International Journal of Latest Trends in Engineering and Technology
Special Issue SACAIM 2017, pp. 138-142
e-ISSN:2278-621X

A COMPARATIVE ANALYSIS OF STRING METRIC ALGORITHMS

BabithaD'Cunha[1], RoshiniKarishmaKarkada[2] & Mrs.SuchethaVijaykumar[3]

Abstract: String metric is an important attribute for analysing a string pattern. A string metric provides a number indicating algorithm-specific indication distance. String metric includes many algorithms. One such algorithm is Levenshtein distance is a measure of the similarity between two strings, which we will refer to as the source string and the target string. Hamming distance algorithm is to find the distance between two strings of same length and the number of positions in which the corresponding symbols are different. Lee distance algorithm is used to state the number and length of disjoint paths between two nodes in a torus. We hereby propose to do a comparative study of all the above three algorithms by taking factors such as time and volume of the string to be searched and do an analysis of the same.
Key Words: String Metric, Lee distance, Hamming distance,Levenshtein distance.

1. INTRODUCTION

A string metric (also known as a string similarity metric or string distance function) is a metric that measures distance ("inverse similarity") between two text strings for approximate string matching. String matching is a technique to discover pattern from the specified input string. String matching algorithms are used to find the matches between the pattern and specified string.

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, similarity measure etc. We analyse the similarity measurements on Protein, DNA and RNA sequences by using various kinds of string matching algorithms such as Boyer Moore (BM) algorithm, NW algorithm, SW algorithm, Hamming Distance, Levenshtein Distance, Aho-Corasick (AC) algorithm, KMP algorithm, Rabin Karp algorithm, Comment Walter (CZW) algorithm.

In this paper we have explained about Hamming distance, Lee distance and Levenshtein Distance. The Hamming distance 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. The Lee distance The Levenshtein distance is a string metric for measuring the difference between two sequences

String metrics are used heavily in information integration and are currently used in areas including fraud detection, fingerprint analysis, plagiarism detection, merging, DNA, RNA analysis, image analysis, evidence-based machine learning, database data deduplication, data mining, incremental search, data integration, and semantic knowledge

2. DIFFERENT TYPES OF STRING METRIC ALGORITHMS

2.1 Hamming distance:

The Hamming distance is named after Richard Hamming, who introduced the concept in his basic paper about the error detection and error signals in 1950s. The hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. In other words, the minimum number of alternatives may be measured to change one string to another, or the number of minimum errors that can be converted from one string to another. In most common scenarios, the Hamming distance is one of several string metrics for measuring the edit distance between two sequences.

The hamming distance is used to define some essential concepts in coding theory, such as error detection and error correction. Specifically, a C code is said to have K-errors detecting if any two codewards c1 and c2 from C, which has a lower Hamming distance k coincide; Otherwise, if a signal detects k-bugs, and the minimum hamming distance between any two codwords is at least k + 1.

It measures the minimum number of Substitutions to forming a DNA sequence to another. The following formula belonging to the hamming distance.d HAD (i, j) = S [yi, k? yj, k] ... (1) where, d HAD, the number of dissimilarity between the two sequences. yi is the first sequence, yj is the second sequence. K is the pairing variable. The equation (1) is used to find the

[1] III SemM.Sc, Department of Software Technology, St. Aloysius College AIMIT, Mangalore
[2] III SemM.Sc, Department of Software Technology, St. Aloysius College AIMIT, Mangalore
[3] Asst. Professor M.Sc.ST, St. Aloysius College AIMIT, Mangalore

minimum number of substitutions to form one sequence to another. The distance itself gives the number of mismatches between the variable paired by k. Distance is applied to the biological sequences to find minimum number of substitutions for changing one sequence to another

Algorithm:

Steps:

1. It is non-negative: $d(x,y) \geq 0$
2. It is only zero for identical inputs: $d(x,y) = 0 \Leftrightarrow x = y$
3. It is symmetric: $d(x,y) = d(y,x)$
4. It obeys the triangle inequality, $d(x,z) \leq d(x,y) + d(y,z)$

2.2 Levenshtein distance:

Levenshtein distance may also be referred to as edit distance. It is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance measure of the similarity between two strings, which we will refer to as the source string (s) and the target string (t). It gives the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other. It is named after Vladimir Levenshtein, who considered this distance in 1965.

For example, the Levenshtein distance between "kitten" and "sitting" 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. kitten → sitten (substitution of "s" for "k")
2. sitten → sittin (substitution of "i" for "e")
3. sittin → sitting (insertion of "g" at the end).

