IJCSC 0973-7391 Vol- 6 • Issue-1 Sep - Mar 2015 pp.140-144 **Impact Factor-2.5** available at www.csjournalss.com

AVL and TRIE Loading Time in Dogri Spell Checker

Shubhnandan S. Jamwal
PG Department of Computer Science and IT, University of Jammu, Jammu
jamwalsnj@gmail.com

Abstract: Spellcheckers are the basic tools needed for word processing and document preparation. In this paper we have implemented the spell checker for Dogri language based on the existing design [1]. The existing algorithms and techniques that are being used to check the spelling and to generate efficient suggestions for mis-spelt words of English and other Western languages are not actually suitable for Dogri and most of the Indian Languages. The dictionary has been implemented in the spell checker using Trie and AVL data structure. The construction time of the data structure in loading all the words of the dictionary has been discussed in this paper. Most the spell checker in Indian Languages has been developed using the AVL data structure. In this research it has been found that the Trie performs better than the AVL in Dogri dictionary of 2000 words.
Keywords: Dogri, Spellchecker, Suggestion List, Tokens, Dictionary.

Introduction

Spellcheckers are the basic tools needed for word processing and document preparation. A spell checker is a tool that enables us to check the spellings of the words in a text file, validates them i.e. checks whether they are right or wrongly spelled and in case the spell checker has doubts about the spelling of the word, suggests possible alternatives

The ideal spell checker would offer one word on the list, and that would be correct. If ten words are offered and the correct word is near the bottom, a poor speller must read through the choices and disregard the first few despite their precedence, instead looking for the word which they know, in some way, to be correct. If so many words are offered that they cannot all be seen at once, it is even more difficult to find the correct word. So in good spell checker the most important consideration is number of effective suggestion offered by the spell checker and they should be less in number as far as possible.

The main task performed by the spell checker is to take the word from the file as its input and look for it in the user defined dictionary. If the word matches any of the word in the dictionary, next word is fetched form the input file. But if the word is not found in the user defined dictionary, then suggestion list of the closest matching patterns is generated based on certain algorithms.

In this paper we have designed the add-in (SpellChecker_Dogri) for the standard word processor i.e., MSWord. The basic step in developing the Word processor solutions for Dogri language needs interaction with the Word object model of .NET. It provides the set of development tools (VSTO-Visual Studio Tools for office) in the form of add-in. The primary interop assembly for word is provided with various classes and interfaces that are defined in the namespace Microsoft.Office.InteropWord. The objects of this word object model are used to interact with MS-Word. Many Properties and methods of these objects are used to develop Application-Level SpellChecker_Dogri.

Brief Description of Dogri Language

Dogri is an Indo-Aryan language. Though it is chiefly spoken in the scenic region of Jammu, the presence of Dogri language can also be felt in northern Punjab, Himachal Pradesh and other places. The people speaking Dogri are called Dogras, whereas the belt where it's spoken is called Duggar. Being the second prominent language of J&K State, it has an important place on the linguistic map of Northern India. Dogri is a member of the Western Pahari group of languages. Originally, this language was written using the Takri script, but now the Devanagari script is employed for the same in India. Dogri language has its own grammar and it own dictionary. The Grammar of Dogri also has a very strong Sanskrit base. Dogri is a phonetic Language and commonly written in Devanagari script. Some of the major properties of the Devanagari alphabet are:

- Devanagari script is evolved from Brahmi script. Its letters usually align the below line of writing.
- It has 47 primary symbols:10 for vowels and 37 for consonants.
- Apart from these, there are nine symbols for vowel matras, one for halant, an apostrophe comma for high falling tone and other symbol for extra length i.e. avgrah (093D).
- Like Hindi language all consonants are written with inherit∴' vowel. Otherwise the consonant symbol mark of (◌्) is used below consonant symbol [2].

The complete character set of Devanagari is depicted in Fig.1.

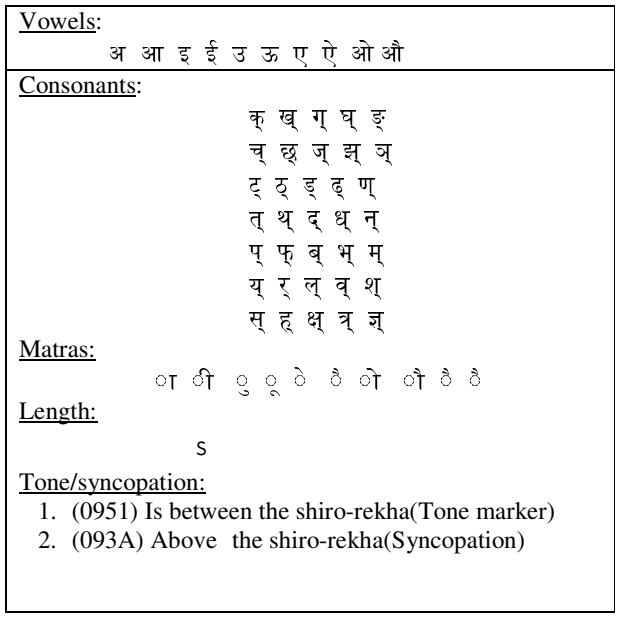

Figure 1: Devanagari Character Set

Spell Checker Architecture

The major components of the spell checker architecture are shown in the Fig. 3. These components are:

- Tokenisation and normalisation,
- Lexicon Lookup/Error Detection Module and
- Suggestion Module [1].

