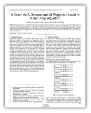
Combination of Levenshtein Distance and Rabin-Karp to Improve the Accuracy of Document Equivalence Level

Andysah Putera Utama Siahaan, Rusiadi Rusiadi



International Journal of Engineering & Technology, *7 (2.27) (2018) 17-21*

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Combination of levenshtein distance and rabin-karp to improve the accuracy of document equivalence level

Andysah Putera Utama Siahaan [1, 6] *, Solly Aryza [1, 6], Eko Hariyanto [1], Rusiadi [2], Andre Hasudungan Lubis [3], Ali Ikhwan [4, 6], Phak Len Eh Kan [5]

[1] *Faculty of Science and Technology, Universities Pembangunan Panca Budi, Medan, Indonesia*
[2] *Faculty of Social Science, Universities Pembangunan Panca Budi, Medan, Indonesia*
[3] *Faculty of Engineering, Universities Medan Area, Medan, Indonesia*
[4] *Faculty of Science and Technology, Universities Islam Negeri Sumatera Utara, Medan, Indonesia*
[5] *School of Computer and Communication Engineering, University Malaysia Perlis, Pauh, Malaysia*
[6] *Ph.D. Student of School of Computer and Communication Engineering, University Malaysia Perlis, Pauh, Malaysia*
**Corresponding author E-mail: andiesiahaan@gmail.com*

Abstract

Rabin Karp algorithm is a search algorithm that searches for a substring pattern in a text using hashing. It is beneficial for matching words with many patterns. One of the practical applications of Rabin Karp's algorithm is in the detection of plagiarism. Michael O. Rabin and Richard M. Karp invented the algorithm. This algorithm performs string search by using a hash function. A hash function is the values that are compared between two documents to determine the level of similarity of the document. Rabin-Karp algorithm is not very good for single pattern text search. This algorithm is perfect for multiple pattern search. The Levenshtein algorithm can be used to replace the hash calculation on the Rabin-Karp algorithm. The hash calculation on Rabin-Karp only counts the number of hashes that have the same value in both documents. Using the Levenshtein algorithm, the calculation of the hash distance in both documents will result in better accuracy.

Keywords: *Rabin-Karp; Levenshtein; Hash; Information Retrieval; Plagiarism.*

1. Introduction

Data is a valuable asset that must be protected [1]–[3]. Data is digital information transmitted through electrical signal [4]–[6] It is divided into two types, writing, and sound. It may be manipulated or duplicated without the owner's knowledge. Plagiarism is an act of evil in the falsification of a work. It is done to increase the popularity of a plagiarist. This act is an act of acknowledging the work of others to be his work without quoting the source. This action is often done in the world of academics. This activity violates the law and can be charged in certain articles. Plagiarism can be detected with the help of methods in computer science [7]. With the advent of computer science technology, plagiarism can be avoided. Text mining is part of the science that discusses the processing of words. Text mining algorithms have several methods that can detect the fraud that occurs in the documents. Frequent methods for detecting plagiarism include Rabin-Karp and Levenshtein methods. Rabin-Karp method is used to detect the level of similarity of documents that occur [8]. This method will calculate the relationship values of each word in the comparable documents [9]. In the process of detecting plagiarism using the Rabin-Karp method, this should be supported by pre-processing to do word mining. Stemming is one of the parts applied to filter words by prefix and word suffix. It aims to improve results on the retrieval of information by transforming the words in the document into a basic word form. The Levenshtein algorithm works by calculating the distance of one word to another so that the degree of similarity of two documents based on the distance weight can be determined. This study combines Levenshtein calculation techniques on hash calculations on the Rabin-Karp algorithm. This combination is expected to improve the accuracy of the documented equivalence level.

2. Theories

2.1. Text mining

Text mining has the definition of data mining in the form of text where the data source is usually obtained from the document [10]–[12]. The purpose of text mining is to search for words that can represent the contents of the document so that it can be analyzed interconnection. Mining can also be done to speed up the search, compression [13], decision support systems [14]–[16] expert systems [17] and improved data security. Text mining is a process of pattern extraction in the form of useful information and knowledge from a large number of text data sources [18]; it is called unstructured data and is the primary differentiator with data mining using structured data or database as input. Text mining may be considered a two-stage process beginning with the application of structures to text data sources and followed by the extraction of relevant information and knowledge from this structured text data using the same techniques and tools like data mining. Common processes performed by text mining include automatic summarization, document categorization, and text placements. In text mining, the system is engineered using things like taxonomy and lexical analysis to determine which part of the text document meets the criteria such as data to be generated

The purpose of text mining is to get useful information from a set of documents. Information retrieval must be done quickly and precisely [19]–[21] Thus, the data source used in text mining is a collection of text that has a format that is not structured or at least semi-structured. The specific tasks of text mining include categorization of text and text groupings. Text mining is the application of data mining concepts and techniques to find patterns in the text, the process of analyzing text to extract useful information for a particular purpose.

