COMPARISON OF LEVENSHTEIN DISTANCE ALGORITHM AND NEEDLEMAN-WUNSCH DISTANCE ALGORITHM FOR STRING MATCHING

KHIN MOE MYINT AUNG

M.C.Sc. **JANUARY2019**

COMPARISON OF LEVENSHTEIN DISTANCE ALGORITHM AND NEEDLEMAN-WUNSCH DISTANCE ALGORITHM FOR STRING MATCHING

BY

KHIN MOE MYINT AUNG
B.C.Sc. (Hons:)

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Computer Science
(M.C.Sc.)

University of Computer Studies, Yangon

JANUARY2019

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere thanks to those who helped me with various aspects of conducting research and writing this thesis.

Firstly, I would like to express my gratitude and my sincere thanks to **Prof. Dr. Mie Mie Thet Thwin**, Rector of the University of Computer Studies, Yangon, for allowing me to develop this thesis.

Secondly, my heartfelt thanks and respect go to **Dr. Thi Thi Soe Nyunt**, Professor and Head of Faculty of Computer Science, University of Computer Studies, Yangon, for her invaluable guidance and administrative support, throughout the development of the thesis.

I am deeply thankful to my supervisor, **Dr. Ah Nge Htwe**, Associate Professor, Faculty of Computer Science, University of Computer Studies, Yangon, for her invaluable guidance, encouragement, superior suggestion and supervision on the accomplishment of this thesis.

I would like to thank **Daw Ni Ni San,** Lecturer, Language Department of the University of Computer Studies, Yangon, for editing my thesis from the language point of view.

I thank all my teachers who taught and helped me during the period of studies in the University of Computer Studies, Yangon.

Finally, I am grateful for all my teachers who have kindly taught me and my friends from the University of Computer Studies, Yangon, for their advice, their co-operation and help for the completion of thesis.

STATEMENT OF ORIGINALITY

I hereby certify that the work embodies in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

---------- ----------

Date Khin Moe Myint Aung

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.

String matching algorithms are used to find the similar characters between the source string and the target string. The proposed system is intended to match song information by comparing Levenshtein Distance Algorithm and Needleman-Wunch Distance Algorithm based on their f-score and execution time. So, user can search effectively their required song information by the title of songs or artist name using English language. Then the proposed system retrieves the user's required song information with similarity score. The matching efficiencies of these algorithms are compared by searching f-score and the 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.

CONTENTS

LIST OF FIGURES

LIST OF EQUATIONS

CHAPTER 1

INTRODUCTION

String searching is a very important component of many problems, including text editing, text searching and symbol manipulation. String searching sometimes called String matching is 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 found in a given sequence of characters.

Various string matching algorithms are used to solve the string matching problems like wide window pattern matching, approximate string matching, polymorphic string matching, string matching with minimum mismatches, prefix matching, suffix matching, and similarity measure [8]. 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.

Levenshtein distance (LD) and Needleman-Wunsch distance are a measure of the similarity between two strings, the source string(s) and the target string (t). The source string is user input and the target string is one of the entries in the dataset.

When the user enters characters, there may be some typographical errors (typos), Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm find similar keywords and displays results for the predicted keywords. The user wants to search for an artist containing keyword “oliver” but he or she types spelling error “olover”. According to Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm the system will be able to display the song containing “oliver”. Similarly the user wants to search for song titles containing keyword “downtown” but he or she types spelling error “downtoun”. According to these algorithms 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 word to another word. 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”.

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”. In this system, gap penalty=2, match=0 and mismatch=1.

The proposed system is to compare the Levenshtein distance algorithm and Needleman-Wunsch distance algorithm based on their f-score and execution time. And it used song and artist feature of this Billboard songs (50 Years of Pop Music Lyrics) dataset (1965-2015). It downloaded from https:// www.kaggle.com /rakannimer / billboard-lyrics and can see sample data at APPENDIX.

1.1Motivation

Nowadays, Radio Stations are very popular in Online.The ever growing amount of music available on the internet calls as intelligent tools for browsing and searching music databases. Discovery and matching of title, song names or artist names is used in an increasing number of applications and it constitutes a central part of many applications, for example, in text or web mining, information retrieval, data cleansing, or automatic spell checking and search engines. If only exact matching was available in these types of applications it would not be possible to deal with name variations, which unavoidably occur in the data and names in real world data sets. In order to get more accurate results, an approximate string matching should be applied instead of exact matching. Although string matching is used in many applications, in this thesis the main reason and motivation of providing a string matching algorithm for English language, is the increasing demand of name matching from the billboard

song dataset. The purpose of the proposed system is to compare the execution time and the accuracy of the Levenshtein distance algorithm and Needleman-Wunsch distance algorithm based on the source string and the target string.

1.2 Objectives of the Thesis

The objectives of the thesis are as follows:

- To help song information system in searching required song information using the song title or artist name.
- To analyze the similarity between two strings using Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm.
- To compare Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm based on their f-score and execution time.
- To study Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm.

1.3 Overview of the Thesis

The proposed system is intended to use for comparison of Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm based on their f-score and execution time. When user types song title or artist name, the proposed system displays the search song information with similarity score that equal and above the threshold by using Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm. Then it evaluates the experimental result by comparing *f*-score and execution time.

1.4 Organization of the Thesis

The organization of the thesis is as follows:

Chapter 1 is the introductory section where the introduction to string searching, motivation, the objectives, the overview and the organization of the thesis are presented.

Chapter (2) describes background theory approach and the general overview of string similarity and different kinds of String Matching Algorithm. And it also describes the important of Levenshtein Distance and Needleman-Wunsch Distance, similarity score, accuracy measure and related work.

Chapter (3) presents system architecture including system detail design and algorithm of Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm.

Chapter (4) implements experimental result of the system and experimental result analysis. It presents how the system is implemented by comparing Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm based on their f-score and execution time.

Chapter (5) concludes with discussion including limitations and further extensions.

CHAPTER 2

BACKGROUND THEORY AND RELATED WORK

In this chapter, background theories, and information related to the field of string similarity introduce.

2.1 Data Mining

Data mining is the task of discovering interesting patterns from large amounts of data, where the data can be stored in databases, data warehouses, or other information repositories. It is a young interdisciplinary field, drawing from areas such as database systems, data warehousing, statistics, machine learning, data visualization, information retrieval, and high-performance computing. Other contributing areas include neural networks, pattern recognition, spatial data analysis, image databases, signal processing, and many application fields, such as business, economics, and bioinformatics. Data mining process are shown in Figure 2.1.

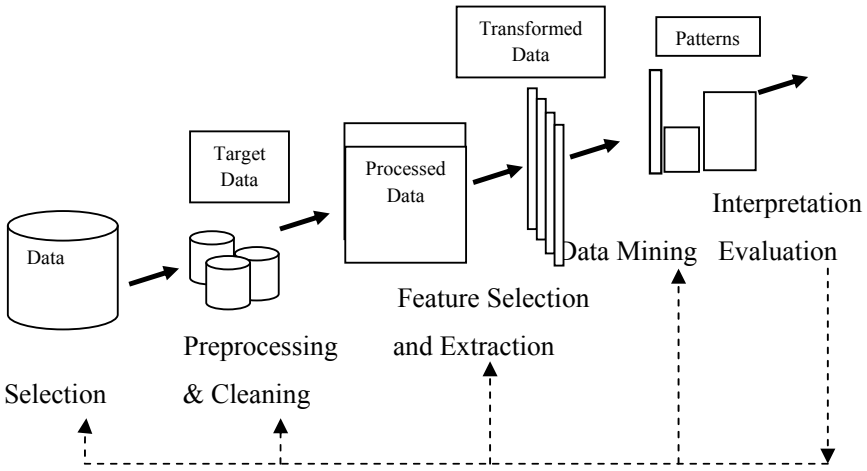

Figure 2.1 Data mining process

There are many data mining tasks. Some of the common ones are supervised learning (or classification), unsupervised learning (or clustering), association rule mining, and sequential pattern mining.

Data mining functionalities include the discovery of concept/class descriptions, associations and correlations, classification, prediction, clustering, trend analysis, outlier and deviation analysis, and similarity analysis. Characterization and discrimination are forms of data summarization.

A data mining application usually starts with an understanding of the application domain by data analysts (data miners), who then identify suitable data sources and the target data.

The whole process also called the data mining process is almost always iterative. It usually takes many rounds to achieve final satisfactory results, which are then incorporated into real-world operational tasks. Traditional data mining uses structured data stored in relational tables, spread sheets, or flat files in the tabular form. With the growth of the Web and text documents, Web mining and text mining are becoming increasingly important and popular. With the advent of the internet and the huge amount of text processing associated with information mining, string search algorithms have gained importance.

In this system, the data mining function, similarity approach is applied to build radio system for song data information by comparing Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm based on their f-score and execution time.

2.2 Overview of String Searching

In computer science, string-searching algorithms, sometimes called string-matching algorithms, 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.

A string searching algorithm aligns the pattern with the beginning of the text and keeps shifting the pattern forward until a match or the end of the text is reached.

Let Σ be an alphabet (finite set). The most basic example of string searching is where both the pattern and searched text are arrays of elements of Σ. The Σ may be a usual human alphabet (for example, the letters A through Z in the Latin alphabet). Other applications may use binary alphabet (Σ = {0, 1}) or DNA alphabet (Σ = {A, C, G, T}) in bioinformatics [4].

The object of string searching is to find the location of a specific text pattern within a larger body of text (e.g., a sentence, a paragraph, a book, etc.). As with most algorithms, the main considerations for string searching are speed and efficiency.

2.3 String Similarity

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. Finding similarity between words is a fundamental part of string similarity which is then used as a primary stage for sentence, paragraph and document similarities.

Words can be similar in two ways lexically and semantically. Words are similar lexically if they have a similar character sequence. Words are similar semantically if they have the same thing. Lexical similarity is introduced though different String-Based algorithms, Semantic similarity are introduced through Corpus-Based and Knowledge-Based algorithms.

String-Based similarity is Lexical similarity measure that operates on string sequences and character composition. Corpus-Based similarity is a semantic similarity measure that determines the similarity between words according to information gained from large corpora. Knowledge-Based similarity is a semantic similarity measure that determines the degree of similarity between words using information derived from semantic networks [11].

2.3.1 String-Based Similarity

String-Based measures operate on string sequences and character composition [11]. A string metric is a metric that measures similarity or dissimilarity (distance) between two strings for approximate string matching or comparison. String metrics are currently used in plagiarism detection, data mining, incremental search and so on. Figure 2.2 shows String-Based Similarity Measures with Character-Based and Term-

Based. Six of them are character based while the other is term-based distance measures.

