Comparison of Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm for String Matching

Khin Moe Myint Aung, Ah Nge Htwe
University of Computer Studies, Yangon
khinmoemyintaung5@gmail.com, anhtwe@gmail.com

Abstract

String similarity measures play an increasingly important role in text related research and applications in tasks and operate on string sequences and character composition. A string metric is a metric that String_Based measures similarity or dissimilarity (distance) between two strings for approximate string matching or comparison. Determining similarity between texts is crucial to many applications such as clustering, duplicate removal, merging similar topics or themes, text retrieval and etc. Among many methods of String similarity, Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm are used in this proposed system. The proposed system intended to present by comparing Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm based on their f-score. So, user can search effectively the required song by typing the title of songs or artist name using English language in this proposed system. Then the proposed system retrieve the user's required song information with similarity score. The matching efficiencies of these algorithms are compared by searching f-score and execution time. The proposed system uses song title and artist feature of billboard song dataset from year 1965-2015 and implements using Java programming language.

Keywords *– Levenshtein Distance Algorithm, Needleman-Wunsch Distance Algorithm, Dataset, Similarity Score, f-score.*

1. Introduction

String searching is a very important component of many problems, including text editing, text searching and symbol manipulation. Strings searching sometimes called String matching are an important class of string algorithms that try to find a place where one or several strings (also called patterns) are found within a larger string or text. In order to search for a pattern within a string, an algorithm is needed to find the pattern as well as to know the locations where it was found in a given sequence of characters.

The proposed system analyzes the similarity measurements on Song Information by using Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm. The objective of this research is to compare the Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm based on their f-score value and execution time.

While entering characters there may be some typographical errors (typos), Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm find similar strings and displays results for the predicted strings. If the user wants to search for an artist containing keyword "oliver" but by mistake he or she types "olover" then because of Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm the system will be able to display the song containing "oliver". Similarly if user wants to search for song titles containing keyword "downtown" but by mistake he or she types "downtoun" the system will be able to display proper song containing "downtown". Since words "oliver" and "olover" are similar, similarly words "downtown" and "downtoun" are similar.

Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm are based on finding similar strings from the billboard song dataset. Levenshtein distance here refers to number of single character operations such as insertion, replacement or deletion need to be done in order to transform one string to another string. For example, edit distance between "bein" and "pin" is two, since replacing character 'b' by 'p', deleting character 'e' then word "bein" can be converted to "pin".

The Needleman-Wunsch Distance Algorithm performs a global alignment to find the best match or alignment of two strings through computing minimal alignment distance. For example minimal alignment distance between "bein" and "pin" is three, since aligning character 'b' by 'p'(mismatch) , aligning character 'e' by '-'(character 'e' align gap cost),

aligning character ‘i’ by ‘i’, aligning character ‘n’ by ‘n’ then word “bein” can be converted to “pin”. Here, gap penalty=2, match=0 and mismatch=1.

2. Related Works

SinglaN [3] et al was exploiting different kinds of string matching algorithms for strings and searching the best algorithm in some application. They were decreed their preprocessing and orders that evaluate the matching.

Pandiselvam.P, Marimuthu.T and Lawrance.R [2] was evaluated different kinds of string matching algorithms for biological sequences such as DNA and Proteins and observed their time and space complexities.

New Zin Oo [10] was proposed the process of checking the spelling of a Myanmar input word and suggestion list if it is missed spelt Myanmar word. This is intended to develop a Myanmar Language Spell Checker (or spell check) by using Levenshtein Distance Algorithm, Dynamic Threshold Algorithm and Transformation Algorithm.

Khaing Su Yee [11] was analyzed the DNA and protein structure of HIV genome structure by using Levenshtein Distance Algorithm and determined what kind of behaviour that the sequence has.

3. Background Theory

String similarity measures play an increasingly important role in text related research and applications in tasks such as information retrieval, text classification, document clustering, topic detection, topic tracking, questions generation, question answering, essay scoring, short answer scoring, machine translation, text summarization and others. String matching algorithms are used to find the matches between the source string and the target string.

3.1. Levenshtein Distance Algorithm

Levenshtein distance (LD) is a measure of the similarity between two strings, 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 [8]. The greater the Levenshtein distance, the more different the strings are [8].

It is fast and best suited for strings similarity. It is not restricted by the strings needing to have the same length. It is not considered order of sequence of characters while comparing.