2.2. Similarity

Data security needs to be improved by checking similar documents [22]–[26] Document similarity measure is a technique for measuring the level of similarity of documents. It is always used in the academic world. The degree of similarity of documents can be arranged using a decision support system to find out which documents are eligible to use [27]. This measurement has several methods, such as:

- Distance-based similarity measure measures the similarity of two objects regarding the geometric distance of the variables included in the two objects. These are several methods of Distance-based similarity, Minkowski, Manhattan, Euclidean, Jaccard Distance, Dice's Coefficient, Cosine Similarity, Levenshtein, Hamming, and Soundex Distances.
- Feature-based similarity measure calculates the level of similarity by representing the object into the feature-feature form. It is used in classification or pattern matching for images and text.
- Probabilistic-based similarity measure calculates the similarity level of two objects by representing two sets of objects compared. For examples, Kullback Leibler Distance and Posterior Probability are two of this method.

Measuring similarity and distance between two information entities is a critical requirement in all cases of information retrieval. Plagiarism detection system is one of the application of document similarity measure.

2.3. Rabin-karp

Rabin Karp algorithm is a string matching algorithm that uses a hash function in performing its task [8]. This algorithm uses hashing to search for one of several sets of string patterns in a text. This algorithm is commonly used to track the practice of plagiarism on media containing text such as papers or articles. The string will be converted to a token, and it will be converted to an integer value called a hash value. This algorithm has the principle that if two strings are the same, then the hash value of the two strings must be the same, so the algorithm will only look for strings that have the same hash value as the value of the string being searched.
The algorithm has a weakness caused by the principle that all strings to look for are considered to have the same hash value as the obtained string [9]. It only compares the hash value of both then it will often happen the case where the string obtained is not a search string but hash value the same one. Fortunately, this avoidance can be combined with the Levenshtein algorithm or add a function to compare matches between strings.

2.4. Levenshtein

Levenshtein Distance algorithm is one of the algorithms used for decision making [28]–[30] This algorithm is often used to measure the distance of words or to determine the similarity level of two words. The auto-correction feature also uses Levenshtein algorithm. It is often seen in search engines on the internet. For example, when an internet user wants to type a word such as "journals," while typing is still at "jour," search engines have suggested "journal," "journals" or "journalist" as the word to search [18][31].
This algorithm calculates the minimum amount of transforming a string into another string that includes replacement, deletion, and insertion [32]–[34]. This algorithm is used to optimize the search because it is very inefficient if it searches every combination of the operation of the string. Therefore, the algorithm is classified as a dynamic program in search of such minimal value [35]. In this algorithm, the second length of the string is selected. If one or both strings is an empty string, the path of this algorithm stops and provides a zero-length edit distance or a non-empty string length. If the length of the string is both non-zero, the algorithm returns the distance of both strings.

3. Methodology

3.1. Analysis

This section is the process of analyzing the Rabin-Karp algorithm. The Rabin-Karp process is not done in its entirety. The Levenshtein process replaces the identical hash value calculation. Rabin-Karp work takes only hash values but does not specify the same hash value between the two documents. The stages performed by both algorithms in designing this system are as follows:

- Preprocessing: Tokenizing, Filtering, and Stemming.
- Divide the word into the form of n-gram, where the value of n is the length value of the word piece.
- Calculate the hash value of each n-gram with a predetermined formula.
- Find the distance using Levenshtein algorithm.

After the Rabin-Karp algorithm does the hash calculation process, the following step is the part to determine the similarity of the two hash value lists. Each document will generate a row of hash values and so will for the next document. These values will be determined the distance of each of the first document with the second document to obtain distance values using the Levenshtein algorithm. The level of similarity is obtained based on this value.

