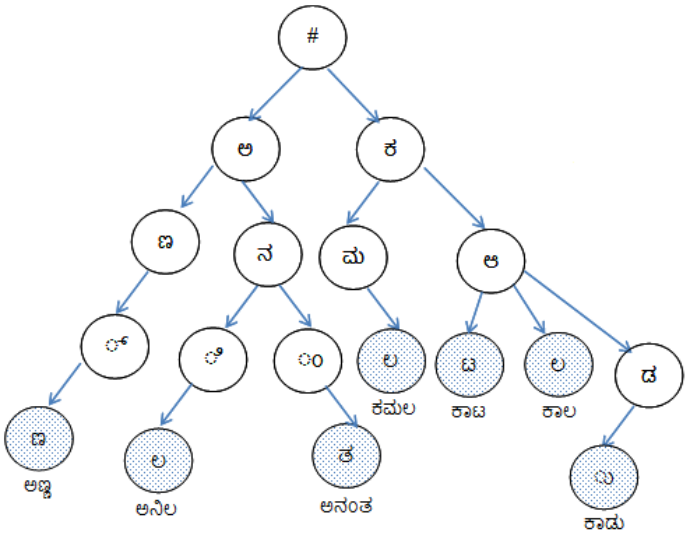

Figure 2. Example of a trie

A trie [7] is a data structure that allows words having common prefix stored once. The tree branches out where the common prefix ends. Figure 2 depicts the trie data structure built for seven root words- ಅಣ್ಣ, ಅನಿಲ, ಅನಂತ, ಕಮಲ, ಕಾಟ, ಕಾಲ, ಕಾಡು. Here every node contains a Unicode character corresponding to Kannada script. Trie enhances the searching speed [8] as it can search a word of length n in linear time, O(n)[9].

Levenshtein Distance[10], LD is used as a metric to measure similarity between words. The LD between two words is defined as the minimum number of edit operations needed to transform one word into the other, with the allowable edit operations being insertion, deletion, or substitution of a single character [11]. Let d(x,y) define the LD between the words x and y. Then d(ಕಣ್ಣು,ಕಣ್ಣ) = 1. This is because ಕಣ್ಣು is composed of five Unicode characters "\u0c95 + \u0ca3 + \u0ccd + \u0ca3 + \u0cc1" and ಕಣ್ಣ is composed of four Unicode characters "\u0c95 + \u0ca3 + \u0ccd + \u0ca3". The Unicode for 'ು' \u0cc1 is missing from ಕಣ್ಣ (one deletion operation). Similarly, d(ಕಣ್ಣೀರು, ಕಣ್ಣೀರ) = 1. Whereas d(ಕಣ್ಣೀರು,ಕಣ್ಣ) = 3 (three deletion operations), d(ಕಣ್ಣು, ಕಣ್ಣೀರ) = 2 (One replacement and one insertion operation). If the misspelled word is ಕಣ್ಣh then ಕಣ್ಣು is suggested, but not ಕಣ್ಣೀರು. Similarly, if the miss spelled word is ಕಣ್ಣೀರ then ಕಣ್ಣೀರು is suggested, but not ಕಣ್ಣು. ThasuThat is word 'x' is suggested for word 'y', if and only if d(x,y) = 1.

III. CHALLENGES AND RELATED WORK

As Kannada is an agglutinative and morphologically rich language, each root word can combine with multiple morphemes to generate huge number of word forms. For the purpose of supporting such inflectionally rich languages, the structure of each word has to be identified.
Kannada has compound, derived and simple nouns. It also has first person, second person, and derived pronouns. Nouns get inflected for number. Gender, number, tense, voice, aspect and mood cause inflections to verbs. Many times it is context which decides whether a word is a noun or adjective or adverb or post position. This increases the complexity of parsing Kannada language. Because of all these reasons development of a spell checker for Kannada language is a challenging task.
A complete spell checker for Kannada language has not been developed yet. There are a few spell checkers, they are very trivial, non-professional and far from complete. Kannada Kasturi has developed a basic version of Kannada spell checker, but the performance of this tool is very poor. This tool is developed based on the transliteration approach. However with this approach the users must be aware of the transliteration mappings [12] to provide the input to the tool.