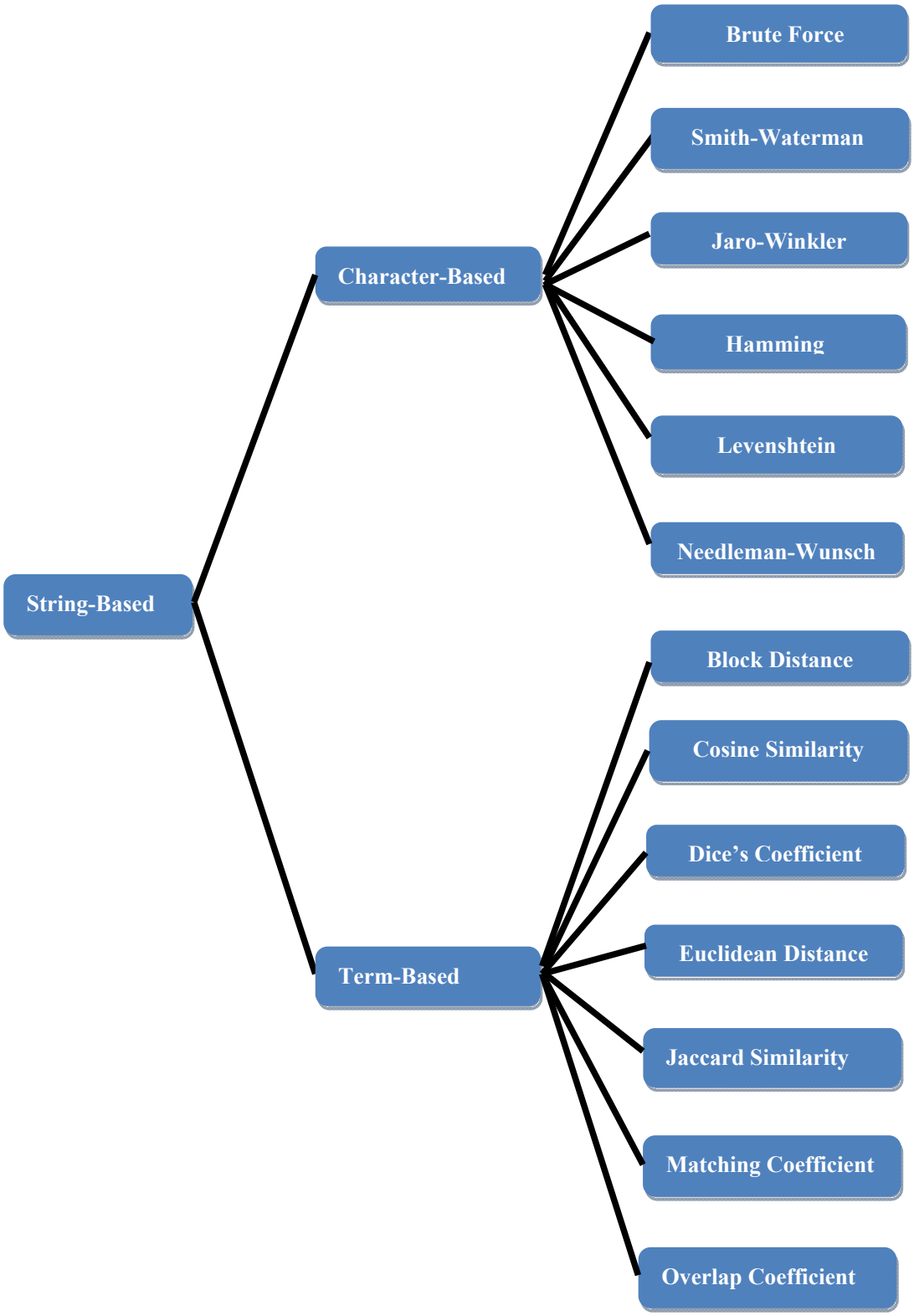

Figure 2.2 String-Based Similarity Measures

2.3.1.1 Character-Based Similarity

Brute Force, Smith-Waterman, Jaro-Winkler, Hamming, Levenshtein and Needleman-Wunsch are Character-Based Similarity Measures. Details are explained in each respective similarity.

2.3.1.2Term-Based Similarity

Block Distance is also known as Manhattan distance, boxcar distance, absolute value distance, L1 distance, city block distance and Manhattan distance. It computes the distance that would be traveled to get from one data point to the other if a grid-like path is followed. The Block distance between two items is the sum of the differences of their corresponding components.

Cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them.

Dice's coefficient is defined as twice the number of common terms in the compared strings divided by the total number of terms in both strings.

Euclidean distance or L2 distance is the square root of the sum of squared differences between corresponding elements of the two vectors.

Jaccard similarity is computed as the number of shared terms over the number of all unique terms in both strings.

Matching Coefficient is a very simple vector based approach which simply counts the number of similar terms, (dimensions), on which both vectors are none zero.

Overlap coefficient is similar to the Dice's coefficient, but considers two strings a full match if one is a subset of the other [11].

2.4 String Matching Algorithm

String matching algorithms are used to find the matches between the source string and the target string. Brute Force, Smith-Waterman, Jaro-Winkler, Hamming, Levenshtein and Needleman-Wunsch are string matching algorithms. Among them Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm are used in this proposed system.

2.4.1 Brute Force Algorithm

The simplest algorithm for string matching is a brute force algorithm. The brute force algorithm compares the pattern in the text starting from left to right, one character at a time, until a mismatch is found. This algorithm has no pre-processing phase. The algorithm can be designed to stop on either the rest occurrence of the pattern, or upon reaching the end of the text. The pattern matching starts with matching the first character of the pattern with the first character of the text. If the match doesn't find then it moves forward to the second character of the text and again compares the first character of the pattern with the second character of the text. In this case, if the match finds then moves to the second character of the pattern comparing it with the next character of the text [9]. Example for Brute Force is shown in Figure 2.3. Figure 2.4 shows Brute Force Algorithm.

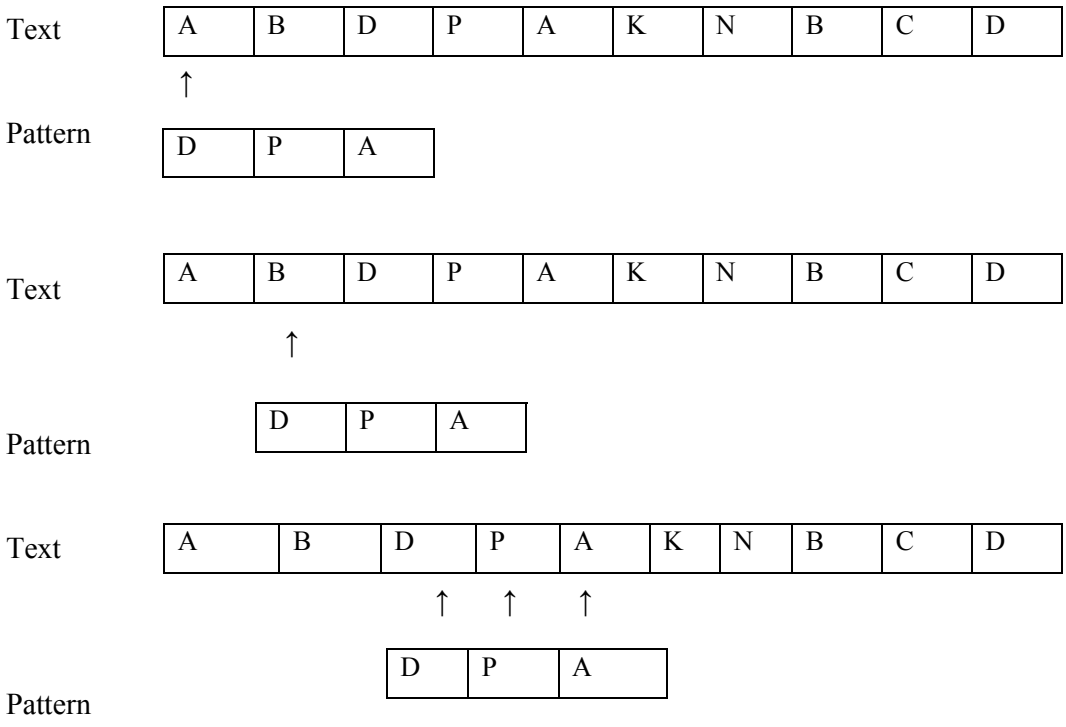

Figure 2.3 Brute Force Matching Example

```
Begin
1. Searching for a pattern, P [0…m-1], in text, T [0…n-1].
// Implements Brute-Force String Matching
2. An array T[0…n-1] of n characters representation a text and
an array P[0…m-1] of m characters representation a pattern
3. The index of the first character in the text that starts a matching substring
   or -1 if the search is unsuccessful
    for i ← 0 to n-m do
    j ← 0
    while j < m and P[j] = T[i+j] do
    j ← j+1
    if j = m return i
     return -1
End
```

Figure 2.4 Brute Force Algorithm

2.4.2 Smith-Waterman Algorithm

Smith-Waterman algorithm is a well-known algorithm for performing local sequence alignment; that is, for determining similar regions between two strings using a dynamic programming approach [6]. This is done by creating a matrix with cells indicating the cost to change from one string to other. By building on the edit distances of the strings, Smith-Waterman provides an efficient way to compare the strings. Figure 2.5 shows Smith-Waterman Algorithm.

The algorithm was first proposed by Temple F. Smith and Michael S. Waterman in 1981. Like the Needleman-Wunsch Algorithm, of which it is a variation, Smith-Waterman is a dynamic programming algorithm. As such, it has the desirable property that it is guaranteed to find the optimal local alignment with respect to the scoring system being used (which includes the substitution matrix and the gap-scoring scheme). The main difference to the Needleman-Wunsch Algorithm is that negative scoring matrix cells are set to zero, which renders the local alignments visible. Trace back procedure starts at the highest scoring matrix cell and proceeds until a cell with score zero is encountered, yielding the highest scoring local alignment.

Begin

1. Initialize edges of matrix with 0.

$M_{i,0} = 0$,

$M_{0,j} = 0$

2. Assign scores

$$M_{i,j} = \begin{cases} 0, & \\ M_{i-1,j-1} + S(A_i, B_j), & \text{Match/Mismatch} \\ \max \; M_{i-1,j} + gap, & \text{Deletion} \\ M_{i,j-1} + gap & \text{Insertion} \end{cases}$$

Where,

$M_{i,j}$= the maximum score between the two characters

gap = -2

$S(A_i, B_j) = 2$ (if $A_i = B_j$)

$S(A_i, B_j) = -3$ (if $A_i \neq B_j$)

i = length of sequence A

j = length of sequence B

3. Trace back begins at the highest score found anywhere in the matrix and continues until the score falls to 0.

End

Figure 2.5 Smith-Waterman Algorithm

2.4.3 Jaro-Winkler DistanceAlgorithm

String Distance of Jaro-Winkler Algorithm is just a metric between two elements, where the same strings have distance 0. Totally different strings have distance 1 and other values are measure of similarity of two strings between 0 and 1[7]. Figure 2.6 shows Jaro-Winkler Algorithm.

Begin

// common chars

1. Input the string x and y
   Set p equal m/2+1.
   Examine each character of x (i from 1 to i<=m).
   Examine each character of y (j from max(i-p,1) to j<=min(i+p,n)).
   If x[i] equal y[j], add one to res and break.
   Else return res.

// transpositions

2. Intialize transpositions to 0.
   Examine each character of x (i from 1 to i<=m).
   If x[i] equal y[j], add one to transpositions.
   Else set transpositions to transpositions divided by 2.

// Jaro metric

3. Set cx to common-chars (n,m).
   Set cy to common-chars (m,n).
   If cx not equal cy or cx equal 0 or cy equal 0, retrun cx/x+ cy/y+(m-transpositions(x,y))/m/3.

// pref length

4. Examine each character of x (i from 1 to i<=min(p,m)).
   If x[i] not equal y[i], retruni.
   else return min (p,m)

// Jaro_Winkler Algorithm

5. Set d to jaro metric (x,y,m,n).
   Return d + pref length (4,x,y) * 0.1* (1-d).

End

Figure 2.6 Jaro-Winkler Algorithm

2.4.4 Hamming Distance Algorithm

The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. In other words, it measures the minimum number of substitutions required to change one string into the other, or the minimum number of errors that could have transformed one string into the other. In a more general context, the Hamming distance is one of several string metrics for measuring the edit distance between two strings. By looking at some examples it becomes clear that Hamming only use substitution and nothing else. The Hamming distance between "ka**rol**in" and "ka**thr**in" is 3 and"k**a**r**ol**in" and "k**er**s**t**in" is 3. Figure 2.7 shows Hamming Distance Algorithm.