3.2. Hash value in levenshtein

The Levenshtein algorithm requires two parameters, the first string, and the second string. This algorithm replaces the position of the string with the hash value generated by the Rabin-Karp algorithm. Hash is an integer value that will be compared to both documents. The document similarity will be calculated based on the distance obtained by the Levenshtein algorithm. The following table is an example of using the Levenshtein algorithm.

Table 3.1: Hash Calculation of Levenshtein

		X_1	X_2	X_3	X_4	X_m
	0	1	2	3	4	m
Y_1	1					
Y_2	2					
Y_3	3					
Y_4	4					
Y_5	5					
Y_n	n					

X is the first document hash value represented horizontally in the above table while Y is the second hash value represented vertically of the document. The total number of the first hash is "m" and the second hash is "n."

4. Result and discussion

This section is a test of two strings. The tested string is a short sentence so that the formation process of N-Gram and hash values are well-defined and clear. The following illustration is an example of similarity testing between two sentences.
First: tiger eats meat
Second: cat eats mouse
Both sentences above will undergo the preprocessing process. Table 4.1 is the result of the process. In the table, the word "eats" returns to the basic form "eat."

Table 4.1: Praprocessing Result

First	Second
tiger	cat
eat	eat
meat	mouse

The resulting words will be included in the Rabin-Karp algorithm. There are three parameters as a determinant of the performance of this algorithm. The data below are the values of those parameters.
N-Gram : 3
Basis: 10
Modulo: 10007
The following table lists the results of the N-Gram formation process in both word lines.

Table 4.2: N-Gram Formation

First	Second
tig	cat
ige	ate
ger	tea
ere	eat
rea	atm
eat	tmo
atm	mou
tme	ous
mea	use
eat	

In the first string, ten words have been formed with N-Gram=3 while in the second string, the number of words is 19 words. The hash value is determined based on the word that has been formed in N-Gram format by using both Base and Modulo parameters. The following is a calculation for obtaining hash values for each word.

// ======== Hash “tiger eats meat”

KGram = (0)

Hash = $(116*10^2) + (105*10^1) + (103*10^0)$

Hash = 12753 Mod 10007

Hash = 2746

KGram = (1)

Hash = $(105*10^2)$ + (103*10^1) + (101*10^0)

Hash = 11631 Mod 10007

Hash = 1624

KGram = (2)

Hash = $(103*10^2)$ + (101*10^1) + (114*10^0)

Hash = 11424 Mod 10007

Hash = 1417

KGram = (3)

Hash = $(101*10^2)$ + (114*10^1) + (101*10^0)

Hash = 11341 Mod 10007

Hash = 1334

KGram = (4)

Hash = $(114*10^2)$ + (101*10^1) + (97*10^0)

Hash = 12507 Mod 10007

Hash = 2500

KGram = (5)

Hash = $(101*10^2)$ + (97*10^1) + (116*10^0)

Hash = 11186 Mod 10007

Hash = 1179

KGram = (6)

Hash = $(97*10^2)$ + (116*10^1) + (109*10^0)

Hash = 10969 Mod 10007

Hash = 962

KGram = (7)

Hash = $(116*10^2)$ + (109*10^1) + (101*10^0)

Hash = 12791 Mod 10007

Hash = 2784

KGram = (8)

Hash = $(109*10^2)$ + (101*10^1) + (97*10^0)

Hash = 12007 Mod 10007

Hash = 2000

KGram = (9)

Hash = $(101*10^2)$ + (97*10^1) + (116*10^0)

Hash = 11186 Mod 10007

Hash = 1179

// ======== Hash “cat eats mouse”

KGram = (0)

Hash = $(99*10^2) + (97*10^1)$ + (116*10^0)

Hash = 10986 Mod 10007

Hash = 979

KGram = (1)

Hash = $(97*10^2) + (116*10^1)$ + (101*10^0)

Hash = 10961 Mod 10007

Hash = 954

KGram = (2)

Hash = $(116*10^2) + (101*10^1)$ + (97*10^0)

Hash = 12707 Mod 10007

Hash = 2700

KGram = (3)

Hash = (101*10^2) + (97*10^1) + (116*10^0)

Hash = 11186 Mod 10007

Hash = 1179

KGram = (4)

Hash = (97*10^2) + (116*10^1) + (109*10^0)

Hash = 10969 Mod 10007

Hash = 962

KGram = (5)