Upper and lower bounds

The Levenshtein distance has several simple upper and lower bounds. These include:

- It is at least the difference of the sizes of the two strings.
- It is at most the length of the longer string.
- It is zero if and only if the strings are equal.
- If the strings are the same size, the Hamming distance is an upper bound on the Levenshtein distance.
- The Levenshtein distance between two strings is no greater than the sum of their Levenshtein distances from a third string (triangle inequality).

An example where the Levenshtein distance between two strings of the same length is strictly less than the Hamming distance is given by the pair "flaw" and "lawn". Here the Levenshtein distance equals 2 (delete "f" from the front; insert "n" at the end). The Hamming distance is 4.

Algorithm:

Steps:

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

2.3 Lee distance:

It displays an extensive-first search for a low-way connection between a connection source and a destination node and searching each grid point by labeling it from the source. This extension will reach the destination node if a connection is possible. The rear phase of the second trace forms a connection followed by any way to reduce labels. This algorithm is guaranteed to find the lowest path between the source and the destination for the given connection. However, a connection can block other contacts when multiple connections are made. The wave expansion marks only points in the routable area of the chip, not in the blocks or already wired parts. It's best to keep in one direction as long as possible to reduce the division.

Algorithm:

Steps:

1. Initialization: - Select start point, mark with 0, i := 0
2. Wave expansion: REPEAT
   - Mark all unlabeledneighbors of points marked with i with i+1

- i := i+1
  i. UNTIL ((target reached) or (no points can be marked))

3. Back trace: go to the target point
   - REPEAT
   - go to next node that has a lower mark than the current node
   - add this node to path
   - UNTIL (start point reached)
4. Clearance: - Block the path for future wirings
   - Delete all marks.

3. ANALYSIS

We have tried to make a comparative study and analysis on three algorithms of string metric namely - Hamming distance, Levenshtein distance and Lee distance and implemented them in Java and studied their behavior. Analysis is made based on time taken for different lengths of input pattern.

AU- All Uppercase	**AL-All Lowercase**	**M-Mixed case**

3.1 Hamming distance:

Minimum length of pattern	Running time (s) for Different Input nature								
	AU			AL			M		
	1	2	3	1	2	3	1	2	3
Short	11	6	5	6	4	4	9	6	8
Medium	13	10	11	12	10	6	15	11	18
Long	28	35	45	22	32	26	30	28	29

Table 1: Hamming distance

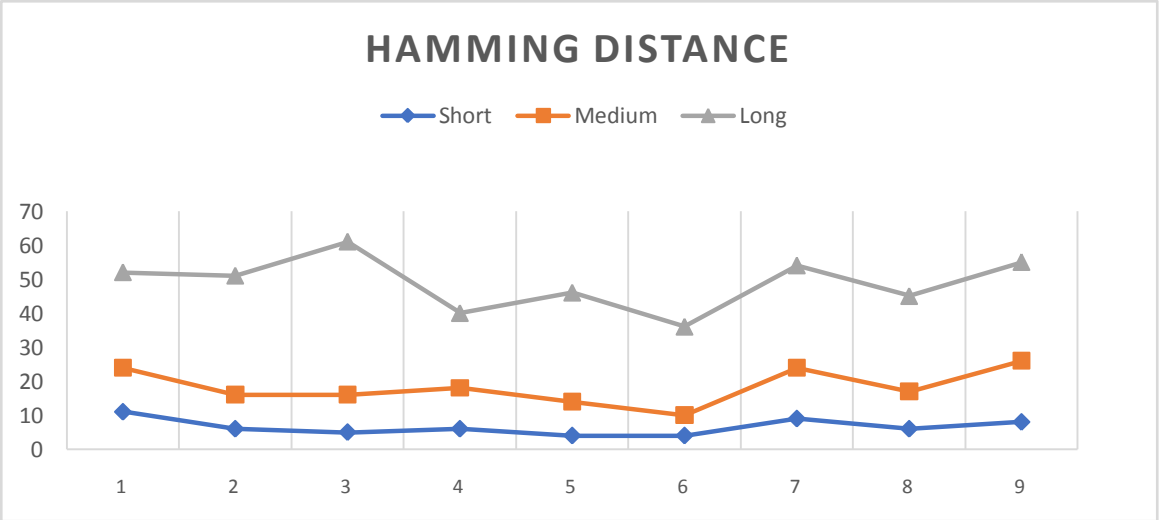

Figure 1: Graph for string length vs Time taken using Hamming Distance

3.2 Levenshtein Distance:

Minimum length of pattern	Running time (s) for Different Input nature								
	AU			AL			M		
	1	2	3	1	2	3	1	2	3
Short	13	5	4	6	5	10	7	9	5
Medium	7	4	11	12	7	8	16	14	14
Long	17	30	21	25	24	15	39	36	37

