IMPLEMENTATION OF SPELL-CHECKING SYSTEM BY USING LEVENSHTEIN DISTANCE ALGORITHM

Win Ko
University of Computer Studies in Mandalay
winkoucsm@gmail.com

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, and 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. This system is intended to develop a spell checker (or spell check) application program by using Levenshtein Distance algorithm.

1. Introduction

In computing, a spell checker (or spell check) 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. 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 a word that was likely to have been intended. One such suggestion algorithm is to list those words in the dictionary having a small Levenshtein distance from the original 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. A spell checker customarily consists of two parts:

(1) A set of routines for scanning text and extracting words, and
(2) An algorithm for comparing the extracted words against a known list of correctly spelled words (ie. the dictionary).

The scanning routines sometimes include language-dependent algorithms for handling morphology. Even for a lightly inflected language like English, word extraction routines will need to handle such phenomena as contractions and possessives. It is unclear whether morphological analysis provides a significant benefit [5].

The word list might contain just a list of words, or it might also contain additional information, such as hyphenation points or lexical and grammatical attributes. As an adjunct to these two components, the program's user interface will allow users to approve replacements and modify the program's operation.

The rest of the paper is organized as follows: Section 2 describes related work. In Section 3, we briefly described tokenization, NLP process, lexicon; levenshtein distance algorithm.Section 4 describes the implementation details of our system. Finally we conclude the paper in Section 5.

2. Related Work

In the Artificial Intelligence field, the utilization of all the level of language processing explained above offers the potential for truly habitable dialogue systems [7]. Most spelling checkers allow the user to add custom words to the spelling checker's vocabulary. Usually, the user also has the option to ignore specific errors [6]. An adaptive spelling checker tool based on 'Ternary Search Tree' data structure is described in [5]. It learned media error pattern reposed by an algorithm and also improved its suggestions with nondeterministic traverse. The comparing results on a variety of algorithms with the proposed method are eventually concluded according to the suggestions of number of iteration tones. Consequently, it is able to give more flexibility, more accuracy, data compression rate, and reliability with comparing to other proposed methods. Alternatively, the used method can adapt and tune itself by interactions by user or outer media and it improves its suggestion list as time goes by.

A comprehensive spelling checker application presented a significant challenge in producing suggestions for a misspelled word when employing the traditional methods in [4]. It learned the complex orthographic rules of Bangla. It described the process of checking the spelling of a Bangla document, compares the methodologies with existing solutions available in the literature, and then proposed solutions for each step. It is shown by the performance and evaluation of the proposed solution. It discussed the existing solutions and

explored their limitations, and proposed a complete spell checking methodology for Bangla.

3. Theoretical background

3.1 Tokenization

Morphological analysis splits one token into several. A token is used to separate by morphological analysis; A token identifies a unit of information. Morphological analysis splits one token into several. At first step of processing is to divide the input text into units called Tokens. Usually, tokens are the results of some processing part that has performed lexical analysis and divided a data set into the smallest units of information used for subsequent processing. A token is an occurrence of a term from the text of the field. It consists of a term's text, the start and end offset of the term in the text of a field, and a type string. The start and end offsets permit applications to re-associate a token with its source text. The type is an interned string, assigned by a lexical analyzer naming the lexical or syntactic class that the token belongs to. Words are not always surrounded by white space. Often punctuation marks attach to words, such as commas, semicolons, and periods (full stops). It at first seems easy to remove punctuation marks from word tokens.

Many Languages do not put spaces in between words at all, and so the basic word division algorithm of breaking on white space is of no use at all. Such languages include major East-Asian languages; they are Myanmar, Chinese, Japanese, and Thai. Ancient Greek was also written by Ancient Greeks without word spaces. Spaces were introduced by those who came afterwards. In such language, word segmentation is a much more major.

3.2 Process of NLP

Natural language process (NLP) system can after possible solution to the problem of communication between human and computer. There are five major element in NLP system;

(1) Parser
(2) Lexicon
(3) Knowledge Base
(4) Understander
(5) Generator [1].