Hash = (116*10^2) + (109*10^1) + (111*10^0)

Hash = 12801 Mod 10007

Hash = 2794

KGram = (6)

Hash = (109*10^2) + (111*10^1) + (117*10^0)

Hash = 12127 Mod 10007

Hash = 2120

KGram = (7)

Hash = (111*10^2) + (117*10^1) + (115*10^0)

Hash = 12385 Mod 10007

Hash = 2378

KGram = [8]

Hash = [117*10^2] + [115*10^1] + [101*10^0]

Hash = 12951 Mod 10007

Hash = 2944

After calculating the N-Gram and hash values, the following table shows the comparison of the result of the hash value between the first and the second string.

Table 4.3: Hash Value

First	Second
2746	979
1624	954
1417	2700
1334	1179
2500	962
1179	2794
962	2120
2784	2378
2000	2944
1179	

Rabin-Karp Calculation:

Similar Hash=962 and 1179

Similarity=$\frac{2*2}{10+9}*100\%=\frac{4}{19}*100\%=0.210526*100\%=21.05$

Levenshtein Calculation:

The following table shows the results of Levenshtein algorithm calculations on the hash values obtained from the Rabin-Karp algorithm.

Table 4.3: Levenshtein Distance of Hash Value

		2746	1624	1417	1334	2500	1179	962	2784	2000	1179
	0	1	2	3	4	5	6	7	8	9	10
979	1	1	2	3	4	5	6	7	8	9	10
954	2	2	2	3	4	5	6	7	8	9	10
2700	3	3	3	3	4	5	6	7	8	9	10
1179	4	4	4	4	4	5	5	6	7	8	9
962	5	5	5	5	5	5	6	5	6	7	8
2794	6	6	6	6	6	6	6	6	6	7	8
2120	7	7	7	7	7	7	7	7	7	7	8
2378	8	8	8	8	8	8	8	8	8	8	8
2944	9	9	9	9	9	9	9	9	9	9	9

Significant differences occur in the results of both algorithms. Rabin-Karp produces 21.05% while Rabin-Karp and Levenshtein combinations produce 10% of document similarity. Accuracy will be better if the combination is done. After calculating the N-Gram and hash values, the following table shows the comparison of the result of the hash value between the first and the second string.

Table.4.4: Hash value

First	Second
2746	979
1624	954
1417	2700
1334	1179
2500	962
1179	2794
962	2120
2784	2378
2000	2944
1179	

Rabin-Karp Calculation:

Similar Hash=962 and 1179

Similarity=$\frac{2*2}{10+9}*100\%=\frac{4}{19}*100\%=0.210526*100\%=21.05\%$

The following table shows the results of Levenshtein algorithm calculations on the hash values obtained from the Rabin-Karp algorithm.

Levenshtein Calculation:

Similar Hash=962 and 1179

Similarity=$1-\left(\frac{9}{10}\right)*100\%=\left(\frac{1}{10}\right)*100\%=0.1*100\%=10\%$

Significant differences occur in the results of both algorithms. Rabin-Karp produces 21.05% while Rabin-Karp and Levenshtein combinations produce 10% of document similarity. Accuracy will be better if the combination is done.

5. Conclusion

The combination of Rabin-Karp and Levenshtein algorithms makes a useful contribution. The level of similarity can be optimized so that documents can be assessed correctly according to the content of the document. This combination can also improve the accuracy of reviewers based on some parameter, N-Gram, Base, and Modulo. This parameter can be adjusted according to the desired sensitivity level of the algorithm. The higher the N-Gram value, the lower the sensitivity level. The lower Base and Modulo values, the lower the test sensitivity level.

References

[1] A. P. U. Siahaan and R. Rahim, "Dynamic Key Matrix of Hill Cipher Using Genetic Algorithm," Int. J. Secur. It is Appl., vol. 10, no. 8, pp. 173–180, Aug. 2016.

[2] R. Meiyanti, A. Subandi, N. Fuqara, M. A. Budiman, and A. P. U. Siahaan, "The Recognition of Female Voice Based on Voice Registers in Singing Techniques in Real-Time using Hankel Transform Method and Macdonald Function," J. Phys. Conf. Ser., vol. 978, no. 1, pp. 1–6, 2018. https://doi.org/10.1088/1742-6596/978/1/012051.