Table 2:Levenshtein distance

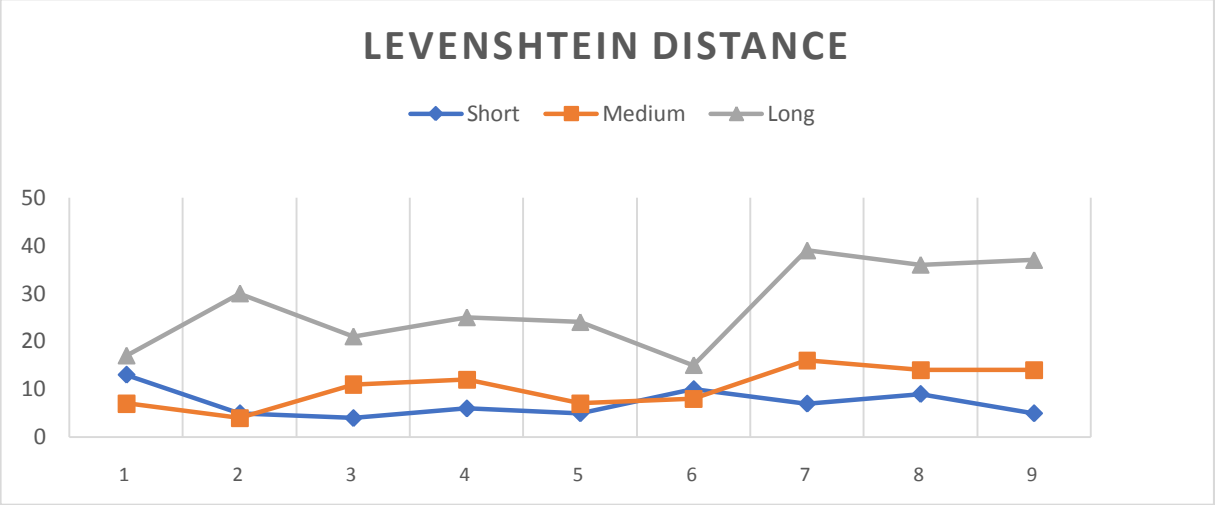

Figure 2:Graph for string length vs Time taken for Levenshtein Algorithm

3.3 Lee Distance:

Minimum length of pattern	Running time (s) for Different Input nature								
	AU			AL			M		
	1	2	3	1	2	3	1	2	3
Short	13	13	10	21	28	18	19	20	20
Medium	14	19	21	28	22	21	30	17	18
Long99	25	26	24	30	23	20	44	56	41