3.2.1 Lexicon

Lexicons are the heart of any language processing system. They include the vocabulary that the system can handle both individual lexical items and multi-word phrases. A lexicon contains a complete inventory of all words in a language along with relevant information conforming to the specifications of a given theoretical framework. A complete lexicon is supposed to contain entries for all morphemes (meaningful units that make up words) for the language. A lexicon is a list of words in a language, for instance, the vocabulary along with some knowledge of how each word is used. A lexicon may be general or domain-specific, for example, a lexicon of several thousand common words or a lexicon of the technical terms of dentistry in some language. The term 'lexicon', on the other hand, is used in a technical sense in linguistics.

Each word or phrase in a lexicon is described in a lexical entry. To say exactly what is included in each entry depends on the purpose of the particular lexicon. The details that are given may include any of its properties of spelling or sound, grammatical behavior, meaning or use and the nature of its relationships with other words. Traditionally, lexicons are quite small and are constructed largely by hand. There is a growing realization that effective NLP requires increased amounts of lexical information. A recent trend has been the use of automatic techniques applied to large corpora for the purpose of acquiring lexical information from text. Statistical techniques are important aspect of automatically mining lexical information [2].

3.3 Levenshtein Distance (LD) Algorithm

In information theory and computer science, the Levenshtein distance is a metric for measuring the amount of difference between two sequences (i.e., the so called edit distance). The Levenshtein distance between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character. A generalization of the Levenshtein distance (Damerau–Levenshtein distance) allows the transposition of two characters as an operation. Some Translation Environment Tools, such as translation memory leveraging applications, use the Levenhstein algorithm to measure the edit distance between two fuzzy matching content segments [8]. For example,

- If s is "test" and t is "test", then LD(s,t) = 0, because no transformations are needed. The strings are already identical.
- If s is "test" and t is "tent", then LD(s,t) = 1, because one substitution (change "s" to "n") is sufficient to transform s into t.

Levenshtein distance is named after the Russian scientist Vladimir Levenshtein, who devised the algorithm in 1965. The Levenshtein distance algorithm has been used in:

- Spell checking
- Speech recognition
- DNA analysis
- Plagiarism detection

4. Implementation

In Figure 1, it shows the overall system design for spelling checking. In spelling checking, the system breaks down the input sentences into word by word and then the system matches the words with the lexicon that has stored words. If the word is not found it is considered to be an error, and an attempt may be made to suggest a word that was likely to have been intended. One such suggestion algorithm is to list those words in the dictionary having a small Levenshtein distance from the original word.

To apply the system database, the administrator must enter the password. If the password is correct, then the administrator can repair the system database such as adding to database, updating, inserting and deleting to the database. If the password is incorrect, the administrator must reenter the password again.

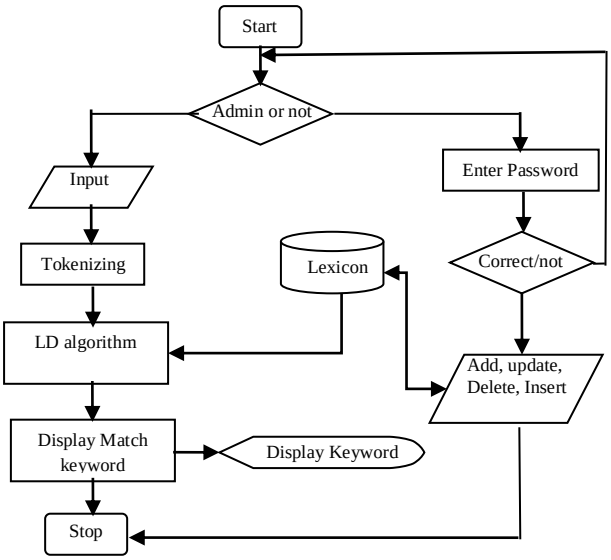

Figure 1. Overall System Design

4.1 The Algorithm

Steps

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], containing 0..m elements.
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, 4, 5, 6) are complete, the distance is found in the cell v1[m].