[3] R. Rahim et al., "Combination Base64 Algorithm and EOF Technique for Steganography," J. Phys. Conf. Ser., vol. 1007, no. 1, pp. 1–5, 2018. https://doi.org/10.1088/1742-6596/1007/1/012003.

[4] Z. Tharo, A. P. U. Siahaan, and N. Evalina, "Improvisation Analysis of Reactive Power Energy Saving Lamps Based on Inverter," Int. J. Eng. Tech., vol. 2, no. 5, pp. 141–145, 2016.

[5] S. Aryza, M. Irwanto, Z. Lubis, A. P. U. Siahaan, R. Rahim, and M. Furqan, "A Novelty Design of Minimization of Electrical Losses in A Vector Controlled Induction Machine Drive," in IOP Conference Series: Materials Science and Engineering, 2018, vol. 300, no. 1.

[6] Z. Ramadhan and A. P. U. Siahaan, "Stop-and-Wait ARQ Technique for Repairing Frame and Acknowledgment Transmission," Inte rnational J. Eng. Trends Technol., vol. 38, no. 7, pp. 384–387, 2016. https://doi.org/10.14445/22315381/IJETT-V38P269.

[7] S. Ramadhani, Y. M. Saragih, R. Rahim, and A. P. U. Siahaan, "Post-Genesis Digital Forensics Investigation," Int. J. Sci. Res. Sci. Technol., vol. 3, no. 6, pp. 164–166, 2017.

[8] A. P. U. Siahaan, "Rabin-Karp Elaboration in Comparing Pattern Based on Hash Data," Int. J. Secur. It is Appl., vol. 12, no. 2, pp. 59–66, Mar. 2018.

[9] A. P. U. Siahaan, Mesran, R. Rahim, and D. Siregar, "K-Gram As A Determinant Of Plagiarism Level In Rabin-Karp Algorithm," Int. J. Sci. Technol. Res., vol. 6, no. 7, pp. 350–353, 2017.

[10] L. Marlina, Muslim, and A. P. U. Siahaan, "Data Mining Classification Comparison (Naïve Bayes and C4.5 Algorithms)," Inte rnational J. Eng. Trends Technol., vol. 38, no. 7, pp. 380–383, 2016. https://doi.org/10.14445/22315381/IJETT-V38P268.

[11] W. Fitriani and A. P. U. Siahaan, "Comparison between WEKA and Salford System in Data Mining Software," Int. J. Mob. Comput. Appl., vol. 3, no. 4, pp. 1–4, 2016.

[12] Suherman and A. P. U. Siahaan, "Huffman Text Compression Technique," Int. J. Comput. Sci. Enginee ring, vol. 3, no. 8, pp. 103–108, 2016.

[13] L. Marlina, A. P. U. Siahaan, H. Kurniawan, and I. Sulistianingsih, "Data Compression Using Elias Delta Code," Int. J. Recent Trends Eng. Res., vol. 3, no. 8, pp. 210–217, Aug. 2017. https://doi.org/10.23883/IJRTER.2017.3406.TEGS6.

[14] Mochammad Iswan Perangin-angin, Khairul, and A. P. U. Siahaan, "Fuzzy Logic Concept in Technology, Society, and Economy Areas in Predicting Smart City," Int. J. Recent Trends Eng. Res., vol. 2, no. 12, pp. 176–181, 2016.

[15] R. F. Wijaya, Y. M. Tondang, and A. P. U. Siahaan, "Take Off and Landing Prediction using Fuzzy Logic," Int. J. Recent Trends Eng. Res., vol. 2, no. 12, pp. 127–134, 2016.

[16] M. I. Perangin-angin, Muslim, and A. P. U. Siahaan, "Frontline Personnel Knowledge Sharing and Transfer (PT. Wahana Ottomitra Multiartha, Tbk.)," Int. J. Comput. Sci. Enginee ring, vol. 3, no. 8, pp. 109–112, 2016.

[17] I. Sumartono, D. Arisandi, A. P. U. Siahaan, and Mesran, "Expert System of Catfish Disease Determinants Using Certainty Factor Method," Int. J. Recent Trends Eng. Res., vol. 3, no. 8, pp. 202–209, Aug. 2017. https://doi.org/10.23883/IJRTER.2017.3405.TCYZ2.