The basic functions of the different modules are explained as follows:

Tokenisation:

Tokenisation refers to process of breaking the text into tokens or words using punctuation marks and spaces as delimiters. The spellchecker reads the text character by character to tokenise the text. As the space or punctuation marks is encountered in the running text the tokeniser will break the line into following tokens: As for example, the following text:

उस बेल्लै दपैह्री दे बारां बज्जा करदे हे ।

The line is broken into 8 words separated by the blank space and the

Normalisation:

The tokens are then made to pass through a normalisation process to convert them to the format in which the lexicon has been stored. Some of the major steps that are performed in this stage are removal of redundant zero width characters and the conversion of the Dogri character into Unicode equivalent. For example, the word झूर is typed in any Dogri font will be normalised as {0919,0942,0930}.

Lexicon Lookup/Error Detection:

In this module the normalised token is searched in the standard dictionary. The standard dictionary words stored in the database has been partitioned into sub-dictionaries based on the word length and at execution time each of these sub-dictionaries is loaded in a height balanced binary search tree (AVL tree). To search for a word of length n, we look for its presence in the AVL tree storing words of length n.

The standard dictionary developed for the testing of the spell checker engine is loaded as avl when Microsoft-word loads the SpellChecker_Dogri. In this process Lexicons are read from the Standard dictionary and the length of lexicon having maximum number of characters is used to create an array of AVL tree objects. After that the length of the each of the lexicon is determined and added in suitable AVL object so that the words are available in minimum time during the generation of the suggestion. Thus झूर will be searched in the AVL tree storing lexicon words of length 3. If the word is not found, then it is searched in the AVL tree storing the lexicon of custom dictionary. If the word is still not found, then it is marked and sent to Suggestion module.

Suggestion:

In this module a list of possible correct words is presented to the user using the lexicons stored in the dictionary. The correct words are reflected on the screen and the user selects the correct word. The user can give the command to replace a single or all occurrences of the mis-spelled word with the word selected in the suggestion list. If the word is not in the dictionary the word can be added in the custom dictionary.

The suggestion list module detects an erroneous word and it generate a list of candidate corrections, rank the spelling variations and select the highest ranking as the most likely correction. In this research paper we have implemented the reverse minimum edit distance to generate the suggestion list. The main time in reverse minimum edit distance approach is spent on checking the dictionary for valid word. In this paper about 20 commonly misspelled words are used in testing. These are common errors generated by typists of the Dogri language and the common errors are identified in the books coming for proof reading to experts. Typing were analysed and it was found that the following six types of Dogri typing errors occur:

i. Insertion error (IE): When at least one extra character is inserted in the desired word. होनहार -> होनहमार

ii. Deletion error (DE): When at least one character is deleted in the desired word. होनहार -> होनहर

iii. Substitution error (SE): When at least one character is substituted by the other character.
होनहार -> होनहमर

iv. Transposition error (TE): When two adjacent characters are transposed. होनहार -> होनहरा

v. Run-on error (ROE): When there is space missing between two or more valid words**.**
होनहार हून-> होनहारहून

vi. Split Word error (SWE): This is Opposite of Run-on error when some extra space is inserted between parts of a word. The error can be removed by removing the extra space होनहार -> हो नहार

After sorting procedure is executed the sorted suggestion list is presented to the user for convenience. The user selects the right word and wrongly spelled word is replaced by the selected word. After changing the word the editor proceed with the next error. In order to sort the suggestion list most usefully, we have used two parameters; frequency of occurrence of the suggested word and the smallest number of substitutions, insertions and deletions required in that order to change the misspelt word to the suggested word.