Begin

1. Input the string s1 and s2.
2. Initialize i to 0 and count to 0.
3. While s1 not equal ' ' do

   If s1[i] not equal s2[i], increase the count.

   Else return count.

End

Figure 2.7 Hamming Distance Algorithm

2.5 Levenshtein Distance Algorithm

The Levenshtein distance is a string metric for measuring the difference between two strings. The Levenshtein distance between two strings is the minimum number of single character edits (i.e. insertions, deletions or substitutions) required to change one string into the other. It is named after Vladimir Levenshtein, who discovered this algorithm in 1965. Levenshtein distance is sometimes called edit distance. It is nearly related to pair wise string alignments. The Levenshtein Distance Algorithm has been used in:

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

There are lots of applications for Levenshtein Distance Algorithm. It is used in Dialectology to estimate the proximity of dialect pronunciations [7]. Moreover, this algorithm is used in biology to find similar sequences of nucleic acids in DNA or amino acids in proteins and is checked spelling efficiently for English language and non English languages. Then Levenshtein Distance can be searched similar song information. Moreover, the Levenshtein Distance is used for automatically reduction for Medical Name Confusion, analysis for DNA sequences for HIV and checks the spelling for traveling and the name of cities in Myanmar country [7].

Levenshtein distance (LD) is a measure of the similarity between two strings, which the user will refer to as the user input and the data in the dataset. The distance is the number of deletions, insertions, or substitutions required to transform one string into other string. For example,

- If s is "hello" and t is "hello", then LD (s, t) = 0, because no transformations are needed. The strings are already identical.
- If s is "helo" and t is "hello", then LD (s, t) = 1, because one insertion (insert "l") is sufficient to transform s into t.
- If s is “game of lave” and t is “game of love”, then LD (s,t) = 1, because one substitution (change “a” to “o”) is sufficient to transform s into t.
- If s is “tam jons” and t is “tom jones”, then LD (s,t) = 2, because one substitution and one insertion (change “a” to “o” and insert “e”) is sufficient to transform s into t.
- If s is “welsonpickette” and t is “wilsonpickett”, then LD (s,t) = 2, because one substitution and one deletion (change “e” to “i” and delete “e”) is sufficient to transform s into t.

The greater the Levenshtein distance, the more different the strings are. And the Levenshtein distance between " Wenderwel " and "Wonderwall" is 3, since the following three edits change one into the other, and there is no way to do it with fewer than three edits:

1. W**e**nderwel → W**o**nderwel (substitution of "o" for "e")

2.Wonderw**e**l → Wonderw**a**l (substitution of "a" for "e")

3. Wonderwal → Wonderwal**l**(insertion of "l" at the end).

2.5.1Advantages and Disadvantagesof Levenshtein Distance

The Levenshtein distance has simple advantages and disadvantages.

Advantages

It is always at least the difference of the sizes of the two strings and at most the length of the longer string. And it is zero if and only if the strings are equal. Then it is fast and best suited for strings similarity. And it is not restricted by the strings needing to have the same length.

Disadvantages

It is not considered order of sequence of characters while comparing. Needleman-Wunsch Distance Algorithm running on two long strings results in a long time and a big cost that is proportional to the product of the two string lengths.

2.5.2 Applicationsof Levenshtein Distance

In approximate string matching, the objective is to find matches for short strings in many longer texts, in situations where a small number of differences are to be expected. The short strings could come from a dictionary, for instance. One of the strings is typically short, while the other is arbitrarily long. This has a wide range of applications; for instance, spell checkers, correction systems for optical character recognition, and software to assist natural language translation based on translation memory.

The Levenshtein distance can also be computed between two longer strings, but the cost to compute it, which is roughly proportional to the product of the two string lengths, makes this impractical. Thus, when used to aid in fuzzy string searching in applications such as record linkage, the compared strings are usually short to help improve speed of comparisons.

2.6 Needleman-Wunsch Distance Algorithm

The Needleman-Wunsch algorithm was introduced by Saul B. Needleman and Christian D. Wunsch in 1970. Its goal is to align two strings, considering the probable gaps, and using a representation matrix to represent scores gained by the comparisons of the possible studied pairs [12]. It is also sometimes referred to as the optimal matching algorithm and the global alignment technique. The Needleman-Wunsch distance algorithm has been used in:

- Computer stereo vision
- Parallelization and vectorization techniques
- DNA analysis
- Measuring Disorientation

There are lots of applications for Needleman-Wunsch Distance Algorithm. It is used in 3D reconstruction to pair stereo images. Moreover, this algorithm is used parallelization and vectorization techniques for improving the performance of the Needleman-Wunsch Distance Algorithm and used in biology to find similar sequences of nucleic acids in DNA or amino acids in proteins. Then Needleman-Wunsch Distance can be searched similar song information. Moreover, the Needleman-Wunsch Distance is used for measuring disorientation in a more precise manner.

It performs a global alignment to find the best alignment over the entire of two strings, the user input and one of the entries in the dataset. The Needleman Wunsch algorithm is an example of dynamic programming, and was one of the first applications of dynamic programming to biological sequence comparison [6].

The Needleman-Wunsch (N-W) algorithm was proposed in 1970 by Saul Needleman and Christian Wunsch. It is commonly used for general sequence alignment and scoring. Although the original purpose of this algorithm was to search for similarities between protein or nucleotide sequences in bioinformatics (Needleman & Wunsch, 1970), it is still widely used for optimal global alignment (Kozik 2011; Skopal & Bustos, 2011; Liner & Clapp, 2004; Gwizdka & Spense, 2007; Stefik, Hundhausen & Patterson, 2011) [2]. The Needleman-Wunsch algorithm uses four parameters for computing the similarity score of two strings. These parameters are:

• The first string to match

• The second string to match

• Similarity matrix (showing relations between each character of the two strings)

• A penalty gap value for the unmatched characters.(which is a value designed to reduce the score when the characters do not match.)

The lower score obtained from the Needleman-Wunsch algorithm shows the better similarity. In order to make scores comparable with each other, the scores should be normalized. For example,

- If s is "hello" and t is "hello", then NW (s, t) = 0, because the string is all match.
- If s is "helo" and t is "hello", then NW (s, t) = 2, because one gap (align “h e l – o”and “hello”) is closely related to s into t.
- If s is “game of lave” and t is “game of love”, then NW (s,t) = 1, because one mismatch (align “game of lave” and “game of love”) is closely related to s into t.
- If s is “tam jons” and t is “tom jones”, then NW (s,t) = 3, because one mismatch and one gap (align “tam jons” and “tom jones”) is closely related to s into t.
- If s is “welsonpickette” and t is “wilsonpickett”, then NW (s,t) = 3, because one mismatch and one gap (align “welsonpickette” and “Wilson pickett-”) is closely related to s into t.

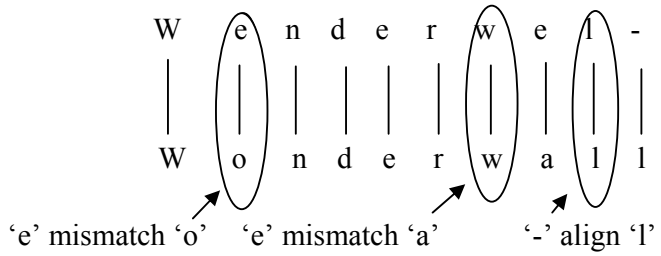

Figure 2.8 Needleman-Wunsch Distance Example

The Needleman-Wunsch distance between "Wenderwel" and "Wonderwall" is 4. Figure 2.8 shows to transform one string into the other string with match = 0, mismatch = 1 and gap penalty= 2, and there is no way to do it best.

2.6.1 Advantages and Disadvantages of Needleman-Wunsch Distance

The Needleman-Wunsch distance has simple advantage and disadvantage.

Advantages

It is one of the best for string comparison because it considers ordering of sequence of characters and it finds the optimal alignment solution between the strings. Then it is zero if and only if the strings are equal.

Disadvantages

It is approximate for finding the best alignment of two strings that are similar in length and similar across their entire lengths. It takes more time to make the alignment this decrease the performance.

2.6.2 Gap Penalty

A Gap penalty is a method of scoring alignments of two or more strings and considering the probable gaps. When aligning strings, introducing a gap in the strings can allow an alignment algorithm to match more terms. However, minimizing gaps in an alignment is important to create a useful alignment. Too many gaps can cause an alignment to become meaningless. Gap penalties are used to adjust alignment scores. Gaps can indicate a missing letter in the incorrectly spelled word. Gap penalties allow algorithms by placing gaps in original sections and matching what is identical. The gap penalty for a certain string quantifies how much of a given string is probably original. The proposed system used gap penalty is 2.

2.7 Similarity Score

Similarity score is the measure to show how similar two set of data are to each other [1]. 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.

One of them could be using the distance for the string. The maximum character length of the longest texts is the maximum distance that can be found. This means that while the distance is that maximum length then the texts are absolutely different (not similar at all) while distance on 0 means that both texts are equal [1]. The system using these points to scale the method to calculate the similarity score.

2.8 F-score

The F-measure or F-score is one of the most commonly used "single number" measures in Information Retrieval, Natural Language Processing and Machine Learning. F-score is a weighted harmonic mean of Precision and Recall, where an f-score reaches its best value at 100 and worst at 0. It considers both the Precision and Recall of the test to compute the score.

2.8.1 Precision

Precision is the fraction of retrieved documents that are relevant to the query. For a text search on a set of documents, Precision is the number of correct results divided by the number of all returned results. Precision takes all retrieved documents into account, but it can also be evaluated at a given out-off rank, considering only the topmost results returned by the system. Precision is a good measure to determine, when the costs of false positive is high. In information retrieval, a perfect Precision score of 100 means that every result retrieved by a search was relevant (but says nothing about whether all relevant documents were retrieved).

2.8.2 Recall

Recall is the fraction of the relevant documents that are successfully retrieved. For a text search on a set of documents, Recall is the number of correct results divided by the number of results that should have been returned. In information retrieval, a perfect Recall score of 100 means that all relevant documents were retrieved by the search (but says nothing about how many irrelevant documents were also retrieved).

2.9 Related Works

Many of the researchers have proposed comparative selection based on various criteria such as, complexity, cost, time, while some are focused on a String matching approach.

Singla N [6] et al was exploiting different kinds of string matching algorithms such as Boyer Moore , Boyer Moore Horspool , Brute Force , Knuth Morris Pratt, Quick Search, Rabin Karp, Approximate string matching, Needleman Wunsch, Smith Waterman. This describes the optimal algorithm for various activities that include string matching as an important aspect of functionality. In all applications test, string and pattern class needs to be matched always. Most applications use Boyer Moore, Boyer Moore Horspool or Knuth Morris Pratt algorithms for their effective and efficient functionality and other applications use the basics of these algorithms for their functionalities as the Knuth Morris Pratt algorithm has less time complexity and Boyer Moore and Boyer Moore Horspool algorithms has preprocessing time complexity less. Other algorithms depend upon the type of input and they are efficient for certain or particular application.

Pandiselvam.P, Marimuthu.T and Lawrance. R [8] evaluated different kinds of string matching algorithms (Hamming, Levenshtein, Needleman-Wunsch, Smith waterman, Knuth Morris Pratt, Brute Force, Boyer Moore, Rabin Karp, AhoCorasick, CommentZ Walter) for biological sequences such as DNA and Proteins and observed their time and space complexities. It is analyzed that KMP algorithm is relatively easier to implement because it does not need to move backwards in the input sequence. It requires extra space. Rabin Karp algorithm used to detect the plagiarism; it requires additional space for matching. Brute Force algorithm does not require preprocessing of the text or the pattern, the problem is to that it's very slow and rarely produces efficient result. AhoCorasick algorithm is useful to multi pattern matching. CommentZ-Walter algorithm is taken more time to produce the result. The Boyer Moore algorithm is extremely fast for on large sequences, it avoids lots of needless comparisons by significantly pattern relative to text and its best case running complexity is sub linear.

