International Journal of
Advances in Scientific Research and Engineering (ijasre)

E-ISSN : 2454-8006

DOI: 10.31695/IJASRE.2020.33780

Volume 6, Issue 4
April - 2020

Similarity Based Information Retrieval Using Levenshtein Distance Algorithm

Daw Khin Po

Lecturer

[1]Faculty of Computer Science

University of Computer Studies (Mandalay)

Mandalay Division, Myanmar

ABSTRACT

Sentence similarity plays an important role in many text-related research and applications such as information retrieval, information recommendation, natural language processing, machine translation and translation memory, and etc. Calculating similarity between sentences is the basis of measuring the similarity between texts which is the key to document classification and clustering. Sentence similarity partially depends on the word similarity. This system will display a similar text of field, areas and other facts in document retrieval. This paper uses a sentence matching method of the Levenshtein Distance algorithm. The similarity between words can be calculated from the spelling of words or the meaning of words. Sentence similarity: The similarities between words in different sentences have a great influence on the similarity between two sentences. This system is retrieved from a similar sentence that included Theories, Methods and other facts in the document database.

Key Words: *Sentence Similarity, Levenshtein Distance, Tokenization, Stop words, Information Retrieval.*

1. INTRODUCTION

String comparison is a central operation in various environments: a spelling error correction program tries to find the dictionary entry which resembles most a given word, in molecular biology. An obvious measure for the closeness of two strings is to find the maximum number of identical symbols in them (preserving the symbol order). When compared to general text, however, personal names have different characteristics that need to be considered.

From the technical point of view, the system wants to link and match as many words as possible with the correct individuals. Words are also important pieces of information when databases are deduplicated and when data sets are linked [2] or integrated and no unique entity identifiers are available. While there is only one correct spelling for many words, there are often several valid variations for specific sentences. The objectives are: to extract similar methods and algorithms in preceding papers, to provide word order similarity between sentences using Levenshtein distance algorithm and to study the sentence matching method of Levenshtein Distance algorithm. The main contributions of this system are the proposal of a similarity measure between sequences of conference papers which searches for matches within sentences.

The paper is organized into six sections. Section 1 include introduction of the system, the main objectives of about the system and motivation of the system. Section 2 describes theoretical background that includes: Pattern matching methods and Levenshtein Distance algorithm. Section 3 discusses design of the system and conferences papers database and explains step by step process for string matching. Also section 4 includes the Implementation results of the system. Finally, Section 5 presented the main conclusions and further extension items.

2. PATTERN MATCHING

Pattern matching techniques are commonly used in approximate string matching, which has widespread applications, from data linkage and duplicate detection, information retrieval, correction of spelling errors [3], to bio and health informatics [5][7].

- Levenshtein or Edit distance
- Damerau-Levenshtein distance
- Bag distance
- Smith-Waterman distance
- Longest common sub-string (LCS)
- Q-grams
- Positional q-grams
- Skip-grams
- Compression
- Jaro algorithm
- Winkler (or Jaro-Winkler) algorithm

2.1 The Concept of Levenshtein Distance

Relying on the works of Damerau, Levenshtein [8] considered three editing operations (insertion, deletion, permutation), and defined his method as edit distance that compares two words while calculating the number of editing operations subjected on a word to turn it into another. This distance is also called Levenshtein distance. The algorithm below simultaneously aligns reference and hypothesis strings and computes the overall word error rate. Partial alignment errors are stored in the matrix *R*. Matrix *B* allows you to backtrack an alignment between strings. An element in *B* is either "up", "left", or "up-left". When backtracking from $B[n,m]$, at point $B[i, j]$, "up" moves you to $B[i-1, j]$, "left" moves you to $B[i, j-1]$ and "up-left" moves you to $B[i-1, j-1]$. The number of insertion errors equals the number of "left"s on this path, the number of deletion errors equals the number of "up"s, and the substitution errors equals the number of "up-left"s in which the aligned words don't match. Levenshtein Distance (LD) is a measure of the similarity between two strings, which we will refer 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. 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.