Table 3:Lee Distance

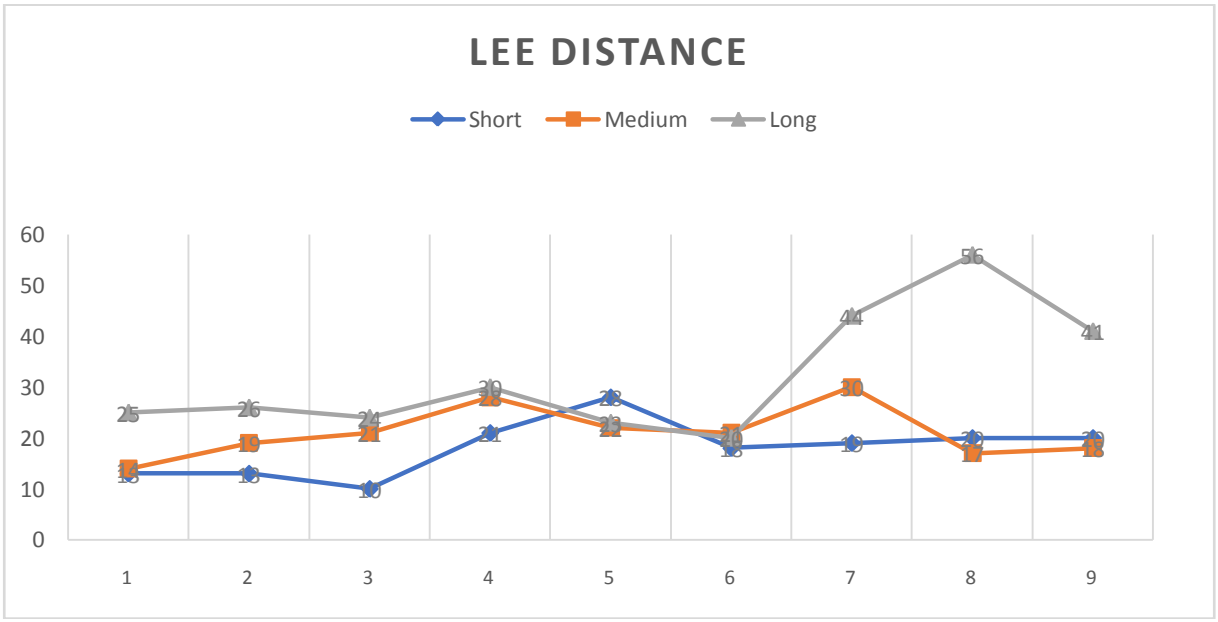

Figure 3:Graph for string length vs Time taken for Lee Distance Algorithm

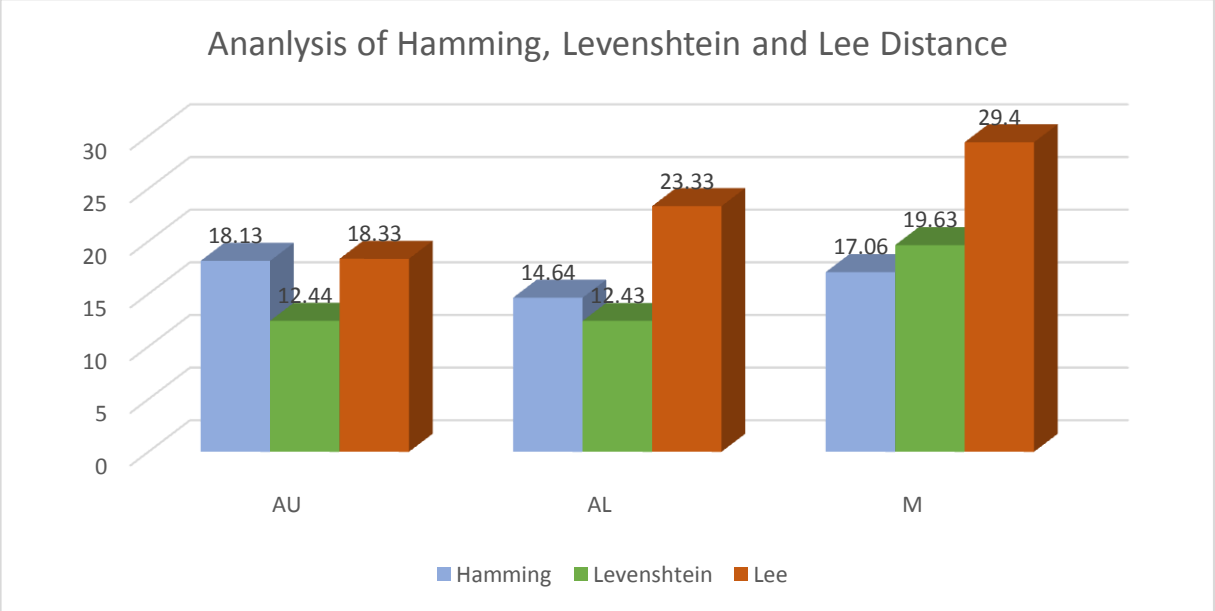

Figure 4: Comparison of Hamming Distance, Levenshtein Distance and Lee distance

4. CONCLUSION

In this paper, we have tried to make a comparative study and analysis on three algorithms of string metric. We have also implemented the three algorithms namely - Hamming distance, Levenshtein distance and Lee distance in Java and studied their behavior. A brief analysis is made based on time taken for different lengths of input pattern. We have observed that Levenshtein distance is faster compared to Hamming distance and Lee distance for all Uppercase and all lowercase inputs including large sized input. Hamming distance algorithm is best for Mixed inputs.

5. REFERENCES

[1] https://github.com/haifengl/smile/blob/master/math/src/main/java/smile/math/distance/HammingDistance.java
[2] http://sanjayshukla1975.blogspot.in/2015/02/hamming-distance.html
[3] https://www.informationvine.com/index?o=603069&l=sem&qo=serpSearchTopBox&q=hamming+code+for+string+in+java
[4] https://blogs.ucl.ac.uk/chime/2010/06/28/java-example-code-of-common-similarity-algorithms-used-in-data-mining/
[5] https://www.cs.cmu.edu/afs/cs/Web/People/wcohen/postscript/kdd-2003-match-ws.pdf
[6] http://www.cs.utexas.edu/~ai-lab/pubs/marlin-ijcai-wkshp-03.pdf
[7] https://arxiv.org/ftp/arxiv/papers/1401/1401.7416.pdf
[8] https://itssmee.wordpress.com/2010/06/28/java-example-of-hamming-distance/
[9] http://www.encyclopedia.com/computing/dictionaries-thesauruses-pictures-and-press-releases/lee-distance
[10] http://www.cs.tut.fi/~jta/astola.pdf
[11] https://www.encyclopediaofmath.org/index.php/Lee_distance
[12] https://www.cs.cmu.edu/afs/cs/Web/People/wcohen/postscript/kdd-2003-match-ws.pdf
[13] https://arxiv.org/ftp/arxiv/papers/1401/1401.7416.pdf