Levenshtein Distance Algorithm

Step 1: Initialization
a) Set n to be the length of s, set m to be the length of t.
b) Construct a matrix containing 0..m rows and 0..n columns.
c) Initialize the first row to 0...n,
d) Initialize the first column to 0...m.
Step2: Processing
a) Examine s (i from 1 to n).
b) Examine t (j from 1 to m).
c) If s[i] equals t[j], the cost is 0.
d) If s[i] doesn't equal t[j], the cost is 1.
e) Set cell d[i,j] of the matrix equal to the minimum of:
i) The cell immediately above plus 1: d[i-1,j] + 1.
ii). The cell immediately to the left plus 1: d [i, j-1] + 1.
iii The cell diagonally above and to the left plus the cost: d [i-1, j-1] + cost.
Step 3: Result
Step 2 is repeated till the d [n, m] value is found.

In the following Table 1 example, finding Levenshtein Distance between “helo” and “hello”.

Table 1. Example of Levenshtein Distance Algorithm

		h	e	l	l	o
	0	1	2	3	4	5
h	1	0	1	2	3	4
e	2	1	0	1	2	3
l	3	2	1	0	1	2
o	4	3	2	1	1	**1**

The distance is in the lower right hand corner of the matrix, i.e., 1.

3.2. Needleman-Wunsch Distance Algorithm

The Needleman-Wunsch algorithm finds the optimal alignment of two strings (the source string (s) and the target string (t)). It is also referred as optimal matching algorithm and the global alignment technique [12]. Needleman-Wunsch distance algorithm is allowed to insert gaps (a blank character) in either or both of the strings in such a way to make

the resulting strings an optimal match. The match is optimal, the score is minimum.

It is best for string comparison because it considers ordering of sequence of characters. It finds the optimal alignment solution between the sequences. It is approximate for finding the best alignment of two strings that are similar in length and similar across their entire length. It takes more time to make the alignment this decrease the performance.

Needleman-Wunsch Distance Algorithm

Set n to be the length of s, set m to be the length of t. If n=0, return respective m and exit. If m=0, return n and exit.
If Construct a matrix containing 0...n rows and 0...m columns.
Initialization
F (0, 0) = 0
F (0, i) = i * d (d is the gap Penalty)
F (j, 0) = j * d
Main Iteration
For each i = 1 . . . M (M is the length of target string)
For each j = 1 . . . N (N is the length of source string)
F (i, j) = min{ F(i − 1, j − 1) + s(xi, yj), case1
F(i − 1, j) + d , case2
F(i, j − 1) + d , case3 }
Ptr (i, j) = {DIAG, if case 1
LEFT, if case 2
UP, if case 3}
The distance is found in cell F [n, m].
Termination

In the following Table 2 example, finding Needleman-Wunsch Distance between "helo" and "hello" with match = 0, mismatch = 1 and gap penalty= 2.

Table 2. Example of Needleman-Wunsch Distance Algorithm

		h	e	l	l	o
	0	2	4	6	8	10
h	2	0	2	4	6	8
e	4	2	0	2	4	6
l	6	4	2	0	2	4
o	8	6	4	2	1	**2**

The alignment distance is in the lower right hand corner of the matrix, i.e., 2.

3.3. Similarity Score

Similarity score is the measure to show how similar two set of data are to each other [13]. In this case, the two set of data are user input and the data in the dataset. It is not only for two texts but also for the different algorithm options given to the user in the application. The method to calculate the similarity can be found by:

$Similarity(source, target)$ = (Distance(source,target))/(Maximumlength(source,target))*100% (1)

For the similarity measures a threshold will be 50 that influences the classification performance (name pairs with a similarity value equal and above the threshold are show to the user and pairs with similarity value below are not show).

3.4. F-Score

To compare the system performance for results of two methods, *f-score* is measured as evaluation method. The proposed system is measured using the *f-measure* (also called *f-score*) which is based on precision and recall. Precision and Recall for each system is calculated using the following formula.

$$Precision = TP/(TP+FP) * 100\% \quad (2)$$

$$Recall = TP/(TP+FN) * 100\% \quad (3)$$

where, **TP** being the true positives (known matched name pairs classified as matches), **TN** the true negatives (known un-matched name pairs classified as non-matches), **FP** the false positives (unmatched name pairs classified as matches) and **FN** the false negatives (known matched name pairs classified as nonmatches). F-score is calculated using the following formula.

$$f = 2*(precision * recall)/(precision + recall) \quad (4)$$

4. Implementation and Experimental Result

From this system, Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm are compared by using f-score and execution time.

In Figure 1, the proposed system showed f-score value for Artist and Input 1 is “elvis Presley”, Input 2 is “sammy johns”, Input 3 is “major harris”, Input 4 is “sweet”, Input 5 is “the doobie brothers”, Input 6 is “sam hunt”, Input 7 is “sia”, Input 8 is “john mayer”, Input 9 is “new vaudeville band” and Input 10 is “debelah morgan”. Then it could be seen that Levenshtein Distance Algorithm has better accuracy than the Needleman-Wunsch Distance Algorithm at the two strings (“major harris” and “new vaudeville band” input string) because it has more relevant information. And also seen that Needleman-Wunsch Distance Algorithm has better accuracy than the Levenshtein Distance Algorithm at the three strings (“the doobie brothers”, “sam hunt” and “john mayer” input string). Some string is seen that it’s have the same accuracy. So, it algorithms depend upon the input may be equal or more.