IV. DESIGN

The tool handles all the text in Unicode format. The Unicode for the Kannada script corresponds from \u0C80 to \u0CFF [13]. User can input a block of text either manually or through a file. The block of text is then split into individual words and then each of these words is analyzed using a morphological analyzer, which makes use of separate root and suffix dictionaries. A one to one correspondence has been established between different categories of root and suffixes with the help of a mapping function. Morphological analysis helps in detecting whether word is spelled correctly or not. If the word is found to be incorrect then further process is carried out to identify the type of error viz. correct root and incorrect suffix, incorrect root and correct suffix, correct root and matching suffix. Each of these errors is handled separately. Further the erroneous words are displayed to the user with appropriate suggestions. The misspelled words are then updated as per the suggestions selected by the user. For the words that are not present in the dictionary, users are provided with an option to add the word to dictionary through a set of questions that is required to identify the appropriate sub-dictionary. The architecture of the system is as shown in the figure 3.

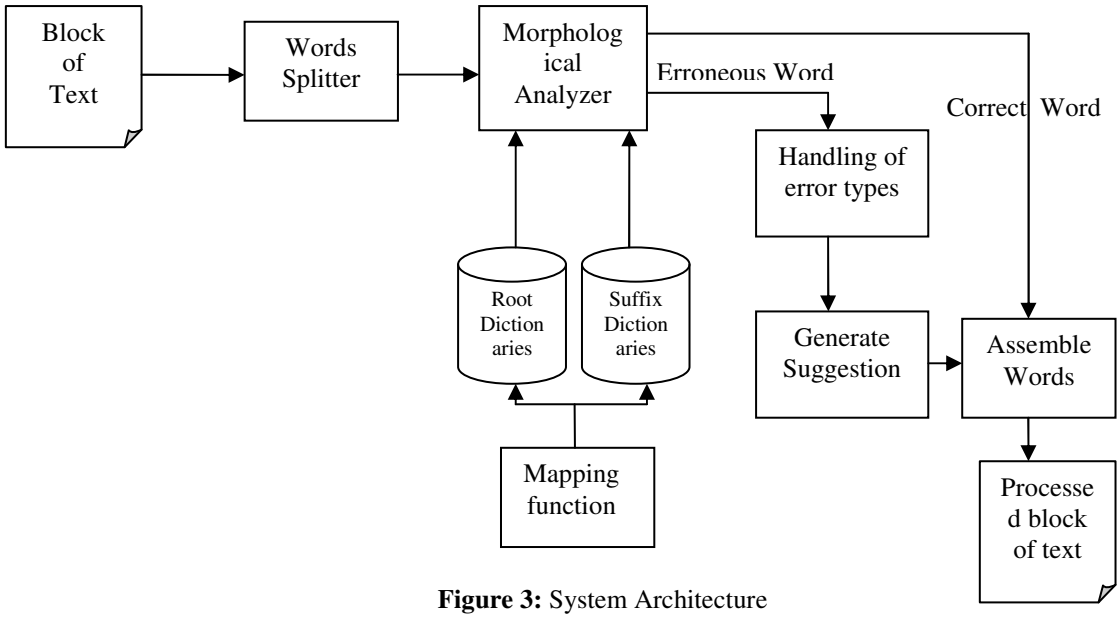

Figure 3: System Architecture

4.1 Dictionary

All the root words of the language can be placed in a single dictionary. This results in a huge dictionary of millions of words. Luckily, there are some inflection patterns observed [14]. For example, it can be seen that most of the words ending with ಅ and whose gender is neutral, are inflected with suffixes – ಉ, ವನ್ನು, ಇಂದ, etc. Similarly, those ending with ಅ and whose gender is masculine, are inflected with the suffixes – ನು, ಅನ್ನು, ನಿಂದ etc. Based on such observations a set of rules are framed and the words are divided into twenty seven mutually disjoint sub-dictionaries and the suffixes are grouped into twenty seven overlapping paradigm tables. This may not hold good for all the words. Hence necessary care has been taken to ensure that these exceptional cases are handled properly. For example, though the words ಕಾಡು, and ಪಶು are neutral gender and both end with ಉ they are inflected by different set of suffixes. That is they are mapped on to different paradigm tables [15].