Nwe Zin Oo [7] 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. This system uses Zawgyi Myanmar Font and implements using Java Programming Language and MySQL Server. For spelling checking, this proposed system corrects the spelling for any words of animals and plants but it can correct only three transformation operations for changing the input Myanmar word to the destination Myanmar word, and adds correctly spelt Myanmar words to the Dictionary. Each Myanmar input word is compared against a dictionary of correctly spelt Myanmar words. This proposed system improves the quality of corrections and suggestions for missed spelt Myanmar words and that need to dynamically determine how similar two strings are, such as Myanmar spell checkers.

Khaing Su Yee [3] analyzed the DNA and protein structure of HIV genome structure by using Levenshtein Distance Algorithm and determined what kind of behavior that the sequence has. Various structures of genome model which describe the characteristics of HIV have stored in the database. This system is implemented with 5 types of HIV DNA sequences, namely GAG, POL, ENV, NEF and LTR. Each data set has a significant DNA structure which forms the HIV virus structure. The system will find what kinds of DNA type that the input sequence has; for example if the input DNA has GAG DNA, then it can be said that the patient of input DNA sequence has been effected by HIV virus, and in case of ENV, NEF and LTR, the DNAs of patient are replicating and going to get the disease soon, and so on. In this system, the threshold is used to define the similarity of the input sequence with a set of sequences in the database. The distance value above threshold is not considered to be aligned; only sequences with distance less than threshold are considered to be similar sequences. A method for the task of computing edits distance and approximate string matching under the Levenshtein edit distance is presented. This thesis emphasized on the retrieval of similar DNA sequences from DNA sequences databases.

CHAPTER 3

SYSTEM ARCHITECTURE

One of the functions of the string matching is matching one string to the other. The proposed system compares Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm based on their f-score and execution time. And it can search not only the song title name but also the artist name. The user can input any English characters consulting with songs. This system can correct spelling errors dealing with songs. In the proposed system, Similarity Score is used for extracting the most similar songs information from the billboard song dataset. Similarity Score can extract the highest similarity songs information from the billboard song dataset depending on the user input songs information. F-score is used for comparing Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm.

3.1 System Detailed Design

Figure 3.1 shows the overview of the proposed system. “Select the Categories” is performed by choosing song type. The proposed system compares Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm. For Levenshtein, “Calculates Distance Using Levenshtein” is performed to calculate minimum edit distance by using Levenshtein Distance Algorithm. Then “Calculates Similarity Score” is performed to calculate similarity score. The proposed system shows song information that equal and above the similarity threshold (50) to the user. Then “Calculates F-Score” is performed to calculate f-score value. Finally display evaluation results to the user. For Needleman-Wunsch, “Calculates Score Using Needleman-Wunsch” is performed to calculate minimum alignment distance by using Needleman-Wunsch Distance Algorithm. And the next step also calculates as Levenshtein Distance.

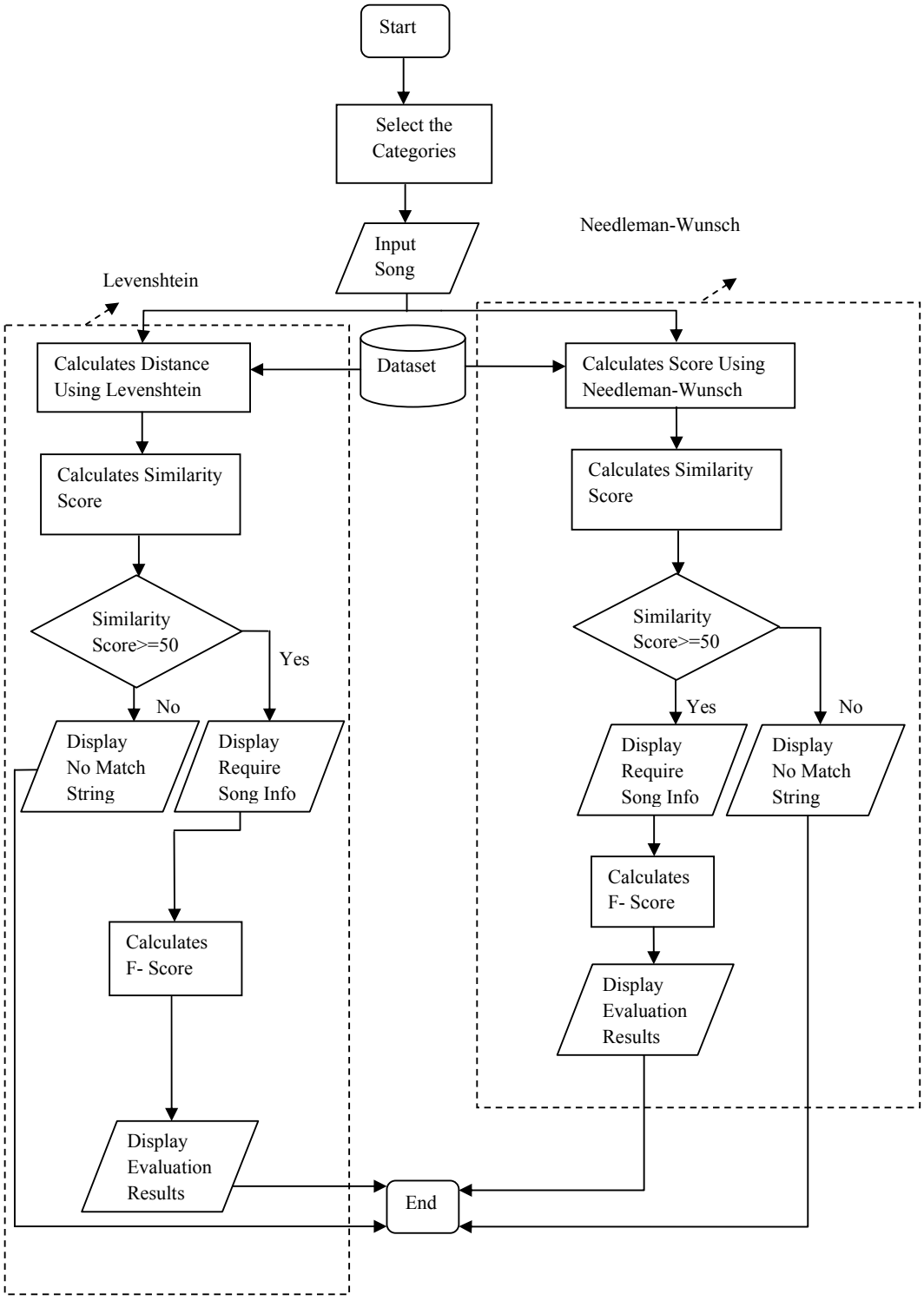

Figure 3.1 Detailed Design of the Proposed System

3.1.1 Finding Levenshtein Distance Algorithm

The allowed edit operations are insertion, deletion or substitution of a single character and each operation has the cost 1.

Mathematically, the Levenshtein distance between two strings a, b (of length |a| and |b| respectively) is given by $lev_{a,b}$ (|a|, |b|) where

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & if\ \min(i,j) = 0, \\ \min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & otherwise. \end{cases}$$

where $1_{(a_i \neq b_j)}$ is the indicator function equal to 0 when $a_i = b_j$ and equal to 1 otherwise, and $lev_{a,b}(i,j)$ is the distance between the first i characters of a and the first j characters of b.

The first element in the minimum corresponds to deletion (from a to b), the second to insertion and the third to match or mismatch, depending on whether the respective symbols are the same.

Computing the Levenshtein Distance constructs a matrix containing 0…m rows and 0…n columns, where n and m are the lengths of the two strings. The source string is s and the target string is t. Since either i or j, or both, is decrement on each pass of the loop, the maximum number of iterations is m+n and thus O (m+n) time. The greater the Levenshtein Distance, the more different the strings are. Figure 3.2 illustrates how Levenshtein Distance Algorithm works.

Begin

1. 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.

   Construct a matrix containing 0...n rows and 0...m columns.

2. Initialize the first row to 0...m. Initialize the first column to 0...n.

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

   incell d[n,m].

End

Figure 3.2 Levenshtein Distance Algorithm

The values in each remaining cell are calculated by considering its three neighbors as shown in Figure 3.3.

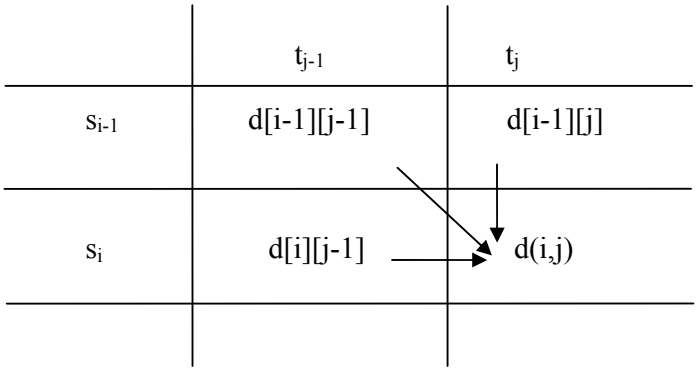

Figure 3.3 Considering Three Neighbors to Calculate Value Cell

Initialization Step is shown in Figure 3.4. In Figure 3.4, the length of source string is 4 and the length of target string is 5. So, it fills the first row to 0…5 and the first column to 0…4. In Figure 3.5, the character 'h' and 'h' compares that characters are match. So, the cost is 0. Then calculate the cell above, left, diagonally and fill the minimum value in cell d [1,1]. So, the minimum value (0) fill in cell d [1,1]. In Figure

3.6, the character ‘e’ and ‘e’ compares that characters are match. So, the cost is 0. Then calculate the cell above, left, diagonally and fill the minimum value in cell d [2,2]. So, the minimum value (0) fill in cell d [2,2].

Source String (s) = “helo”, Target String (t) = “hello”

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

(1)

begin at upper left (<=0)

to fill the first row to 0…m.

to fill the first column to 0…n.

Figure 3.4 Initialization Step

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

(2)

s[1] = t[1] ⟶ cost = 0
Above = d[0,1]+1 = 1+1 = 2
Left = d[1,0]+1 = 1+1 =2
Dia = d[0,0]+cost = 0+0 = 0
d[1,1] = min(Above, Left, Dia)

= 0

Figure 3.5 Matching “h” and “h”

		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			
o	4	3	2			

(3)

s[2] = t[2] ⟶ cost = 0
Above = d[1,2]+1 = 1+1 = 2
Left = d[2,1]+1 = 1+1 =2
Dia = d[1,1]+cost = 0+0 = 0
d[2,2] = min(Above, Left, Dia)

= 0

Figure 3.6 Matching “e” and “e”

In Figure 3.7, the character ‘l’ and ‘l’ compares that characters are match. So, the cost is 0. Then calculate the cell above, left, diagonally and fill the minimum value in cell d [3,3]. So, the minimum value (0) fill in cell d [3,3]. In Figure 3.8, the character ‘o’ and ‘l’ compares that characters are unmatch. So, the cost is 1. Then calculate the cell above, left, diagonally and fill the minimum value in cell d [4,4]. So, the minimum value (1) fill in cell d [4,4].