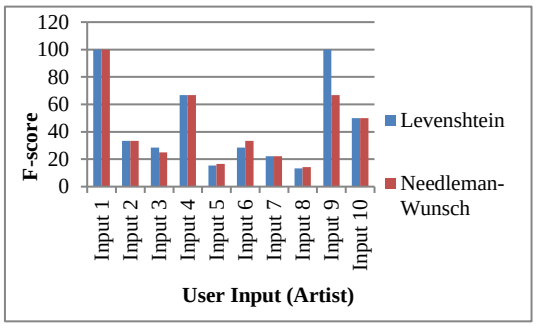

Figure 1. Comparison by using *F*- score graph for Artist Data

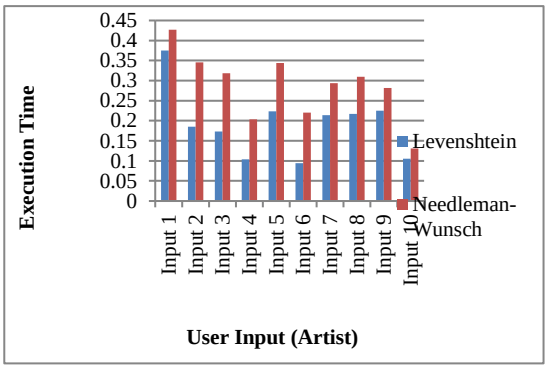

Figure 2. Comparison by using Time graph for Artist

The execution time for the proposed system is measured by using seconds. In Figure 2, the proposed system showed execution time for Artist. Then it could be seen that Needleman-Wunsch Distance Algorithm has a long execution time than the Levenshtein Distance Algorithm tested with ten input data. So, a lot of data for Needleman-Wunsch Distance Algorithm has a long execution time than the Levenshtein Distance Algorithm.

In Figure 3, the proposed system showed f-score for Song Title (using wrong spelling data) and Input 1 is “daedi”, Input 2 is “fly like an bird”, Input 3 is “nice to e with you”, Input 4 is “fox on the rood”, Input 5 is “what dreamss of the brokenhearted”, Input 6 is “19th”, Input 7 is “were an singabore band”, Input 8 is “crocodile rock wish”, Input 9 is “day by week” and Input 10 is “ovar yeu”. Then it could be seen that Levenshtein Distance Algorithm has better accuracy than the Needleman-Wunsch Distance Algorithm at the two strings (“fox on the rood” and “what dreams of the brokenhearted” input string) because it has more relevant information. Some string is seen that it’s have the same accuracy. So, it algorithms depend upon the input may be equal or more.

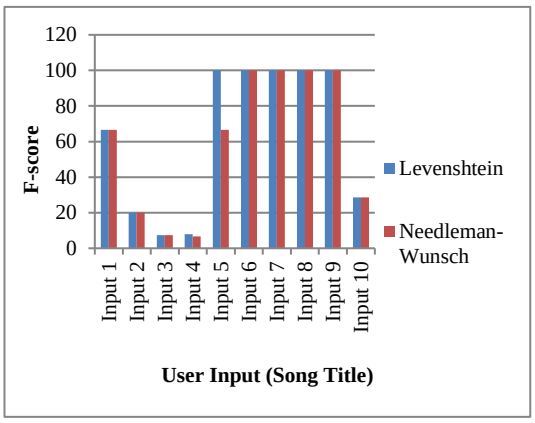

Figure 3. Comparison by using *F*- score graph for Song Title Data

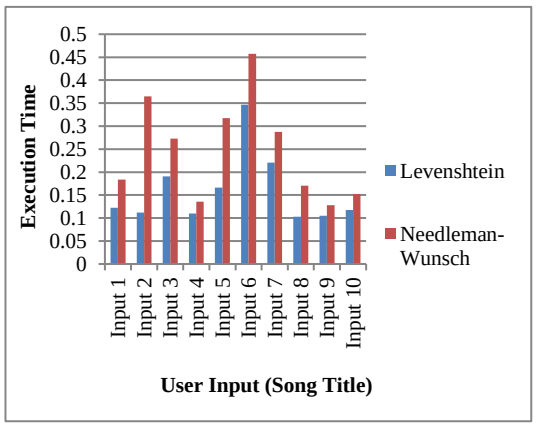

Figure 4. Comparison by using Time graph for Song Title

In Figure 4, the proposed system showed execution time for Song Title. Then it could be seen that Needleman-Wunsch Distance Algorithm has a long execution time than the Levenshtein Distance Algorithm tested with ten input data. Here, the proposed system used wrong spelling for input data. So, a lot of data for Needleman-Wunsch Distance Algorithm has a long execution time than the Levenshtein Distance Algorithm.