The greater the Levenshtein Distance, the more different the strings are [4][8]. The Levenshtein Distance algorithm has been used in: Spell checking, Speech recognition, DNA analysis and Plagiarism detection.

2.2. Levenshtein Distance Algorithm

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 a matrix containing 0..m rows and 0..n columns.
2	Initialize the first row to 0..n. Initialize the first column 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] doesn't equal t[j], the cost is 1.
6	Set cell d[i,j] of the matrix equal to the minimum of: a. The cell immediately above plus 1: d[i-1,j] + 1. b. The cell immediately to the left plus 1: d [i,j-1] + 1. c. The cell diagonally above and to the left plus the cost: d [i-1,j-1] + cost.
7	After the iteration steps (3, 4, 5, 6) are complete, the distance is found in cell d[n,m].

2.3. Operation of Levenshtein Distance Algorithm

The Levenshtein algorithm (also called Edit-Distance) calculates the least number of edit operations that are necessary to modify one string to obtain another string. A matrix is initialized measuring in the (m,n)-cell the Levenshtein Distance between the m-character prefix of one with the n-prefix of the other word. The matrix can be filled from the upper left to the lower right

DOI: 10.31695/IJASRE.2020.33780

corner. Each jump horizontally or vertically corresponds to an insert or a delete, respectively. The cost is normally set to 1 for each of the operations. The diagonal jump can cost either one, if the two characters in the row and column do not match or 0, if they do. Each cell always minimizes the cost locally. This way the number in the lower right corner is the Levenshtein Distance between both words. There are two possible paths through the matrix that actually produce the least cost solution. Namely"=" Match; "o" Substitution; "+" Insertion; "-" Deletion [8].

3. STRING MATCHING SYSTEM

The editor (with property rights on the conference) should make the basic decision of whether a paper is worth publishing or not. So this system can provide for the new paper publication with similar preceding papers. Editor or user input the title of paper. This system will be tokenized process [6]. Using these token, this system will search similar title in preceding papers database. The outputs are similar title, author name, conference name, and published date with similarity value.

Using Levenshtein Distance, firstly accept input as user desired title in preceding conference papers. This input as sentence, so this system tokenized this sentence and remove the insignificant words [1] using database. This token words uses as keyword and searches similar word of title in database. The results are many titles that are similar word (theory, method, algorithm, application, approach, and techniques). So the system will retrieve these titles with similarity order. This result can support for redundant title for students and supervisors. The system will give us the string for each input sentences. We use these strings to compare based on the distance algorithm. The user will be able to choose from a list of resulting possible strings according to their respective meaning. By using this algorithm, the correct and same strings will be known and the amount of variations and needed character also will be understood. Finally, this system will give to the user for the Field of Thesis, methods, approach, application, and their applied Theories. We use a String database of more than 300 title strings which contains not only the spelling, but also the same words or similar words. Then the user can be seen the previous titles from the resulting list of database.