The Example

This section shows how the Levenshtein Distance is computed when the source is “GUMBO” and the target string is “GAMBOL”.

Steps 1 and 2

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

Steps 3 to 6 When i=1

		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**				

Steps 3 to 6 When i= 2

		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			

Steps 3 to 6 When i= 3

		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		

Steps 3 to 6 When i= 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	

Steps 3 to 6 When i= 5

		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

Steps 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).

4.2 Sequence Diagram of the System

There are four class components in this system. Each component is responsible to provide the functions of spell checking procedures. Firstly, each user inputs a sentence into the system. The sentence is split and also checked to prove the word validation. The last component Suggestion sends the result back to the user depending on the spell correction. A suggestion list is additionally sent while the wrong spell is recognized by the system. The change word () method also works on changing the correct word defined by the user.

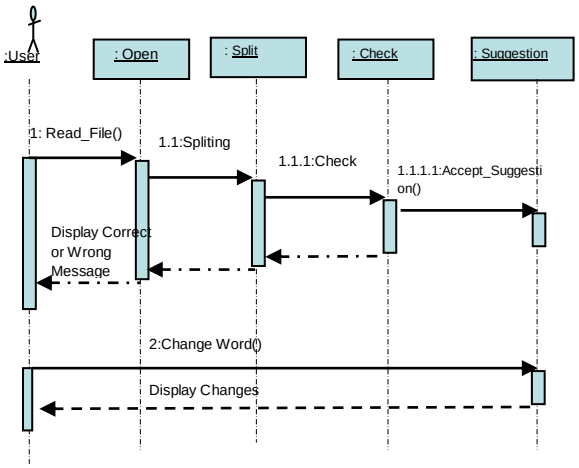

Figure 2. Sequence Diagram of the System

4.3 Spell checking with suggestion

In the Figure 4, the user must enter the sentence in to the working area. And then, the user can check the spelling. When the user checks the spelling, the system matches the input words with Lexicon. If the spelling error is found, the system will display the error words to the user.

The user chooses the error word and then the user can used the suggestion button. The system is used to give the possible suggestion list by using the Levenshtein Distance algorithm. This suggested list allows the user to choose the correct word. The user can use Correction button to change with correct word.

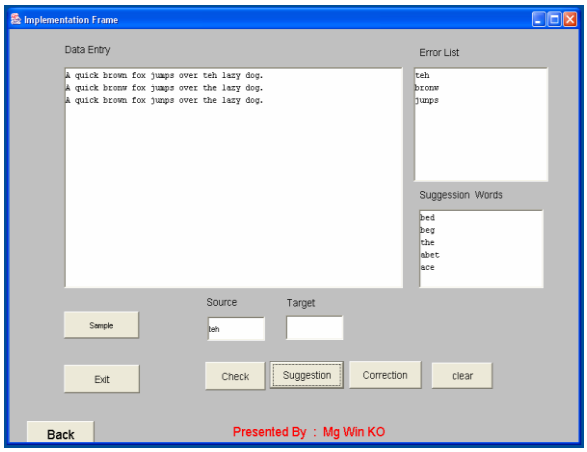

Figure 4. Error Checking and Suggestion

5. Conclusion

This system implement 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 English languages. In this system, it allows the simple sentence. It is just only for spelling checking with possible suggestion. In this system, the lexicon contains nearly about 3000 words. So, the system cannot check more than 3000 words. The system can not 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] Efarim Turban, Expert System and Applied Artifical Intelligence, ISBN 0-02-94b5b5-b, Macmillan Publishing Company, New York, 1992

[2] H.Mishkoof, "Understanding Artificial Intelligence", The Staff of the Texas Instrument Information Publishing Center, First Edition ISBN:0-672-27021-8

[3} N.Daniel, "A Rule-Based Style and Grammer Checker", 28 August 2003

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

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

[6]http://en.wikipedia.org/wiki/Spelling-checker

[7]http://www.03NLP.LIS.Encyclopedia.htm

[8]http://en.wikipedia.org/wiki/Levenshtein_distance