International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 8, Issue 9, September 2019)

Study on Spell-Checking System using Levenshtein Distance Algorithm

Thi Thi Soe[1], Zarni Sann[2]
[1]*Faculty of Computer Science*
[2]*Faculty of Computer Systems and Technologies*
[1,2]*FUniversity of Computer Studies (Mandalay), Myanmar*

***Abstract*— Natural Language Processing (NLP) is one of the most important research area carried out in the world of Artificial Intelligence (AI). NLP supports the tasks of a portion of AI such as spell checking, machine translation, automatic text summarization, and information extraction, so on. Spell checking application presents valid suggestions to the user based on each mistake they encounter in the user's document. The user then either makes a selection from a list of suggestions or accepts the current word as valid. Spell-checking program is often integrated with word processing that checks for the correct spelling of words in a document. Each word is compared against a dictionary of correctly spelt words. The user can usually add words to the spellchecker's dictionary in order to customize it to his or her needs. In this paper, the system is intended to develop a spell checker application program by using Levenshtein Distance algorithm.**

***Keywords*—Artificial Intelligence, Spell checker, Levenstein distance.**

I. INTRODUCTION

Natural Language Processing is massively locally ambiguous at every level. In the language translation, the spell checking problem is one of the most important issues. A spell checker is an application program that flags words in a document that may not be spelled correctly. Spell checkers may be stand-alone capable of operating on a block of text, or as part of a larger application, such as a word processor, email client, electronic dictionary, or search engine. A spell checker consists of two parts: a set of routines for scanning text and extracting words, and an algorithm for comparing the extracted words against a known list of correctly spelled words (ie. the dictionary). Simple spell checkers operate on individual words by comparing each of them against the contents of a dictionary, possibly performing stemming on the word.

If the word is not found, it is considered to be an error, and an attempt may be made to suggest that word. When a word which is not within the dictionary is encountered, most spell checkers provide an option to add that word to a list of known exceptions that should not be flagged. An adaptive spelling checker tool based on Ternary Search Tree data structure is described in [1]. A comprehensive spelling checker application presented a significant challenge in producing suggestions for a misspelled word when employing the traditional methods in [2]. It learned the complex orthographic rules of Bangla. The objectives of the paper are to study how spell-checking works and how the suggestions are produced, to realize and apply Levenshtein Distance algorithm in developing a spell checking system. The organization of remaining sections is as follows: Section II describes the basic aspect of Levenshtein Distance algorithm. In section III, we devote the design process to build the spell checking system. Initial testing is carried out on the system, which is explored in section IV. Finally, we conclude the paper in section V.

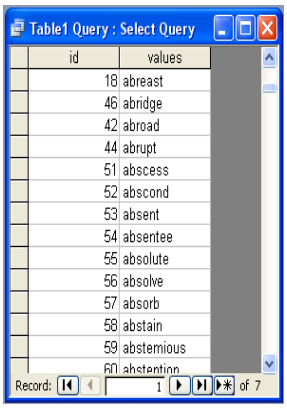
II. METHOD AND MATERIAL

A. Lexicon

Lexicon contains all of the words that the program is capable of recognizing. The parser works in conjunction with a lexicon, or dictionary. The parser searches through the lexicon comparing each input word with all of the words stored there. For this process, the system has an interface to update the user lexicon because the new word is necessary when they are added to the user lexicon. The lexicon is presented in table form as shown in Table I. The parser is required in the first step which is breaking input text into individual words or token.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 8, Issue 9, September 2019)

TABLE I
WORD TABLE



B. Levenshtein Distance

Levenshtein Distance is named after the Sviet mathematician, Vladimir Levenshtein, who devised the algorithm in 1965 [3]. The Levenshtein Distance (LD) algorithm has been used in: spell checking, speech recognition, DNA analysis, and plagiarism detection. Levenshtein Distance (LD) is a measure of the similarity between two strings, which refers to as the source string, s, and the target string, t. The distance is the number of deletions, insertions, or substitutions required to transform s into t. The processing steps for the algorithm is as following:

TABLE III
STEPS FOR EDIT DISTANCE

Step	Description
1	Set n to be the length of s. Set m to be the length of t. If n = 0, return m and exit. If m = 0, return n and exit. Construct two vectors, $v0[m+1]$ and $v1[m+1]$
2	Initialize $v0$ to 0..m.
3	Examine each character of s (i from 1 to n).
4	Examine each character of t (j from 1 to m).
5	If $s[i]$ equals $t[j]$, the cost is 0. If $s[i]$ is not equal to $t[j]$, the cost is 1.
6	Set cell $v1[j]$ equal to the minimum of: a. the cell immediately above plus 1: $v1\ [j-1]$ + 1. b. The cell immediately to the left plus 1: $v0\ [j]$ + 1. c. The cell diagonally above and to the left plus the cost: $v0[j-1]$ + cost.
7	After the iteration steps 3 to 6 are completed, the distance is found in the cell $v1[m]$.

As a case study, below processes illustrate how the Levenshtein Distance is computed when the source is “GUMBO” and the target string is “GAMBOL”.