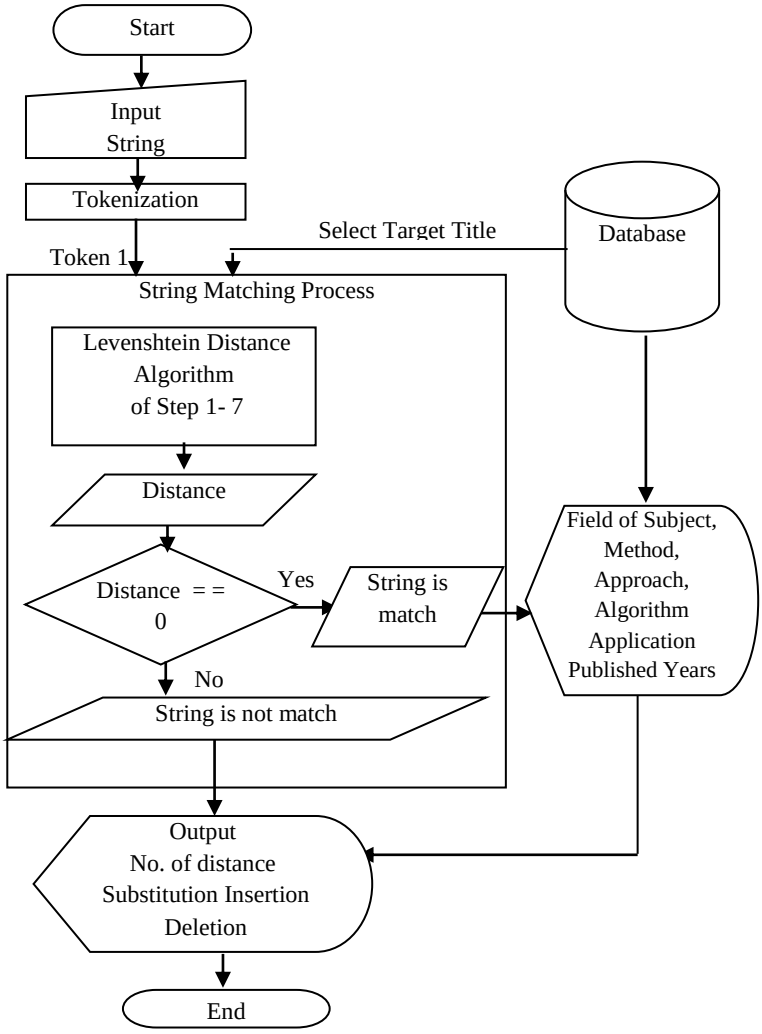


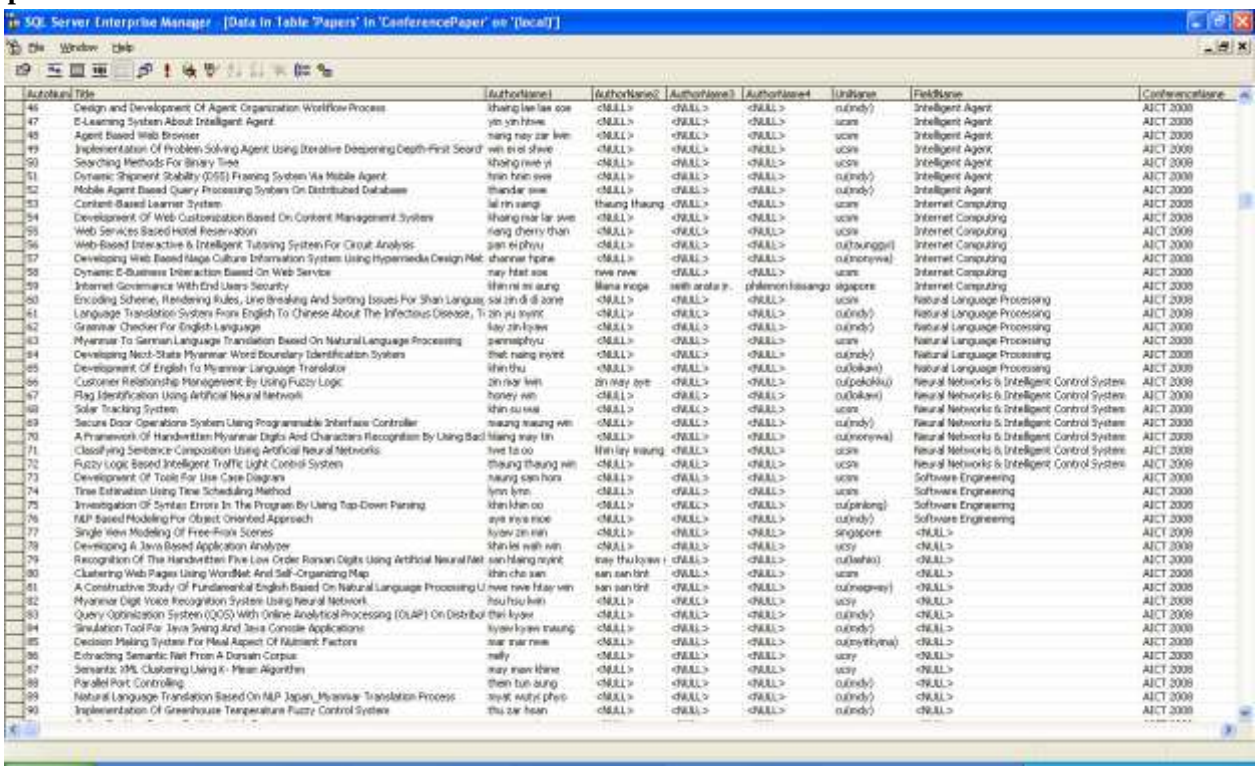
Figure1. System Flow Diagram

DOI: 10.31695/IJASRE.2020.33780

4. IMPLEMENTATION RESULT

The paper proposed string variations for different paper titles to guide the implementation of string matching system, currently worked out for Conference papers. This paper will be understood the characteristics of Levenshtein Distance algorithm which help to find reasonable variants of string. Experimental results on different paper titles data sets have shown that there is no single best technique available. The characteristics of the titles data to be matched have to be considered when selecting a matching technique. The paper provides the output result for many String variations using several real world data sets containing paper titles. This system provides a comprehensive review of existing distance literature with particular emphasis on data representation. The second and main objective will be to design and implement a comprehensive similarity and text distance measure incorporating new algorithm. This system uses LD to obtain the similarity of the query and the title of the documents to find more similar documents in the retrieved documents. The main evaluation of the system will be based for the most part on its correctness on strings.

4.1 Sample Database



5. CONCLUSION

The paper discusses the characteristics the potential sources of variations and errors in string, and present an overview of pattern matching based on string matching techniques. Measures of text similarity have been used for a long time in applications in natural language processing and related areas. This system considers for finding two strings S1 and S2 such that common string is a same solution LD. An effective method to compute the similarity between texts or sentences has many applications in natural language processing and related areas such as information retrieval and text filtering. The name matching techniques covered by this investigation comprise only a small selection of those existing, but they are the representative of many of the current approaches to the problem of string matching. This paper develops the problems involved in approximate string matching in general and in string matching specifically. The paper work suggests that methods based on distance measures are the best in these situations for the obvious reason.