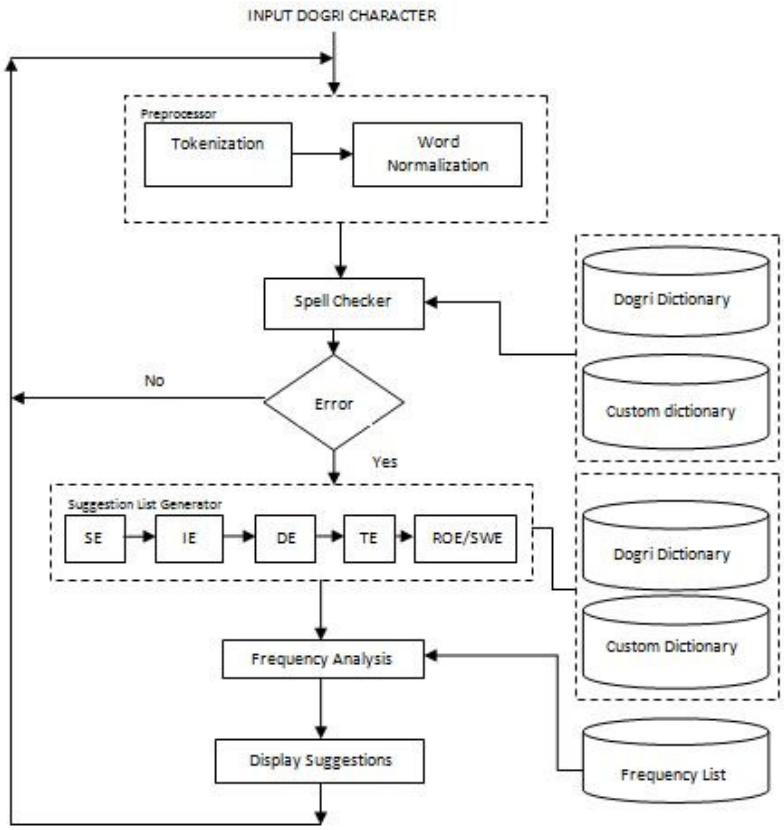

Figure 3: Spell Checker Architecture

Evaluating the Suggestion List:
As described in the earlier sections that an ideal spell checker is the one that will give only one suggestion list or it may be the first word in the suggestion list. Another important issue is in the generation of the suggestion list.

Here the number of possible correction is considerable issue. The count of the words offered in the suggestion list and whether the desired word appears on the suggestion list are the two major issues, addressed in the suggestion list generation.

Experimentation

In this paper we have designed the add-in (SpellChecker_Dogri) for the standard word processor i.e., MSWord. The basic step in developing the Word processor solutions for Dogri language needs interaction with the Word object model of .NET. It provides the set of development tools (VSTO-Visual Studio Tools for office) in the form of add-in. The primary interop assembly for word is provided with various classes and interfaces that are defined in the namespace Microsoft.Office.InteropWord. The objects of this word object model are used to interact with MS-Word. Many Properties and methods of these objects are used to develop Application-Level SpellChecker_Dogri.

In this paper, AVL tree and Trie data structure for English dictionary have been modified to suit the non-linear nature of Dogri. A comparison is made to determine which data structure gives the best performance, in terms of loading time at the creation of the data structure. Because most of the dictionaries are implemented using the AVL data structure for the Indian languages. We have implemented two data structures i.e. AVL and Trie for Dogri dictionary of 2000 words. Results were compared and found that the TRIE data structure takes less time in loading the words and, whereas, the AVL takes more time to load into the memory. The results of experiment are depicted in Table 1and visually represented in figure 3.

Dictionary Size	AVL	Trie
5	105	79
10	109	86
20	107	103
40	104	84
80	105	96
160	94	84
320	104	98
640	131	115
1705	126	86

Table 1

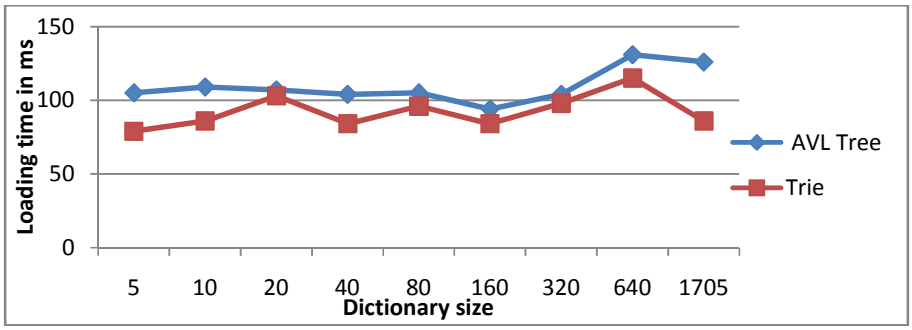

Figure 3

Conclusion and Future Scope

In this paper we have implemented the spell checker for Dogri language using the AVL and Trie data structure. We have implemented the spell checker in .Net platform using C#. The main issue which is addressed in the paper is the loading time of the dictionary. It has been found and observed that Trie performs better than the AVL in implementing the Dogri dictionary. The phonetic similarity of the suggested word is not addressed here.

References

1. G S Lehal, "Design and Implementation of Punjabi Spell Checker", *International Journal of Systemics, Cybernetics and Informatics,* pp. 70-75 (Jan 2007).
2. www.ciil-lisindia.net/Dogri/Dogri.html
3. E. Brill and R. Moore, "An improved error model for noisy channel spelling correction," *Proceedings of the ACL 2000*, 2000, 286-293.
4. A. R. Golding, "A Bayesian hybrid method for context- sensitive spelling correction ," *Proceedings of the Workshop on Very Large Corpora*, 1995, 39-53.
5. A.R. Golding and D. Roth, "Applying winnow to context-sensitive spelling correction," *Proceedingsof ICML*, 1996, 182-190.
6. K. Kukich, "Techniques for automatically correcting words in a text," *Computing Surveys*24(4), 1992, 377-439.
7. E.J. Yannakoudakis and D. Fawthrop,"An Intelligent spelling corrector," *InformationProcessing and Management* 19(12), 1983, 101-108