These twenty seven sub-dictionaries can cover most of the words taken from secondary and higher secondary text books. However there may be exceptional cases for which separate set of rules can be written and tables can be updated accordingly. By following this approach space required to store all forms of words can be substantially reduced.

4.2. Suffix Stripping

The tool makes use of a simple suffix stripping algorithm. This algorithm makes use of grammar based rules to strip a word into its root word and suffix. However exact suffix stripping is possible only for a correctly spelled word. In case of misspelled word, as there is an ambiguity as to whether the error exists in the root or suffix, only probable suffix stripping can be done. In this process, each individual word is scanned from right to left in the suffix trie. Upon finding a valid suffix, it is stripped from the word. The resulting word is then searched in the root trie. If it is found then the word is spelled correctly else the same step is repeated for all the valid suffixes in each individual word. All such pairs of valid suffix and their corresponding stripped part are stored in a buffer for further processing.

4.3 Error Detection

The pairs obtained after suffix stripping is processed in this stage to classify into one of the following cases.

a) Correct root word + Matching suffix

On identifying the valid suffix, the root is searched in the root dictionaries. In this case since root word is found, no further processing is needed and hence it is considered as a valid word.

Example: ರಾಮನ =>ರಾಮ+ ನ

(root) + (suffix)

b) Invalid root+ valid suffix

In this case since the root is invalid, so the exact suffix can't be determined. Hence we get a list of possible pairs of stripped word and valid suffix. The valid suffix gives the specific category of the possible root word. Further in this category the stripped part is checked for a word match. Then the words obtained are combined with the suffix to form a suggestion. The same process is repeated for all possible pairs to generate a list of suggestions. For example, consider the word 'ರಮನಿಂದ'

Possible pairs	Suggestions
1) ರಮನಿಂ + ದ (root) + (suffix)	1) ರಾಮನಿಂದ
2) ರಮ + ನಿಂದ (root) + (suffix)	2) ಯಮನಿಂದ 3) ರಮಣನಿಂದ

c) Root noun/verb+ invalid suffix

On finding that the suffix is invalid, the word is scanned from the beginning i.e. from left to right in the root trie. Upon finding a valid root, the given word is stripped into a valid root and a possible suffix. This process of stripping provides a list of pairs. For each pair, the suffix is searched in the paradigm table corresponding to the root. If no suffix match is returned then the pair is considered to be invalid and hence it is dropped. However, if a suffix match is found then the obtained suffix is combined with the root word to form a suggestion. For example, consider the word 'ಸೀತೆಯನ್ನ'

Possible pairs	Suggestions:
1) ಸೀತ + ಯನ್ನ (root) + (suffix)	1) ಸೀತೆಯನ್ನು
2) ಸೀತೆ + ಯನ್ನ (root) + (suffix)	

4.4. Providing Suggestions

Once the process of error detection is completed, the next step is to provide suggestions for the detected error. Automatic error correction is not advisable as it may lead to further errors. A demonstration of this can be made in the word processor Microsoft Word 2007. When a user types the word 'eyt' the word processor auto-corrects it as 'yet', even though the user intended 'eye'. For this reason in the tool the auto-correct feature is provided as an option as user intervention would provide better performance in spell checking. Hence upon detecting the error, the user will be provided with a list of probable correct words which the user can select to update the misspelled word. It may also be possible that the correct word expected by the user may not be listed in the suggestions hence the tool also provides an option for the user to type in the correct word and update it in the place of misspelled word. Ranking of suggestions is not required since only one edit distance length words are provided as suggestions.