[18] R. Rahim et al., "Searching Process with Raita Algorithm and its Application," J. Phys. Conf. Ser., vol. 1007, no. 1, pp. 1–7, 2018. https://doi.org/10.1088/1742-6596/1007/1/012004.

[19] S. Hartanto, M. Furqan, A. P. U. Siahaan, and W. Fitriani, "Haversine Method in Looking for the Nearest Masjid," Int. J. Recent Trends Eng. Res., vol. 3, no. 8, pp. 187–195, Aug. 2017. https://doi.org/10.23883/IJRTER.2017.3402.PD61H.

[20] A. P. U. Siahaan, "Heuristic Function Influence to the Global Optimum Value in Shortest Path Problem," IOSR J. Comput. Eng., vol. 18, no. 05, pp. 39–48, May 2016. https://doi.org/10.9790/0661-1805053948.

[21] Z. Sitorus and A. P. U. Siahaan, "Heuristic Programming in Scheduling Problem Using A* Algorithm," IOSR J. Comput. Eng., vol. 18, no. 5, pp. 71–79, 2016.

[22] M. Saragih, H. Aspan, and A. P. U. Siahaan, "Violations of Cybercrime and the Strength of Jurisdiction in Indonesia," Int. J. Humanit. Soc. Stud., vol. 5, no. 12, pp. 209–214, 2017.

[23] D. Kurnia, H. Dafitri, and A. P. U. Siahaan, "RSA 32-bit Implementation Technique," Int. J. Recent Trends Eng. Res., vol. 3, no. 7, pp. 279–284, 2017. https://doi.org/10.23883/IJRTER.2017.3359.UX-AIW.

[24] Haryanto, A. P. U. Siahaan, R. Rahim, and Mesran, "Internet Protocol Security as the Network Cryptography System," Int. J. Sci. Res. Sci. Technol., vol. 3, no. 6, pp. 223–226, 2017.

[25] W. Fitriani, R. Rahim, B. Oktaviana, and A. P. U. Siahaan, "Vernam Encypted Text in End of File Hiding Steganography Technique," Int. J. Recent Trends Eng. Res., vol. 3, no. 7, pp. 214–219, Jul. 2017. https://doi.org/10.23883/IJRTER.2017.3351.6ON8H.

[26] A. P. U. Siahaan, "Vernam Conjugated Manipulation of Bit-Plane Complexity Segmentation."

[27] Z. Tharo and A. P. U. Siahaan, "Profile Matching in Solving Rank Problem," IOSR J. Electron. Commun. Eng., vol. 11, no. 05, pp. 73–76, May 2016. https://doi.org/10.9790/2834-1105017376.

[28] A. P. U. Siahaan, "Rail Fence Cryptography in Securing Information."

[29] A. P. U. Siahaan, "A Fingerprint Pattern Approach to Hill Cipher Implementation."

[30] R. Rahim, Mesran, A. P. U. Siahaan, and S. Aryza, "Composite Performance Index for Student Admission," Int. J. Res. Sci. Eng., vol. 3, no. 3, pp. 68–74, 2017.

[31] Khairul, M. Simaremare, and A. P. U. Siahaan, "Decision Support System in Selecting the Appropriate Laptop Using Simple Additive Weighting," Int. J. Recent Trends Eng. Res., vol. 2, no. 12, pp. 215–222, 2016.

[32] T. Ho, S.-R. Oh, and H. Kim, "A parallel approximate string matching under Levenshtein distance on graphics processing units using warp-shuffle operations," PLoS One, vol. 12, no. 10, p. e0186251, Oct. 2017. https://doi.org/10.1371/journal.pone.0186251.

[33] C. Nyirarugira, H.-R. Choi, J. Kim, M. Hayes, and T. Kim, "Modified levenshtein distance for real-time gesture recognition," in 2013 sixth International Congress on Image and Signal Processing (CISP), 2013, pp. 974–979.

[34] R. Umar, Y. Hendriana, and E. Budiyono, "Implementation of Levenshtein Distance Algorithm for ECommerce of Bravoisitees Distro," Int. J. Comput. Trends Technol., vol. 27, no. 3, pp. 131–136, Sep. 2015. https://doi.org/10.14445/22312803/IJCTT-V27P123.

[35] L. Yujian and L. Bo, "A Normalized Levenshtein Distance Metric," IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, no. 6, pp. 1091–1095, Jun. 2007. https://doi.org/10.1109/TPAMI.2007.1078.