		G	U	M	B	O
	0	1	2	3	4	5
G	1					
A	2					
M	3					
B	4					
O	5					
L	6					

		G	U	M	B	O
	0	1	2	3	4	5
G	1	0				
A	2	1				
M	3	2				
B	4	3				
O	5	4				
L	6	5				

		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1			
A	2	1	1			
M	3	2	2			
B	4	3	3			
O	5	4	4			
L	6	5	5			

		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1	2		
A	2	1	1	2		
M	3	2	2	1		
B	4	3	3	2		
O	5	4	4	3		
L	6	5	5	4		

		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1	2	3	
A	2	1	1	2	3	
M	3	2	2	1	2	
B	4	3	3	2	1	
O	5	4	4	3	2	
L	6	5	5	4	3	

		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1	2	3	4
A	2	1	1	2	3	4
M	3	2	2	1	2	3
B	4	3	3	2	1	2
O	5	4	4	3	2	1
L	6	5	5	4	3	2

Step 7

The distance is in the lower right hand corner of the matrix, $v1[m]$ == 2. This corresponds to our intuitive realization that "GUMBO" can be transformed into "GAMBOL" by substituting "A" for "U" and adding "L" (one substitution and one insertion = two changes).

III. System Architecture

There are five components to provide overall system implementation. Three components such as parsing on each word, recognizing the meaning of contents and applying Levenshtein distance algorithm for suggestion the number of edit times. As a result, the user can get correct spelling words.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 8, Issue 9, September 2019)

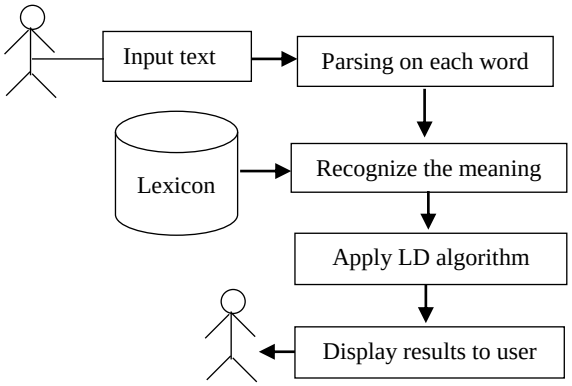


Fig. 1: System Architecture

IV. Initial Investigation

To test the implemented system, the user must enter the sentence into the working area. And then, the user can check the spelling as shown in fig. 2.

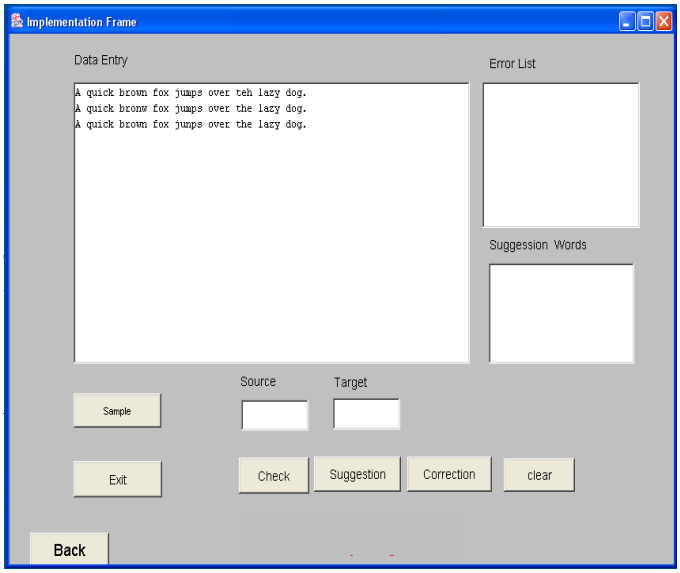


Fig. 2: System User Interface

The system matches the input words with Database. If the spelling error is found, the system displays the error words and possible suggestion lists to the user as shown in fig. 3(a) and (b) respectively. In this way, the system generates the correct sentence.

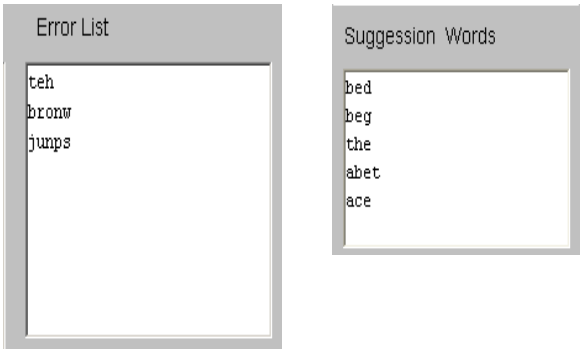


Fig. 3: Error list and Possible Correct Words

V. Conclusion

In this paper, the presented spell checking system provided the spelling checking based on Levenshtein Distance Algorithm as well as the spelling check of English words or sentence. By using this system, it supports the correct spelling for English words occurrence. It can help to overcome the language barrier between people using the English language. In this system, the lexicon contains nearly about 3000 words. So, the system cannot check more than 3000 words. But, the administrator can add more common words to the system. The system cannot check the grammar rules. Complete Dictionary and grammar rules can be added into the existing system. This system can be extended to implement the sentence patterns and language translation.

REFERENCES

[1] B.Loghman, Q.Z.Behrang, "CloniZER Spell Checker Adaptive, Language Independent Spell Checker" AIML 05 Conference, 19-21 December 2005

[2] U.Z.Naushad and K.Mumit "A Comprehensive Bangla Spelling Checker", Center for Research on Bangla Language Processing, BRAC, University, Bangladesh.

[3] Levenshtein, Vladimir I. "Binary codes capable of correcting detections, insertions, and reversals", February 1996.