4.5 Adding words to dictionary.

It is impossible to build a dictionary that can cover all the root words of a language. Hence it is necessary to update the dictionary and to add more and more words as and when they are encountered. To make this possible the tool also provides an option to add new words to the dictionary. As the dictionary is divided into different paradigms which are further categorized based on the predefined rules, it is necessary to determine the category of the word before adding it to the dictionary. To accomplish this task the tool follows a questionnaire based approach wherein it provides the user with a set of questions which are framed based on the set of the rules that define a particular category. The questions used are built in such a way that they can be easily understood by the user and are asked to select one of the available options rather than to type in any input. Based on the answers given by the user, probable categories of root word along with their suffixes are displayed and user selects the particular category to which the word is added. Here it may also be possible that either by negligence or by lack of knowledge the user may answer the questions incorrectly thereby forcing the tool to place the word in a wrong category. Hence to avoid such errors whenever a user adds a new word to the dictionary, it is also written to a temporary file along with category selected for that word. Further, periodically the administrator can review the temporary file to ensure the correctness and to place the word in the correct category. If the user tries to add a word that is already present in the dictionary an alert message is displayed and the user is stopped from doing so.

Example: consider the process of adding the word "ಸೋಮು".

Here the tool provides a set of questions like gender to be selected, type of paradigm, singular or plural, the ending letter and so on. Based on the answers to these questions the tool displays probable categories of root words along with their suffixes. The user can select a particular category and add the word to that category.

V. RESULTS AND DISCUSSION

The tool is implemented in JAVA programming language. This makes the tool platform independent. It makes use of the light-weight swing components to build the user-interface and provides a GUI-based window with menus and buttons for the user interaction. Suitable GUI components are used to display the various features of the tool. The user is provided with a text area to input the block of text which has to be corrected. Options like reading from a file, writing to a file, clearing the text area and other options are provided by making use of menus.

Auto correction of words can lead to further context sensitive errors. Hence Auto-correct option is provided as an option which the user can either enable or disable depending upon his requirements. When Auto-correct option is enabled all the misspelled words that have only one suggestion will be corrected automatically without any user intervention. Multiple suggestions for a single misspelled word is provided with a combo box in which user can make a choice out of the listed possible words. The processed text is further displayed in another panel with different colors viz. red for misspelled word, green for correct words, magneta for auto-corrected words and blue for the words not found in the dictionary. To improve the performance of the tool multi-threaded approach is used [16]. Searching in multiple sub-dictionaries is done simultaneously by multiple threads. As the tool uses multi-way searching in a trie data structure the search efficiency also increases. The usage of Levenshtein edit distance ensures that suggestions for all types of non-word errors are provided properly. To make it easier for the first time users a detailed description on usage of the tool has been provided in help option.

The tool has been tested extensively for various test cases. The tool is found to provide correct results for both single word and also for a block of text. Test cases were designed to cover all types of errors. Figure 4 shows the results produced by the tool. The word "ರಾಮುನಿ" has its suffix wrong and the word "ಶಾಲಿಗೆ " has its root wrong. While the word "ಅತುರದಿಂದ" is not in dictionary and the word "ಹೋದನು" is spelled correctly. The suggestions for the two misspelled words are shown in the combo boxes in the lower half of the snapshot.