		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		

(4)

s[3] = t[3] ⟶ cost = 0
Above = d[2,3]+1 = 1+1 = 2
Left = d[3,2]+1 = 1+1 =2
Dia = d[2,2]+cost = 0+0 = 0
d[3,3] = min(Above, Left, Dia)

= 0

Figure 3.7 Matching “l” and “l”

		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	

(5)

s[4] = t[4] ⟶ cost = 1
Above = d[3,4]+1 = 1+1 = 2
Left = d[4,3]+1 = 1+1 =2
Dia = d[3,3]+cost = 0+1 = 1
d[4,4] = min(Above, Left, Dia)

= 1

Figure 3.8 Matching “o” and “l”

		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

(6)

s[4] = t[5] ⟶ cost = 0
Above = d[3,5]+1 = 2+1 = 3
Left = d[4,4]+1 = 1+1 =2
Dia = d[3,4]+cost = 1+0 = 1
d[4,5] = min(Above, Left, Dia)

= 1

Figure 3.9 Matching “helo” and “hello”

In Figure 3.9, the character ‘o’ and ‘o’ compares that characters are match. So, the cost is 0. Then calculate the cell above, left, diagonally and fill the minimum value in cell d [4,5]. So, the minimum value (1) fill in cell d [4,5].

The algorithm finishes when the length of the longest string is ended. The Levenshtein Distance of these two strings is in the lower right hand corner of the matrix, i.e., LD = 1 that shows how many changes are needed. This corresponds to our intuitive realization that “helo” can be transformed into “hello” by insertion of “l” at the fourth of input word (Addition). So, matching the strings “helo” and “hello”, one insertion need to make.

3.1.2 Finding Needleman-Wunsch Distance Algorithm

Needleman-Wunsch (NW) is used for string comparison to find the optimal global alignment between the two. The elements of two strings are joined up in an *alignment*. With respect to order, elements can be skipped. To skip an element, a gap (a blank character) is inserted in either or both of the strings.

Computing the Needleman-Wunsch Distance constructs a matrix containing 0…n rows and 0…m columns, where n and m are the lengths of the two strings. The source string is s and the target string is t. Since either i or j, or both, is decrement on each pass of the loop, the maximum number of iterations is m*n and thus O (m*n) time. The greater the Needleman-Wunsch Distance, the more different the strings are. Figure 3.10 illustrates how Needleman-Wunsch Distance Algorithm works.

Begin

1. 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.

   Construct a matrix containing 0...n rows and 0...m columns.

2. Initialization

   F (0, 0) = 0

   F (0, i) = i * d(d is the gapPenalty)

   F (j, 0) = j * d

3. 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)

4. F(i, j) = min { F(i − 1, j − 1) + s(xi, yj), case 1
   F(i − 1, j) + d , case2
   F(i, j − 1) + d , case3 }

5. Ptr(i, j) = { DIAG , if case 1
   LEFT , if case 2
   UP , if case 3 }

   The distance is found in cell F[n,m].

   End

Figure 3.10 Needleman-Wunsch Distance Algorithm

The algorithm gives two strings, one as source (s) and the other as target (t), then by examining all possible ways between the two strings, this method attempts to find the least cost of converting the source string to target string . The Needleman-Wunsch distance between two strings is defined as the minimum number of alignment distances needed to transform one string into the other. In alignment distance algorithm are considered match cost, mismatch cost and gap penalty cost. In the proposed system, the match cost is 0, the mismatch cost is 1 and the gap penalty cost is 2 are used.

The values in each remaining cell are calculated by considering its three neighbors as shown in Figure 3.11.

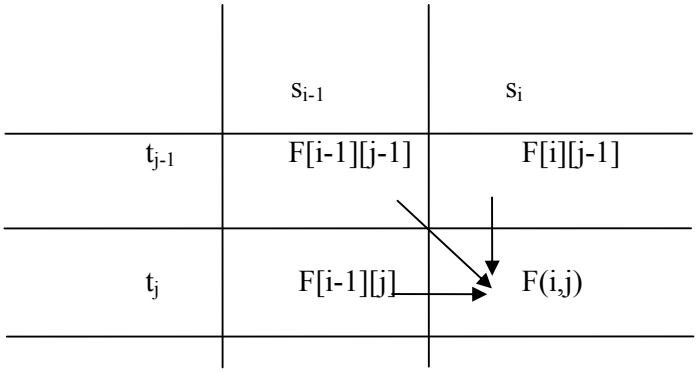

Figure 3.11 Considering Three Neighbors

Initialization Step is shown in Figure 3.12. In Figure 3.12, the length of source string is 4 and the length of target string is 5. So, it fills the first row to 0…5 and the first column to 0…4. In Figure 3.13, the character 'h' and 'h' aligns that characters are match. So, the value is 0. Then calculate the cell diagonally, left, up and fill the minimum value in cell F [1,1]. So, the minimum value (0) fill in cell F [1,1].

Source String (s) = "helo", Target String (t) = "hello"

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

(1)

begin at upper left (<=0)

to fill the first row to i*d.
to fill the first column to i*d.

Figure 3.12 Initialization Step for Needleman-Wunsch

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

(2)

"h" align "h" ⟶ match = 0

DIAG = F(0,0) + s(x0,y0) = 0

LEFT = F(0,1) + d = 4

UP = F(1,0) + d = 4

F(1,1) = min{DIAG,LEFT,UP) = 0

Figure 3.13 Aligning "h" and "h"

In Figure 3.14, the character 'e' and 'e' aligns that characters are match. So, the value is 0. Then calculate the cell diagonally, left and up and fill the minimum value in cell F [2,2]. So, the minimum value (0) fill in cell F [2,2]. In Figure 3.15, the character 'l' and 'l' aligns that characters are match. So, the value is 0. Then calculate the cell diagonally, left and up and fill the minimum value in cell F [3,3]. So, the minimum value (0) fill in cell F [3,3]. In Figure 3.16, the character 'o' and 'l' aligns that characters are unmatch. So, the value is 1. Then calculate the cell diagonally, left

and up and fill the minimum value in cell F [4,4]. So, the minimum value (1) fill in cell F [4,4].

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

(3)

"e" align "e" ⟶ match = 0

DIAG = F(1,1) + s(x2,y2) = 0

LEFT = F(1,2) + d = 4

UP = F(2,1) + d = 4

F(2,2) = min{DIAG,LEFT,UP) = 0

Figure 3.14 Aligning "e" and "e"

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

(4)

"l" align "l" ⟶ match = 0

DIAG = F(2,2) + s(x3,y3) = 0

LEFT = F(2,3) + d = 4

UP = F(3,2) + d = 4

F(3,3) = min{DIAG,LEFT,UP) = 0

Figure 3.15 Aligning "l" and "l"

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

(5)

"o" align "l" ⟶ mismatch = 1

DIAG = F(3,3) + s(x4,y4) = 1

LEFT = F(3,4) + d = 4

UP = F(4,3) + d = 4

F(4,4) = min{DIAG,LEFT,UP) = 1

Figure 3.16 Aligning "o" and "l"

In Figure 3.17, the character ‘o’ and ‘o’ aligns that characters are match. So, the value is 0. Then calculate the cell diagonally, left and up and fill the minimum value in cell F [4,5]. So, the minimum value (2) fill in cell F [4,5].

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

(6)

“o” align “o” ⟶ match = 0

DIAG = F(3,4) + s(x4,y5) = 2

LEFT = F(3,5) + d = 6

UP = F(4,4) + d = 3

F(4,5) = min{DIAG,LEFT,UP) = 2

Figure 3.17 Aligning “helo” and “hello”

The algorithm finishes when the length of the longest string is ended. The Needleman-Wunsch Distance of these two strings is in the lower right hand corner of the matrix, i.e., NW = 2. This corresponds to the intuitive realization that “helo” can be transformed into “hello” by aligning of h e l - o.

h e l l o

3.1.3 Calculating Similarity Score

Similarity Score is used to get more similar song information from the dataset. The proposed system shows the similar song information which is depended on the threshold value, Levenshtein Distance value and Needleman-Wunsch Distance value. To calculate the similarity score can be found by:

$$Similarity\ (source,target) = \frac{Distance(source,target)}{MaximumLength(source,target)} * 100 \qquad (4.1)$$

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

3.1.4 Measuring 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 4.2 and 4.3 formula.

$$Precision = \frac{TP}{TP+FP} * 100\% \quad (4.2)$$

$$Recall = \frac{TP}{TP+FN} * 100\% \quad (4.3)$$

where ,**TP** means the true positives (known matched name pairs classified as matches), **TN** means the true negatives (known un-matched name pairs classified as non-matches), **FP** means the false positives (unmatched name pairs classified as matches) and **FN** means the false negatives (known matched name pairs classified as non-matches). The proposed system is calculated for accuracy using the following formula.

$$f\text{-}score = 2((Precision * Recall)/(Precision+Recall)) \quad (4.4)$$

CHAPTER 4

IMPLEMENTATION

4.1Implementation of the Proposed System

The proposed system finds the string for all songs information and determines which input string has a similar with which song information. When the user wants to search the songs information, the user must choose the categories of songs (title or artist) and must enter the string in the input box. If it is not input the string, the system shows the message alerts that “Please enter a string”. If the user does not choose a type, the system shows the message alerts that “Please choose a type”.

When the user types song information, the system searches the required song information from the billboard song dataset using Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm. Even if the users input wrong spelling in the song title or artist name to search a song, it calculates the distance value between the input song and destination song. Similarity score also calculate the similarity value depending on the input string to get more similar songs. The similarity threshold is 50. Similarity values are equal or above the threshold, the song information shows to the user and are considered to get more similar. Similarity values are less than threshold value, the system shows the message alert that is no match string. So, the results can show the nearest song information with similarity score to the user.

F-score measures accuracy for Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm. The user can view average F-score graph by clicking “Average Accuracy Result” menu. For artist, user must click “For Artist” menu and for song title, user must click “For Song Title” menu. By clicking “For Artist” menu, the user can see “Experimental Result for Artist” page. By clicking “For Song Title” menu, the user can see “Experimental Result for Song Title” page. In “Experimental Result for Artist” page, the user can see testing data table and “Comparison of Accuracy” button. In “Experimental Result for Song Title” page, the user can also see testing data table and “Comparison of Accuracy” button. By clicking “Comparison of Accuracy” button in both pages, the user can know average accuracy

of Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm and can know which algorithm has better accuracy.

Then the user can also view average execution time graph by clicking “Result on Execution Time” menu. For artist, user must click “Average Time for Artist” menu and for song title, user must click “Average Time for Song Title” menu. By clicking “Average Time for Artist” menu, the user can see “Experimental Result for Artist” page. By clicking “Average Time for Song Title” menu, the user can see “Experimental Result for Song Title” page. In “Experimental Result for Artist” page, the user can see testing data table and “Comparison of Execution Time” button. In “Experimental Result for Song Title” page, the user can also see testing data table and “Comparison of Execution Time” button. By clicking “Comparison of Execution Time” button in both pages, the user can know the average execution time of Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm and can know which algorithm has long execution time.When the user uses this system, the user can see the home page as shown in Figure 4.1.

Figure 4.1 The Main Page of the Proposed System

The user can search the required song information by typing in the text box. But user does not type in the text box; the system shows the message alerts that

"Please enter a string" as in Figure 4.2. And it is also alerts "Please choose a type" as shown in Figure 4.3.

Figure 4.2 Showing Message Alert for Text Box

Figure 4.3 Showing Message Alert for Radio Button

When the user types "frrfld f dlfd" string in the text box by choosing "Artist" radio button, the system alerts that "Sorry! No match Strings!" as shown in Figure

4.4. In Figure 4.4, the user input is below the threshold similarity value (50). So the system does not show the song information to the user. Figure 4.5 shows that the user types “helo” string in the text box by choosing “Song Title” radio button.