5. Conclusion

String matching algorithm plays the vital role in String Computation. The time complexity of Levenshtein Distance Algorithm is O (N+M) and the time complexity of Needleman-Wunsch Distance Algorithm is O (NM). The proposed system presents comparison of Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm for song information based on their *f*-score and execution time. It can search not only the song title but also search artist name. This system is tested with many input data (about 500 input) for song information. And it is also tested with average f-score value. By looking at the experimental results, it could be seen that the f-score value of Levenshtein distance algorithm and the Needleman-Wunsch distance algorithm depend upon the input may be equal or more for single input measurement. But at the average f-score, Levenshtein Distance Algorithm has better accuracy than the Needleman-Wunsch Distance Algorithm. So, it algorithms may be equal or more for the single input measurement but at the average f-score, Levenshtein Distance Algorithm has better accuracy than the Needleman-Wunsch Distance Algorithm.

And as a time complexity of its algorithm, Needleman-Wunsch Distance Algorithm has more complexity than the Levenshtein Distance Algorithm. For execution time, although seeing some data for Levenshtein Distance Algorithm also has a long time that a lot of data for Needleman-Wunsch Distance Algorithm has a long execution time than the Levenshtein Distance Algorithm.

References

[1] Ratmalana and Sri Lanka, "A Comparative Analysis of Various String Matching Algorithms", DU Vidanagama Department of Information Technology, Faculty of Computing, General Sir John Kotelawala Defence University, Proceedings of 8th International Research Conference, KDU.

[2] Pandiselvam.P, Marimuthu.T and Lawrance. R, "AComparative Study On String Matching Algorithms Of Biological Sequences", Department of Computer Applications, AyyaNadarJanakiAmmal College, Sivakasi, India, Jan 29, 2014, Selected for International Conference on Intelligent Computing, Cornell University Library.

[3] NimishaSingla and Deepak Garg, "String Matching Algorithms and their Applicability in various Applications", Department of Computer Science, Thapar University, Ludhiana, India, International Journal of Soft Computing and Engineering (IJSCE)ISSN: 2231-2307, Volume-I, Issue-6.

[4] Maria del Pilar Angeles and Adrian Espino-Gamez, "Comparison of methods Hamming Distance, Jaro, and Monge-Elkan", Facultad de Ingenieria Universidad NacionalAutonoma de Mexico Mexico, D.F, The Seventh International Conference on Advances in Databases, Knowledge, and Data Applications.

[5] Koloud Al-Khamaiseh*, ShadiALShagarin**, "A Surveryof String Matching Algorithms",Koloud A1 Khamaiseh Int. Journal of Engineering Research and Applications ISSN : 2248-9622, Vol 4, Issue 7(Version 2), July 2014, pp.144-156.

[6] Wael H. Gomaa, Aly A. Fahmy, "A Survey of TextSimilarity Approaches", International Journal of Computer Applications (0975 – 8887) Volume 68– No.13, April 2013.

[7] Levenshtein distance- Wikipedia, the free ency... https://en.wikipedia.org/wiki/Levenshtein_distance

[8] Rishin Haldar and Debajyoti Mukhopadhyay, "Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach", Web Intelligence & Distributed Computing Research Lab Green Tower, India.

[9] Güyer, Atasoy, and Somyürek, "Measuring Disorientation Based on the Needleman-Wunsch Algorithm", Gazi University, Turkey, International Review of Research in Open and Distributed Learning Volume 16, Number 2.

[10] NweZinOo, "Myanmar Words Spelling Checking Using Levenshtein Distance Algorithm", M.C.Sc, 2010, University of Computer Studies, Yangon.

[11] Khaing Su Yee, "Detecting the Behaviours of HIV DNA Sequences Using Levenshtein Distance Algorithm", M.C.Sc, 2007, University of Computer Studies, Yangon.

[12] Needleman-Wunsch algorithm-Wikipedia, the free ency..https://en.wikipedia.org/wiki/Needleman Wunsch_algorithm.

[13] Abdulla Ali, "Textual Similarity", Technical University of Denmark Informatics and Mathematical Modelling Building 321, DK-2800 KongensLyngby, Denmark, IMM-BSc: ISSN 2011-19.

[14] F1 score-, Wikipedia, the free ency... https://en.wikipedia.org/wiki/F1_score.