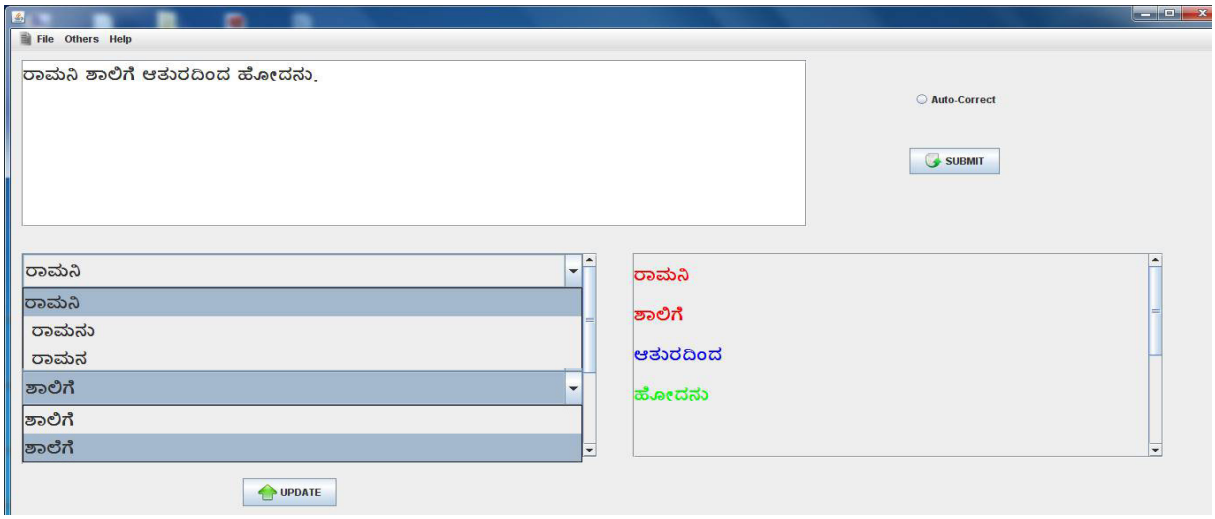

Figure 4: Snapshot of input involving all test cases**.**

At the time of the development of the tool there were no well-known spell checkers for Kannada that can handle block of text. Hence the comparison has been made with a basic spell checker developed by Kannada Kasturi. This software handles only one word at a time. For the word ರಾಮನಿಂದೆ, result from our spell checker tool and Kannada Kasturi are shown in Figure 5 and Figure 6 respectively. The two snapshots clearly show the efficiency of the tool developed by us. Suggestions provided by the Kannada Kasturi spell checker are more in number and are irrelevant to the actual word. It actually displays all the words in the dictionary starting with 'ರಾ'. In the other snapshot one can see that our tool provides precise suggestions.