Figure 4.4 Showing Message Alert for No Match String

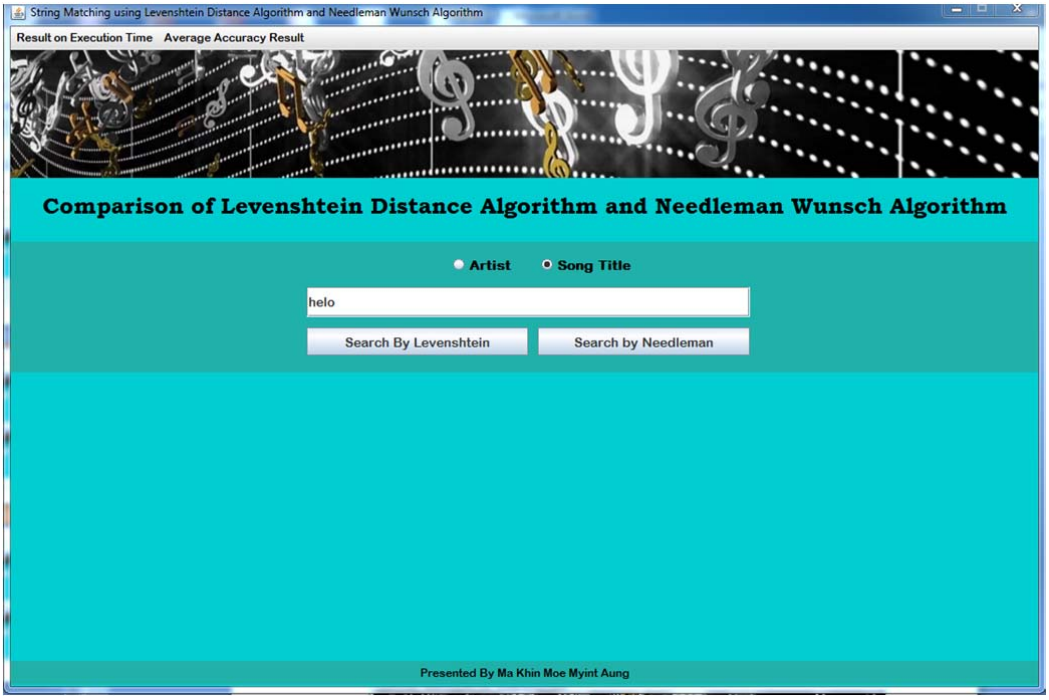

Figure 4.5 The Proposed System According to the User Input

Figure 4.6 Shows Song Information by Using Levenshtein Distance Algorithm

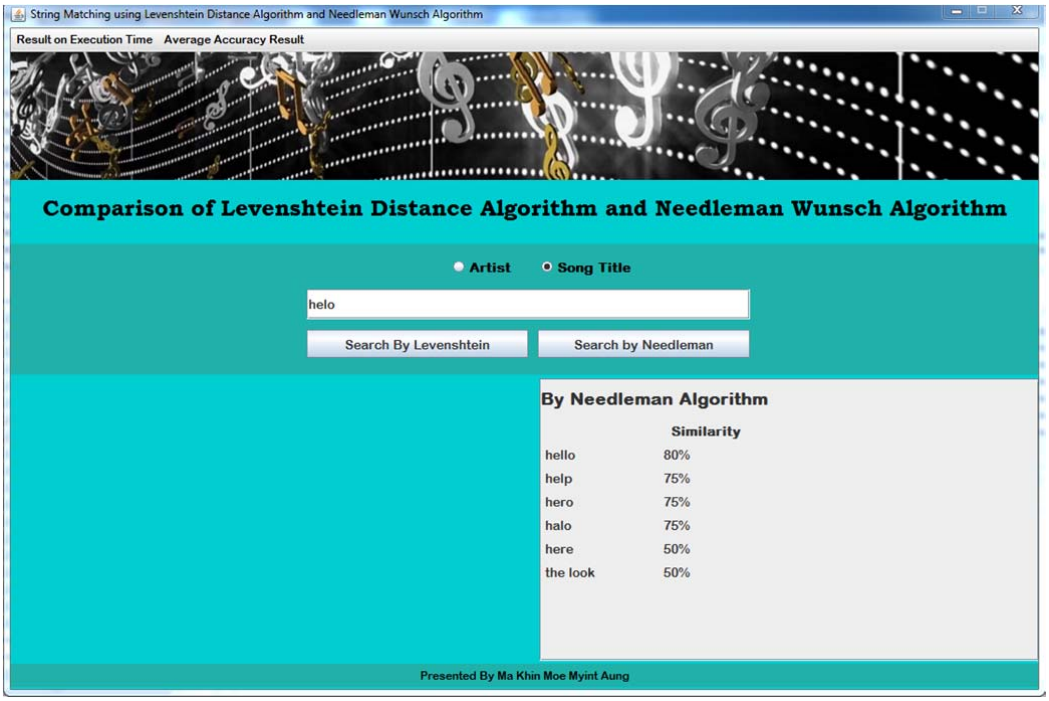

Figure 4.7 Shows Song Information by Using Needleman-Wunsch Distance Algorithm

In Figure 4.6, the user input “helo” song by choosing “Song Title” radio button. And then clicks “Search by Levenshtein” button. So, the system shows required song information with similarity score to the user.

In Figure 4.7, the user input “helo” song by choosing “Song Title” radio button. And then clicks “Search by Needleman” button. So, the system shows required song information with similarity score to the user.

When the user clicks both buttons by typing “helo” string, the system shows the song information to the user as shown in Figure 4.8.

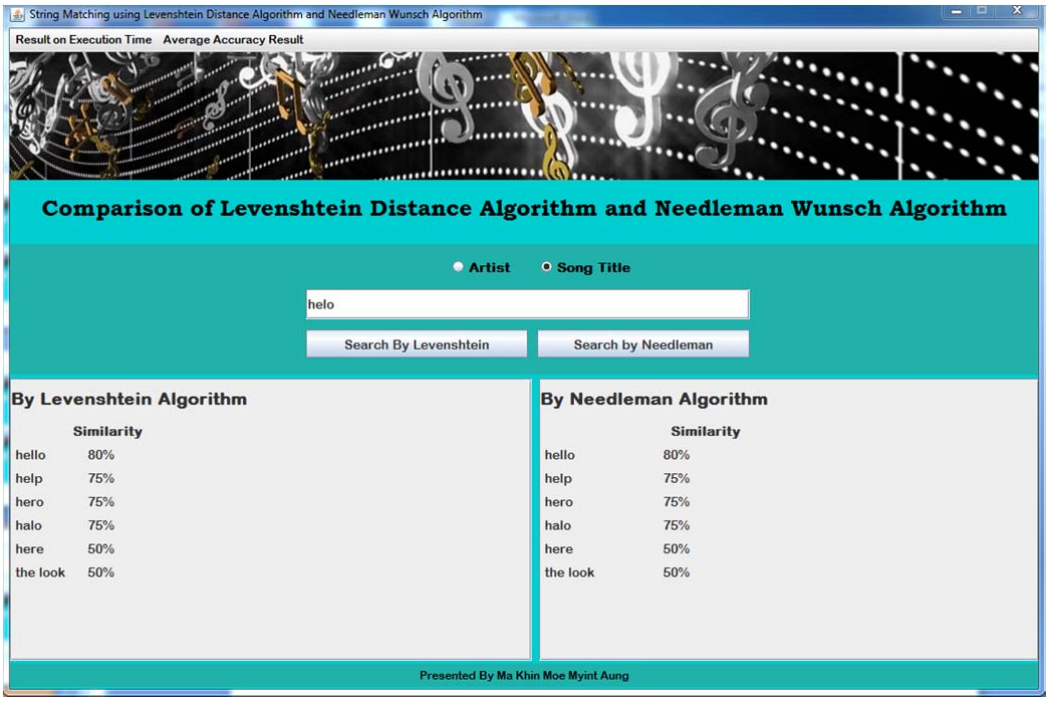

Figure 4.8 Shows Song Information by Using Both Algorithms

The user can view average execution time graph by clicking “Result on Execution Time” menu. For artist, user must click “Average Time for Artist” menu and for song title, user must click “Average Time for Song Title” menu. By clicking “Average Time for Song Title” menu, the user can see “Experimental Result for Song Title” page. In “Experimental Result for Song Title” page, the user can also see testing data table and “Comparison of Execution Time” button in Figure 4.9. When the user clicks “Comparison of Execution Time” button, the system shows the information to the user in Figure 4.10. So, the user can see the average execution time of Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm and can know which algorithm has long execution time.