The advantage of this process would be an improvement of searching algorithms for paper titles in databases as well as in the internet. Here the system will need string matching algorithms. A further benefit of this process would be an optimization for string searching. The next step of development is to take into account different cultures. The benefits in number of matches being able to find all, or almost all, of the possible matches contained in the database, were with the current database and set of search words. This system can extend the performance of information retrieval. To try to give a more qualitative view of the results compared to the results with other methods.

DOI: 10.31695/IJASRE.2020.33780

REFERENCES

1. A. Islam and D. Inkpen. "Semantic text similarity using corpus-based word similarity and string similarity". ACM Transactions on Knowledge Discovery from Data (TKDD), 2(2):1–25, 2008.
2. Bouchard, G. and Pouyez, C. (1980). Name Variations And Computerised Record Linkage. Historical Methods, 13(2), 119-125.
3. Domeij, Rickard, Hollman, Joachim, and Kann, Viggo. Detection of Spelling Errors in Swedish Not Using a Word List en Clair, Journal of Quantitative Linguistics, Vol 1(3), 1994.
4. Du, M. W., and Chang, S. C. A model and a fast algorithm for multiple errors spelling correction, Acta Informatica, Springer-Verlag, Vol 29, 1992, pp 281-302.
5. G. Navarro. 2001. A Guided Tour to Approximate String Matching, ACM Computing Survey, Vol. 33, No. 1, pp. 31-88.
6. Jonathan A. Zdziarski, Ending Spam, No Starch Press, Copyright © 2005 by "tokenization: the building blocks of spam".
7. Lin, D. (1998). An Information-Theoretic Definition of Similarity. ICML '98 Proceedings of the Fifteenth International Conference on Machine Learning. pp. 296-304. Retrieved from http://www.cs.ualberta.ca/~lindek/papers/sim.pdf
8. Levenshtein, V., (1966) Binary codes capable of correcting deletions, insertions and reversals, Soviet Physics Doklady, 10(8): 707-710.

DOI: 10.31695/IJASRE.2020.33780