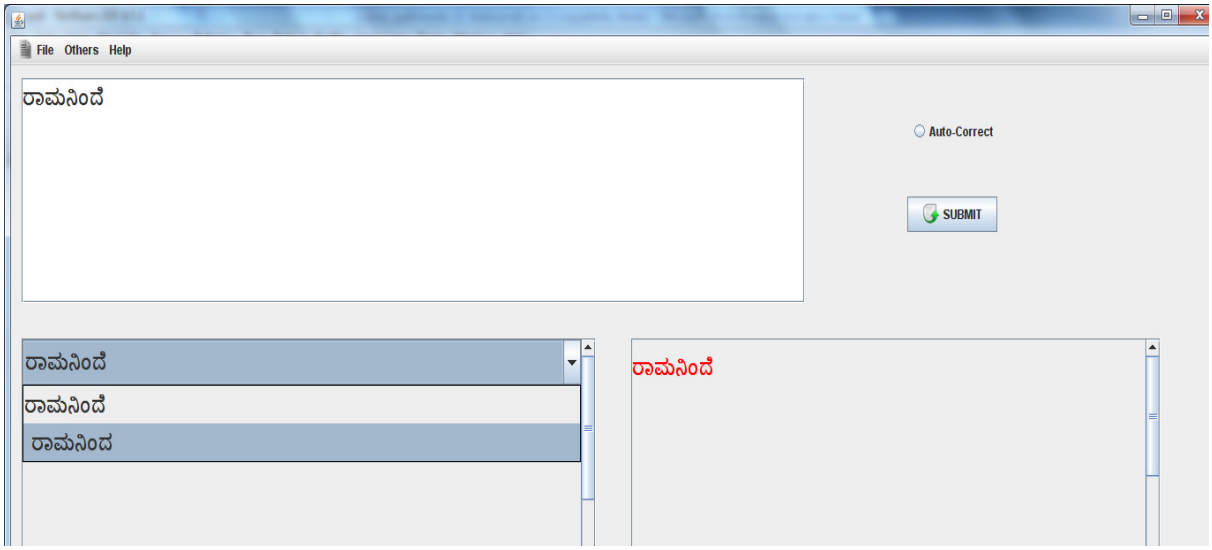

Figure 5: Snapshot of our Spell Checker

VI. CONCLUSION

A Spell Checker that is capable of identifying non-word errors and providing suggestions is implemented. The application is both space and time efficient as it uses the trie data structure and splits the large dictionary into a set of dictionaries of smaller size. The Spell Checker developed performs better in comparison with commercially available Spell Checkers. Upon finding the error, it suggests only those words that are similar to the erroneous word, where similarity is measured by Levenshtein distance.

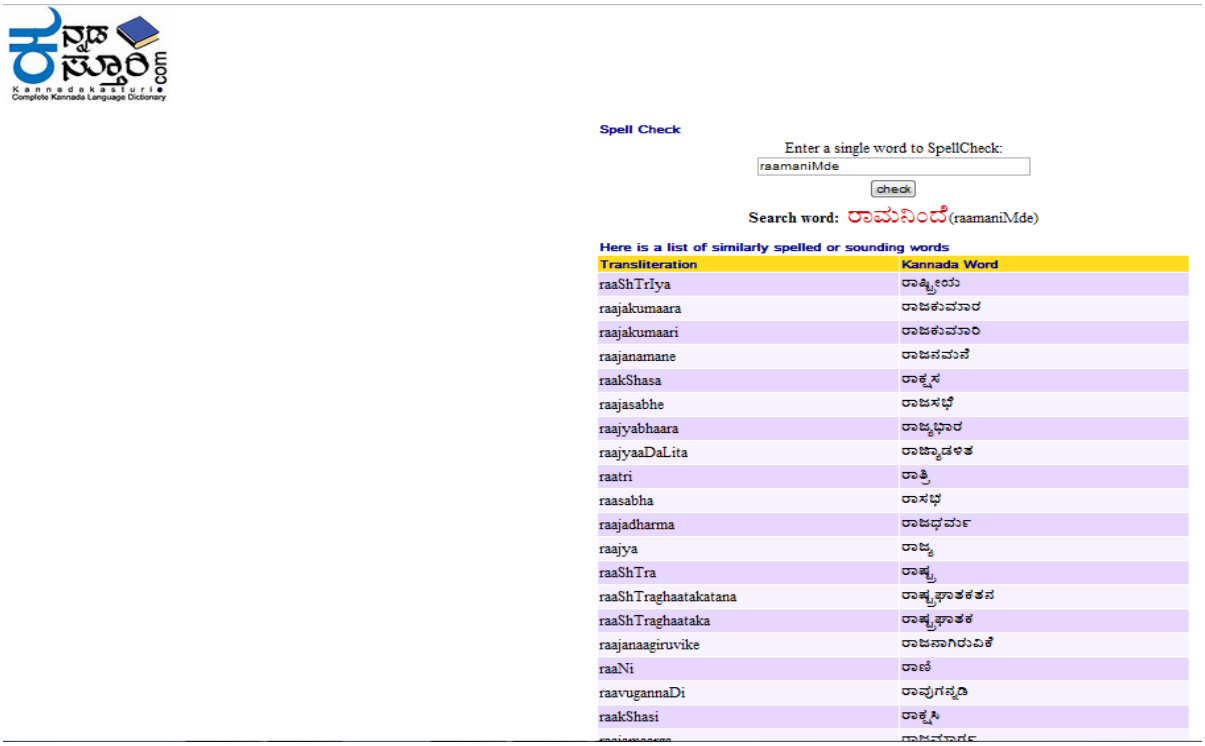

Figure 6: Snapshot of Kannada Kasturi for single word

A rule based questionnaire is prepared that is used to add new words into the dictionary. It helps in enhancing the size of the dictionary dynamically. Though auto-correction is not encouraged, auto-correction option is provided. With all these features, the tool is useful as a stand-alone component or plug-in for many of the natural language processing tools.