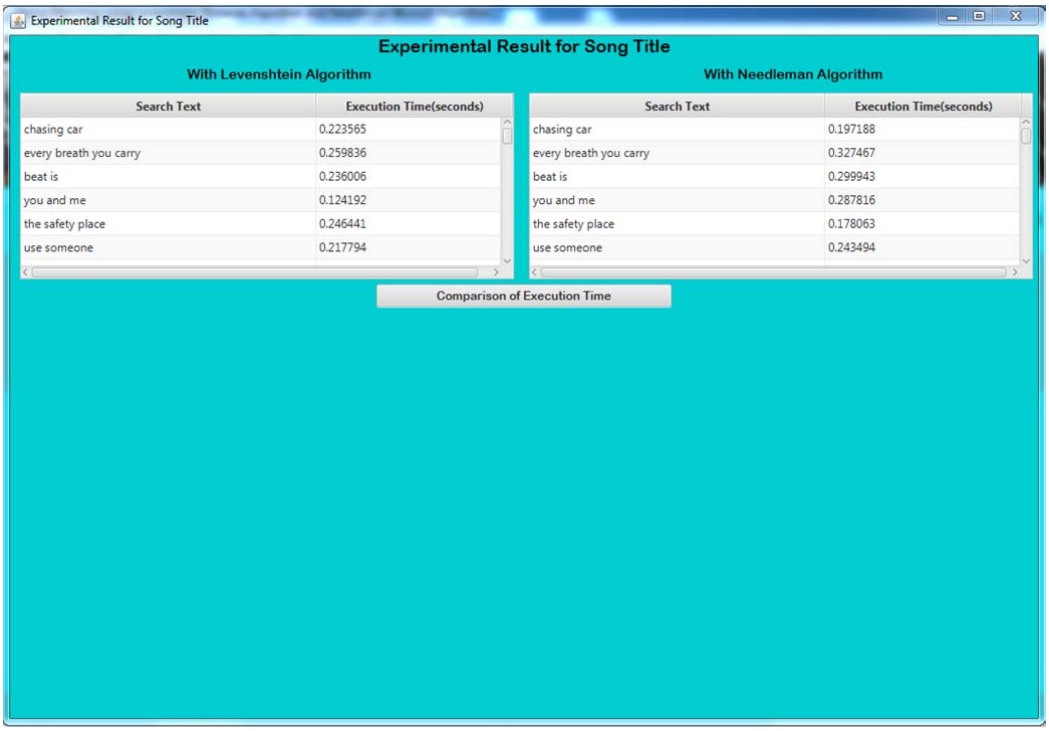

Figure 4.9 Results for Song Title

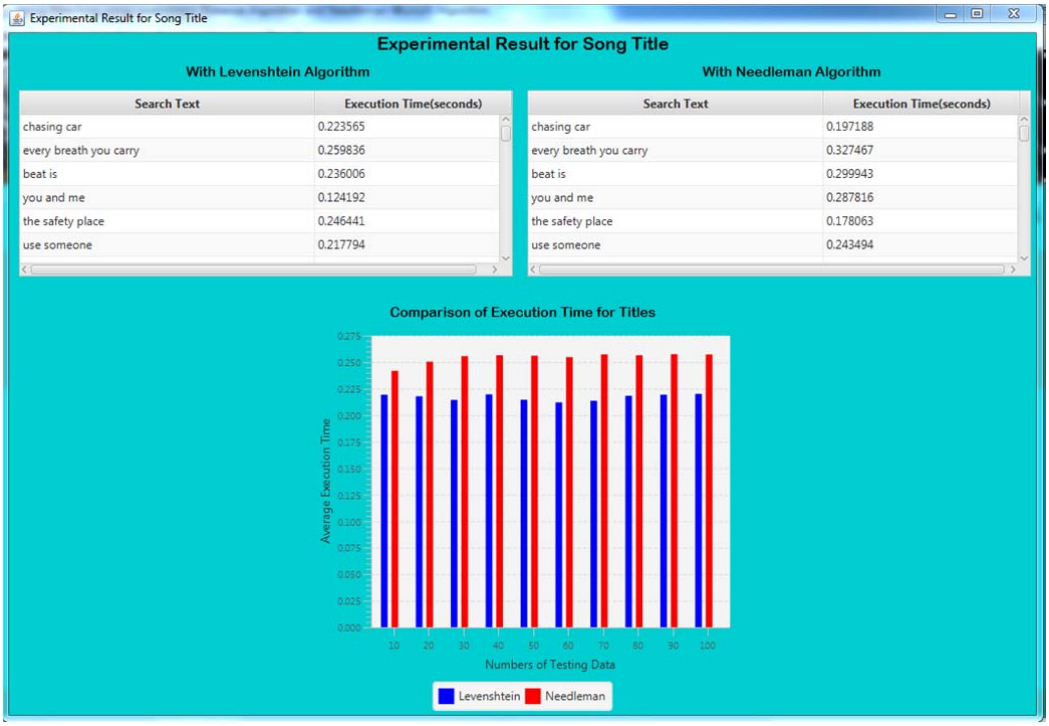

Figure 4.10 Experimental Results for Song Title with Time Graph

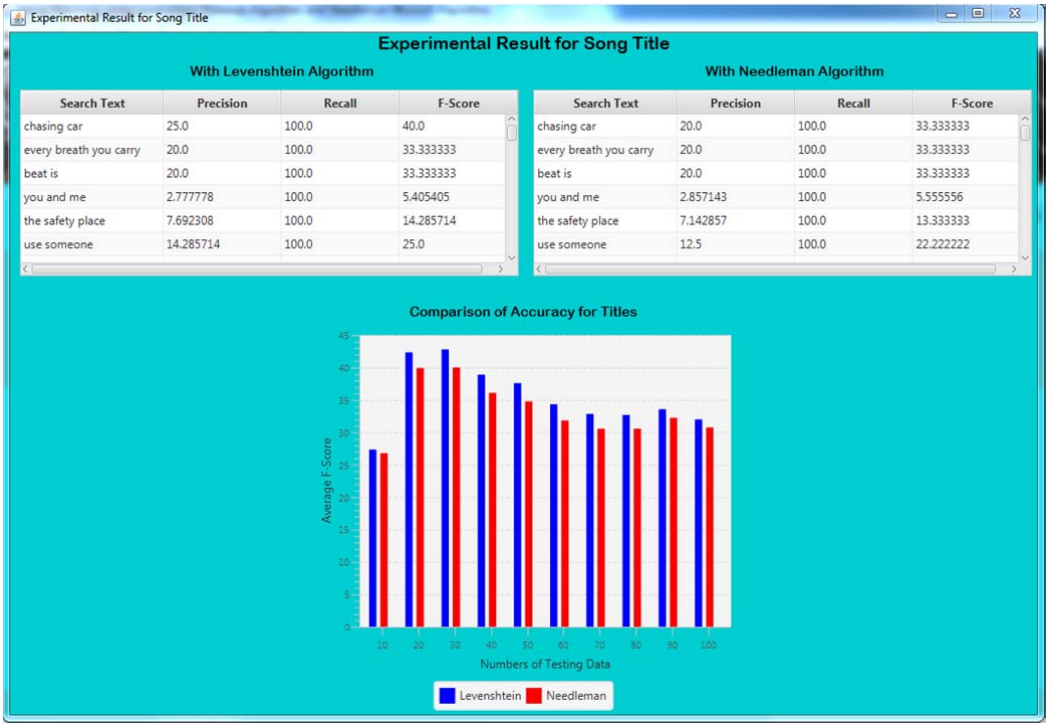

Figure 4.11 Experimental Results for Song Title with Graph

The user can also view average F-score graph by clicking “Average Accuracy Result” menu. If the user wants to search artist name, the user must click “For artist” menu and if the user wants to search song title name, user must click “For Song Title” menu. When the user clicks “For Song Title” menu, the user can see “Experimental Result for Song Title” page. In “Experimental Result for Song Title” page, the user can also see testing data table and “Comparison of Accuracy” button. When the user clicks “Comparison of Accuracy” button, the system shows the information to the user in Figure 4.11. So, the user can see average accuracy of both algorithms and can know which algorithm has better accuracy.

4.2 Experimental Result Analysis

The proposed system uses song information for Case 1 and Case 2.Case 1 is used 100 data for Artist feature in billboard song dataset with user input is same to the dataset. Case 2 is used 100 data for song title feature in billboard song dataset with wrong spelling. Levenshtein Distance Algorithm and Needleman-Wunsch Distance Algorithm are compared by using f-score and execution time for Case 1 and Case 2. Based on the number of f-score, different quality measures can be calculated.

The comparison of f-score graph for case 1 using artist data is shown in Figure 4.12and the user can know that Levenshtein Distance Algorithm has more relevant information than the Needleman-Wunsch Distance Algorithm and less irrelevant information than the Needleman-Wunsch Distance Algorithm. So, the user can see Levenshtein Distance Algorithm has better accuracy than the Needleman-Wunsch Distance Algorithm for case 1.

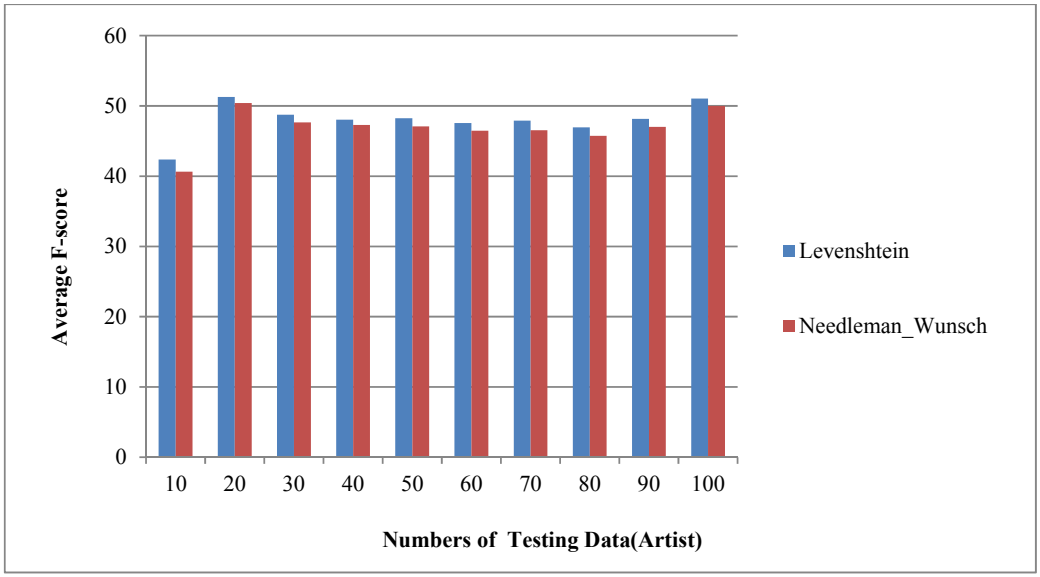

Figure 4.12 Comparison of *F*- score Graph for Case 1

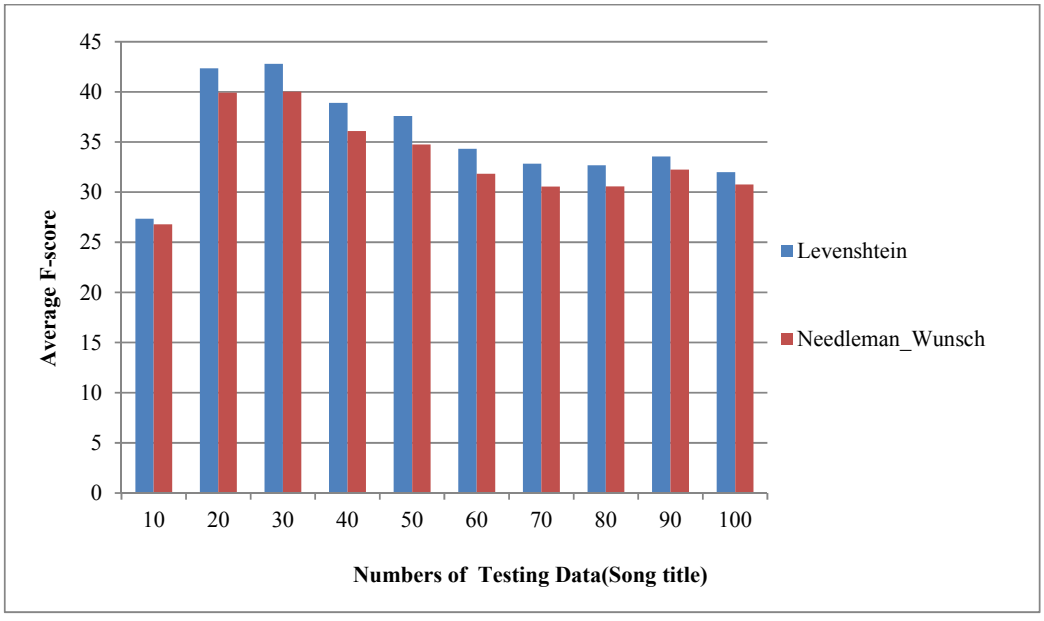

Figure 4.13 Comparison of *F*- score Graph for Case 2

The comparison of f-score graph for case 2 using song title data is shown in Figure 4.13 and the user can know that Levenshtein Distance Algorithm has more relevant information than the Needleman-Wunsch Distance Algorithm and less irrelevant information than the Needleman-Wunsch Distance Algorithm. So, the user can see Levenshtein Distance Algorithm has better accuracy than the Needleman-Wunsch Distance Algorithm for case 2.

So, according to the graph shown in Figure 4.12 and Figure 4.13, Levenshtein Distance Algorithm has better accuracy than the Needleman-Wunsch Distance Algorithm for both cases.

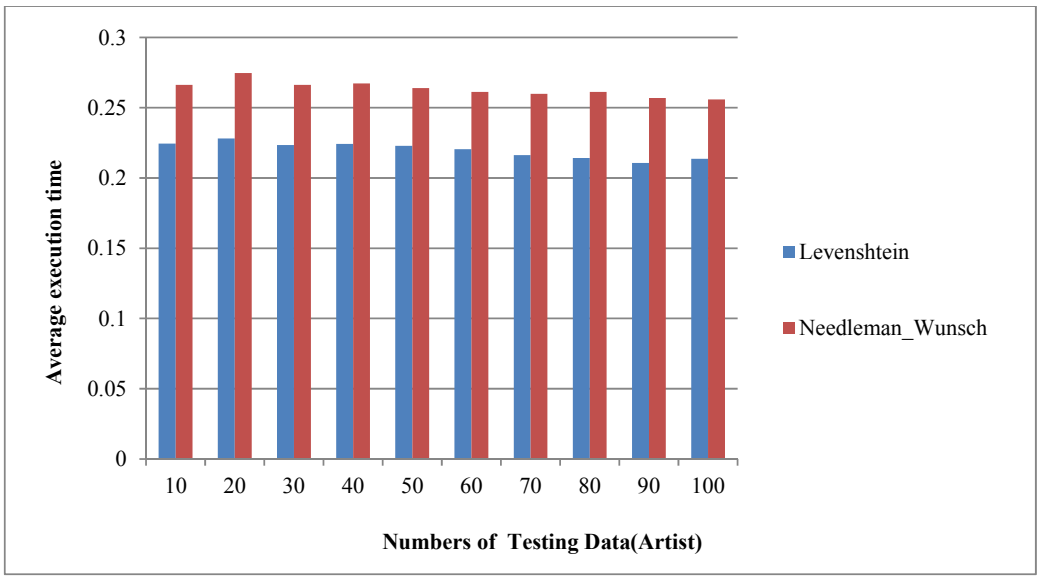

Figure 4.14 Comparison of Time Graph for Case 1

The execution time for the proposed system is measured in seconds. The comparison of execution time graph for case 1 using artist data is shown in Figure 4.14 and the user can see that Needleman-Wunsch Distance Algorithm has longer execution time than the Levenshtein Distance Algorithm. The comparison of execution time graph for case 2 using song title data is shown in Figure 4.15 and also sees that Needleman-Wunsch Distance Algorithm has longer execution time than the Levenshtein Distance Algorithm.

