IJITKM 0973-4414 Volume-9, Number-1 Jun-Dec 2015 pp. 95-99 Impact Factor 2.8 available online at www.csjournalss.com

AVL and TRIE Look Up time in Implementation of Dogri Spell Checker

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

Abstract: In this paper look up time of the word in the Dogri dictionary are presented. The dictionary has been implemented and tested for different set of sizes. AVL tree and Trie data structure for English dictionary have been modified to suit the non-linear nature of Dogri. A comparison is made is to determine which data structure gives the best performance, in terms of look up time. We have implemented two data structures i.e. AVL and Trie for Dogri dictionary of approx 2000 words. The results reveals that the AVL data structure takes more time in finding the word for the Dogri dictionary.
Keywords: Dogri, Spellchecker, Suggestion List, Tokens, Dictionary.

Introduction

Spellcheckers are the basic tools needed for word processing and document preparation. Till now there is no spell checker tool available for the Dogri language. In this paper we have implemented the spell checker for Dogri language based on the existing design [1]. Designing a spell checker for Dogri language poses many new challenges not found in other western languages. Dogri language is far different from Western languages in phonetic properties and grammatical rules. Thus 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; rather it needs different algorithms and techniques for expected efficiency. In this research paper the minimum edit distance algorithms is used to generate the suggestion and spell checker has been implemented on MSword 2007.

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.

Vowels:
Consonants: Matras: Length: Tone/syncopation: 1. (0951) Is between the shiro-rekha(Tone marker) 2. (093A) Above the shiro-rekha(Syncopation)

Figure 1: Devanagari Character Set

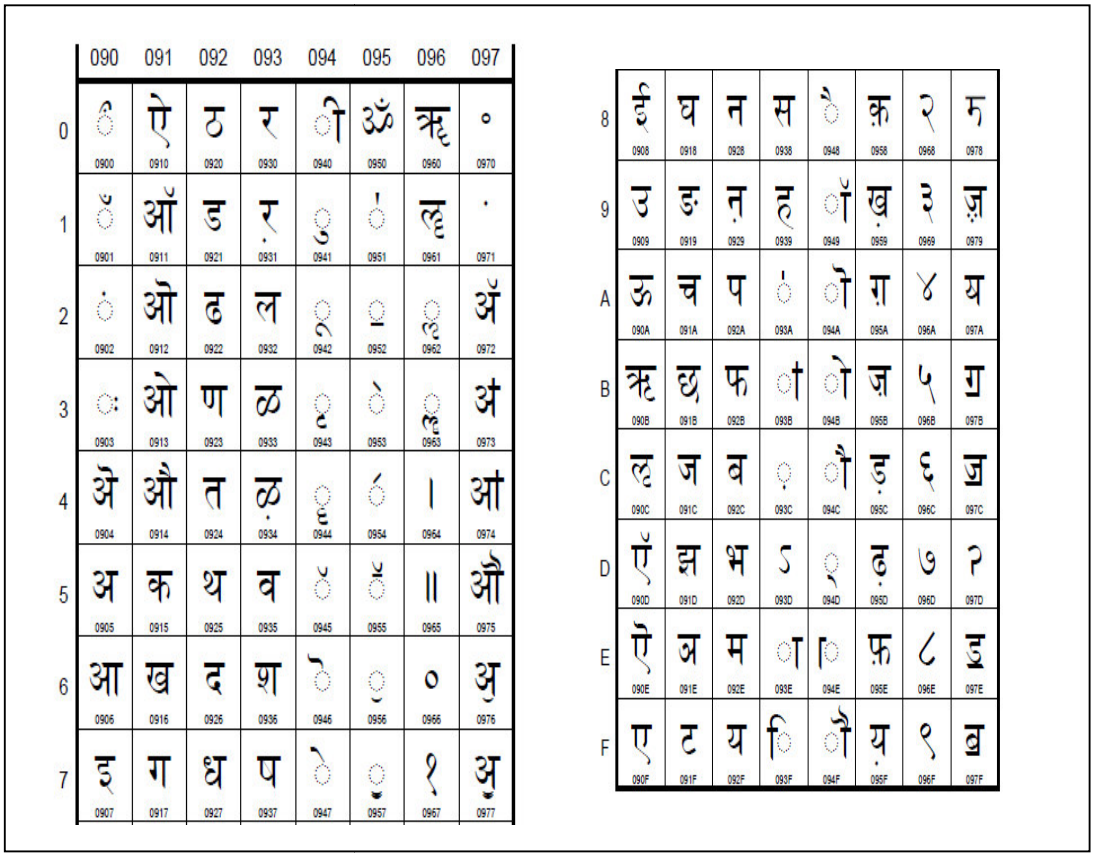

Figure 2: Devanagari Coding Scheme:

Lexicon creation

The first step in implementation of the spell checker is the creation of a lexicon of correctly spelled words, which will be used by the spell checker to check the spellings as well as generate the suggestions. The lexicon has been stored in the access database and around 2000 words have been used in the implementation of the spell checker. We will store all the possible forms of words of Dogri lexicon in the access dictionary. In the current implementation we have stored around 2000 lexicon of the language for testing of the spell checker engine. Around 22000 unique words have been identified and shall be stored in the spell checker engine of the database. During the execution of the program, the words are loaded into AVL trees and different AVL trees will be used for different word lengths.

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:

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 is typed in any Dogri font will be normalised as { }.

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.

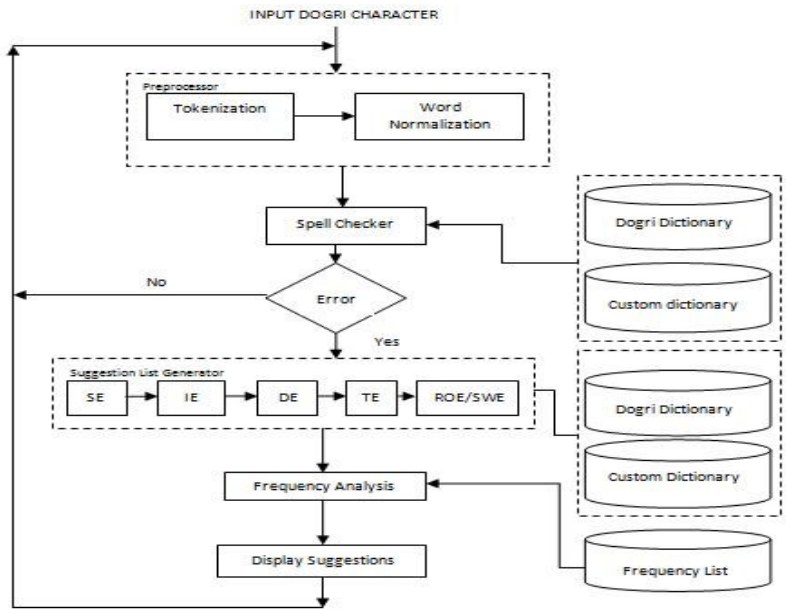

Figure 3: Spell Checker Architecture

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 will be searched in the AVL tree storing lexicon words of length . 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.

Experimentation

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 is to determine which data structure gives the best performance, in terms look up time. We have implemented two data structures i.e. AVL and Trie for Dogri dictionary of approx 2000 words. The programming language used in the experiments is the C#. The results of experiment are depicted in Table 1 and visually represented in figure 3. The results of experiment done on these implementations in context of and look-up time are shown in Table 1 and represented pictorially in figure 3.

Dictionary Size	AVL	Trie
5	52	7
10	78	7
20	103	8
40	106	14
80	155	18
160	163	23
320	222	39
640	396	65
1705	492	316

Table 1

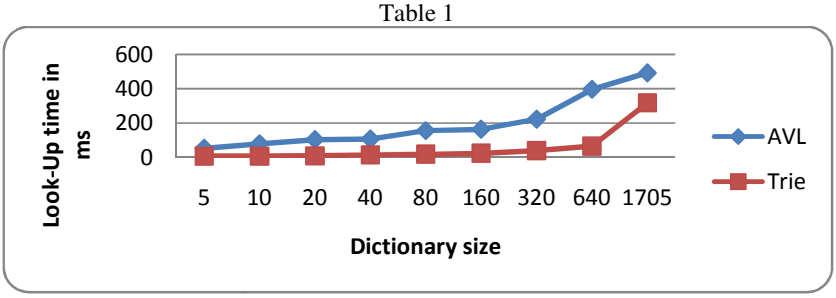

Figure 3

Conclusions

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 look up time of 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