ACKNOWLEDGEMENTS

The authors wish to thank Dr. B.P.Hemaananda, linguist for the discussions in deciding Paradigm tables. We also thank Shambhavi B.R., Vignesh C. Gadiyar, and Alok B., for their help in the development of the tool.

REFERENCES

[1] Indian Census Data – 2001 http://censusindia.gov.in/Census_Data_2001/Census_Data_Online/ Language / Statement3.htm
[2] History of Kannada Language: http://www.lisindia.net/Kannada/Kan_hist.html
[3] F.J. Damerau, “A technique for computer detection and correction of spelling errors”, Communication of ACM, 7(3), 171-176, 1964.
[4] B. B. Chaudhuri, “Reversed word dictionary and phonetically similar word grouping based spell-checker to Bangla text”, Proc. LESAL Workshop, Mumbai, 2001.
[5] Kunte, R.S., An OCR System For Printed Kannada Text Using Two - Stage Multi-Network Classification Approach Employing Wavelet Features, Conference On Computational Intelligence And Multimedia Applications, 2007. International Conference On 13-15 Dec. 2007.
[6] Karen Kukich, Automatically correcting words in text, ACM computing surveys, volume-24, no.4, Dec-1992.
[7] E.Fredkin, “Trie Memory”, Communications of the ACM, Volume 3 Issue 9, September 1960, Pages 490-499
[8] Ranjan Sinha, David Ring and Justin Zobel , Efficient String Sorting Using Copying - School of Computer Science and Information Technology RMIT University, Australia
[9] H. Shang and T.H. Merrett, Tries for Approximate String Matching, September 8, 1995
[10] Hanada, H. , A practical comparison of edit distance approximation algorithms , Granular Computing (GrC), 2011 IEEE International Conference on 8-10 Nov. 2011.
[11] Gusfield, Dan (1997). Algorithms on strings, trees, and sequences: computer science and computational biology. Cambridge, UK: Cambridge University Press. pp. 263–264. ISBN 0-521-

58519-8.
[12] Mallamma V Reddy, Hanumanthappa M, "English To Kannada/Telugu name transliteration in CLIR: A statistical Approach", International Journal of Machine Intelligence, Volume 3, Issue 4, 2011, pp-340-345
[13] Kannada Unicode Chart extracted from Unicode consortium , http://unicode.org/charts/PDF/U0C80.pdf
[14] Shambhavi. B. R , Dr. Ramakanth Kumar P , Srividya K, Jyothi B J, Spoorti Kundargi, Varsha Shastri G , Kannada Morphological Analyser and Generator Using Trie ,IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.1, January 2011
[15] S.N. Sridhar ,"KANNADA", Rotledge, 1990
[16] Daniel Thunell, Benefits of Multi-Threaded Applications, Master of Science Thesis Stockholm, Sweden 2007.

About the Authors

Rajashekara Murthy S, obtained his Master's Degree in Computer Engineering from VTU and Bachelor's degree in Computer Science & Engineering from Bangalore University. His research interests are Natural Language Processing and Algorithms. He has guided more than 15 undergraduate projects and 5 post graduate projects. He has to his credit1 conference and 1 International Journal paper.

Vadiraj Madi, is currently pursuing his bachelor degree in computer science and engineering. He is currently in 8th semester (4th year). He interned at MindTree, Bangalore from June 2011 to August 2011. He is an ORACLE Certified Java Programmer. He has 1 conference paper to his credit.

Sachin D, is currently pursuing his bachelor degree in computer science and engineering. He is currently in 8th semester (4th year). He worked in the project titled "Handwriting Recognition on tablet PC" at IISc Bangalore. He is an ORACLE Certified Java Programmer. He has 1 conference paper to his credit.

Ramakanth Kumar P, was awarded Doctorate from Mangalore University, has teaching experience of around 14 years in academics and Industry. His area of research is on Artificial Intelligence, Pattern recognition. He has to his credits 03 National Journals, 02 International Journals, 12 Conferences and 15 Research Publications. He has guided 14 M.Tech. students and currently supervising 05 Ph.D. students.