So, according to the graph shown in Figure 4.14 and Figure 4.15, Needleman-Wunsch Distance Algorithm has longer execution time than the Levenshtein Distance Algorithm for both cases.

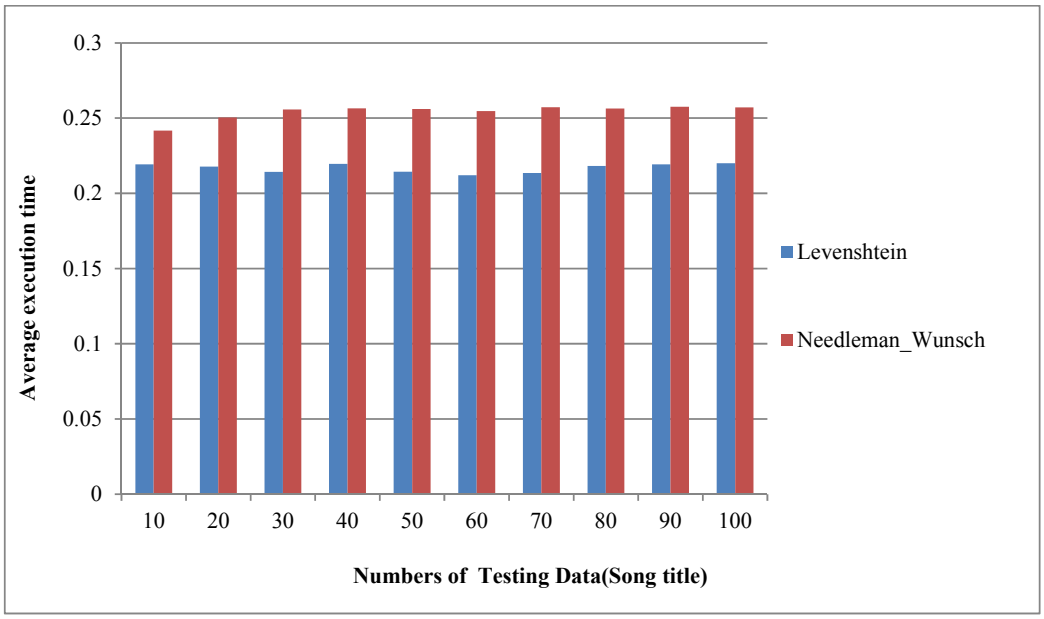

Figure 4.15 Comparison of Time Graph for Case 2

CHAPTER 5

CONCLUSION

5.1 Conclusion

String matching algorithm plays the vital role in String Computation. 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's title but also artist's name. This system is tested with many input data (about 500 input) for song information. And it is tested with a comparison of a single input. Then it could be seen that the f-score value of both algorithms depend on the input may be equal or more for the comparison of a single input. But Levenshtein Distance Algorithm has better accuracy than the Needleman-Wunsch Distance Algorithm for the comparison of average f-score. So, both algorithms may be equal or more for the comparison of a single input but the comparison of average f-score, Levenshtein Distance Algorithm has better accuracy than the Needleman-Wunsch Distance Algorithm for both cases.

And as a time complexity of both algorithms, Needleman-Wunsch Distance Algorithm has more complexity than the Levenshtein Distance Algorithm. But it could be seen that some data for Levenshtein Distance Algorithm also has a long execution time measured in seconds. But for the comparison of average time, Needleman-Wunsch Distance Algorithm also has longer execution time than the Levenshtein Distance Algorithm.

5.2 Limitations and Further Extensions

There are some limitations in the proposed system. This system can show the matching string with a similarity value equal and above the threshold. In this system, we mainly match song information consulting with Song Titles and Artists. This system mainly matches "character", so as a "word" it would be not flexible.

In the further, this system can be extended other desired threshold, other suitable algorithms to match the song information and other method to match "word" is flexible. And can be added to match lyrics information and can use other languages for song information such as Myanmar, Chinese, Japanese, Thai Languages and so on.

APPENDIX

Sample Datasets

The following are the sample of billboard song dataset for 1965 (rank 1 to 100).

Rank	Song	Artist	Year
1	wooly bully	sam the sham and the pharaohs	1965
2	icant help myself sugar pie honey bunch	four tops	1965
3	icant get no satisfaction	the rolling stones	1965
4	you were on my mind	we five	1965
5	youve lost that lovinfeelin	the righteous brothers	1965
6	Downtown	petulaclark	1965
7	Help	the beatles	1965
8	cant you hear my heart beat	hermans hermits	1965
9	crying in the chapel	elvispresley	1965
10	my girl	the temptations	1965
11	help me Rhonda	the beach boys	1965
12	king of the road	roger miller	1965
13	the birds and the bees	jewel akens	1965
14	hold me thrill me kiss me	mel carter	1965
15	Shotgun	junior walker the all stars	1965
16	i got you babe	sonny cher	1965
17	this diamond ring	garylewis the playboys	1965
18	the in crowd	ramseylewis trio	1965
19	mrs brown youve got a lovely daughter	hermans hermits	1965
20	stop in the name of love	the supremes	1965
21	unchained melody	the righteous brothers	1965
22	Silhouettes	hermans hermits	1965
23	ill never find another you	the seekers	1965
24	caramia	jay and the americans	1965
25	mr tambourine man	the byrds	1965
26	cast your fate to the wind	sounds orchestral	1965
27	yes im ready	barbara mason	1965
28	whats new pussycat	tom jones	1965
29	eve of destruction	barrymcguire	1965
30	hang on sloopy	the mccoys	1965
31	ticket to ride	the beatles	1965
32	red roses for a blue lady	bertkaempfert	1965
33	papas got a brand new bag	james brown	1965

Rank	Song	Artist	Year
34	game of love	waynefontana the mindbenders	1965
35	the name game	shirleyellis	1965
36	i know a place	petulaclark	1965
37	back in my arms again	the supremes	1965
38	baby im yours	barbaralewis	1965
39	the jolly green giant	the kingsmen	1965
40	hush hush sweet charlotte	patti page	1965
41	like a rolling stone	bob dylan	1965
42	im telling you now	freddie and the dreamers	1965
43	ferry cross the Mersey	gerry and the pacemakers	1965
44	just once in my life	the righteous brothers	1965
45	the seventh son	johnny rivers	1965
46	imhenery the eighth i am	hermans hermits	1965
47	a walk in the black forest	horst jankowski	1965
48	for your love	the yardbirds	1965
49	california girls	the beach boys	1965
50	go now	the moody blues	1965
51	Goldfinger	shirleybassey	1965
52	down in the boondocks	billy joe royal	1965
53	baby the rain must fall	glennyarbrough	1965
54	catch us if you can	the daveclark five	1965
55	eight days a week	the beatles	1965
56	just a little	the beau brummels	1965
57	you turn me on	ianwhitcomb	1965
58	ill be doggone	marvingaye	1965
59	save your heart for me	garylewis the playboys	1965
60	tired of waiting for you	the kinks	1965
61	count me in	garylewis the playboys	1965
62	all day and all of the night	the kinks	1965
63	what the world needs now is love	jackiedeshannon	1965
64	its not unusual	tom jones	1965
65	shes about a mover	sir douglas quintet	1965
66	Shake	samcooke	1965
67	wonderful world	hermans hermits	1965
68	nowhere to run	martha and the vandellas	1965
69	heart full of soul	the yardbirds	1965
70	love potion no 9	the searchers	1965
71	laurie strange things happen	dickey lee	1965
72	baby dont go	sonny cher	1965
73	it aint me babe	the turtles	1965
74	tell her no	the zombies	1965
75	i go to pieces	peter and gordon	1965

Rank	Song	Artist	Year
76	red roses for a blue lady	vic dana	1965
77	dont just stand there	patty duke	1965
78	the tracks of my tears	the miracles	1965
79	too many rivers	brenda lee	1965
80	i like it like that	the daveclark five	1965
81	little things	bobby goldsboro	1965
82	true love ways	peter and gordon	1965
83	its the same old song	four tops	1965
84	youve got your troubles	the fortunes	1965
85	hold what youve got	joe tex	1965
86	we gotta get out of this place	the animals	1965
87	laugh laugh	the beau brummels	1965
88	the last time	the rolling stones	1965
89	do you believe in magic	the lovin spoonful	1965
90	all i really want to do	cher	1965
91	take me back	little anthony and the imperials	1965
92	i want candy	the strangeloves	1965
93	ooo baby baby	the miracles	1965
94	laugh at me	sonny bono	1965
95	treat her right	roy head	1965
96	the race is on	jack jones	1965
97	im a fool	dino desi billy	1965
98	the boy from new york city	the ad libs	1965
99	keep searchin well follow the sun	del shannon	1965
100	how sweet it is to be loved by you	marvingaye	1965

REFERENCES

[1] Abdulla Ali, "*Textual Similarity*", Bachelor Thesis, Institute for Molecules and Materials: ISSN 2011-19, Denmark, June 2011.

[2] Guyer, Atasoy, Somyurek, "*Measuring Disorientation Based on the Needleman-Wunsch Algorithm*", International Review of Research in Open and Distributed LearningVolume 16, Number 2, 2015.

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

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

[5] Maria Del Pilar Angeles, Adrian Espino-Gamez, "*Comparison of methods Hamming Distance, Jaro, and Monge-Elkan*", The 7th International Conference on Advances in Databases, Knowledge, and Data Applications, DBKDA 2015.

[6] NimishaSingla, Deepak Garg, "*String Matching Algorithms and their Applicability in various Applications*", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-I, Issue-6, January 2012.

[7] Nwe Zin Oo,"*Myanmar Words Spelling Checking Using Levenshtein Distance Algorithm*", M.C.Sc Thesis, University of Computer Studies, Yangon, December 2010.

[8] Pandiselvam.P, Marimuthu.T, Lawrance.R, "*A Comparative Study On String Matching Algorithms Of Biological Sequences*", International Conference on Intelligent Computing, Cornell University Library, Jan 29, 2014.

[9] Ratmalana, Sri Lanka, "*A Comparative Analysis of Various String Matching Algorithms*", Proceedings of 8th International Research Conference, KDU, November 2015.

[10] RishinHaldar, Debajyoti Mukhopadhyay, "*Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach*", The Computing Research Repository, Cornell University Library, January 2011.

[11] WaelH.gomaa, Aly A.Fahmy, "*A Survey of Text Similarity Approaches*", International Journal of Computer Applications (0975 – 8887) Volume 68– No.13, April 2013.

[12] Yaser Jararweh1.Mahmoud A1-Ayyoub1.Maged Fakirah1.Luay Alawneh1.Brij B.Gupta2, "*Improving the performance of the needleman-wunsch algorithm using parallelization and vectorization techniques*", Multimedia Tools and Applications, DOI: 10.1007/s11042-017-5092-0, 2017.