Enhancing Levenshtein's Edit Distance Algorithm for Evaluating Document Similarity

Shama Rani and Jaiteg Singh(✉)

Chitkara University, Chandigarh, India
shamajhansla@gmail.com, Jaiteg.singh@chitkara.edu.in

Abstract. The content directly taken from pre-published sources is called plagiarized text. Plagiarism is considered to be a major challenge in contemporary research manuscripts. It is very easy to use internet as a source of information. There is a need to find suitable technique, so as to find similarity between two documents. Though there are several methods for text comparison, yet this paper is primary focused on Levenshtein's edit distance. It is a string metric, which is an effective technique for comparing text documents and for calculating efforts required to transforms one document to another. An effort has been made to improve the performance of Levenshtein's edit distance algorithm by eliminating stop words while calculating transformation effort.

Keywords: Comparison of documents · Levenshtein's edit distance
Modified Levenshtein's edit distance · Similarity between documents
Cost and time evaluation

1 Introduction

Plagiarism is defined as the use of another's thoughts, literature, and information, when done without proper citation of the original source. Plagiarism for the text documents occurs in different ways. Plagiarized text may be copied from one-to-one passages may be modified to a larger or reduced extent or text may be interpreted [15]. Data Comparison relates to the methods of calculating differences and similarities so to replace the strings and data objects. The objects that are compared usually program code, algorithms, computer files, text versions, or complex data structures [16].

To detect plagiarism in software presents some problems due to the nature of programming. The reasons for similarity between the programs can be categorized in different categories, one of which is plagiarism. Similarities including metric, textual, features in depth and some recommendations are made for measures of syntax and semantics, program execution, input-output, shared information, program dependency graph similarity [3, 4].

There are two most probable features used to compare documents are: importing one single file for online plagiarism check, or matching two file for a comparative check. It follows the following steps:

Step1. The text is exported from the file with ignorance of images and other diagrams.

Step 2. The text is divided into n –grams or sets of words.

R. Sharma et al. (Eds.): ICAN 2017, CCIS 805, pp. 72–80, 2018.
https://doi.org/10.1007/978-981-13-0755-3_6

Step 3. Each group is searched by the software.

Step 4. The search engine results are stored and loaded on pages.

Step 5. The page is parsed when the website has been loaded to extract the text from the HTML code.

Step 6. The string is explored inside the mined text.

Step 7. If a matching sentence has been found, the input text is added to the source list and the next Sentence is starting to be analyzed.

Step 8. If the sentence has not been found, another website is loaded until results were analyzed. The two algorithms used for the detection of Text is Levenshtein's Edit distance method [5, 13].

2 Levenshtein's Edit Distance

The Levenshtein's distance between two documents is defined as the minimum number of edit operations required to transform one text document into the other. The following edit operations are used by Levenshtein's edit distance algorithm to modify one document to another:

- Insertion
- Deletion
- Substitution

Example: Levenshtein's edit Distance between different strings:
right → fight (substitution of 'f' for 'r')
book → books (insert operation is performed at the end 's')

The Levenshtein's edit Distance between given strings depends on three basic operations to replace one string to another.

The results of Levenshtein's distance is based on the perception that if a matrix holds the edit distance between all prefixes of the first string and all the prefixes of the second, Thus find the distance between the two full strings as the last value calculated [6, 10].

The algorithm:

Step 1: Initialization

I. Set a is the length of document 1 say d1, set b is length of document 2 say d2.
II. Create a matrix that consists 0 – b rows and 0 – a columns.
III. Initialize the first row from 0 to a.
IV. Initialize the first column from 0 to b.

Step2: Processing

I. Observe the value of d2 (i from 1 to a).
II. Observe the value of d1 (j from 1 to b).
III. If the value at d2[i] is equals to value at d1[j], the cost becomes 0.
IV. If the value at d2[i] does not equal d1[j], the cost becomes 1.
V. Set block of matrix M[d1, d2] of the matrix equal to the minimum of:
i. the block immediately above add 1: M [d2–1, d1]

ii. the block immediately to the left add 1: M [d2, d1–1] +1.
iii. The block is diagonally above and to the left adds the cost: M [d2–1, d1–1] + cost.

Step 2 is repeated till the distance M[a, b] value is found.
Step 3: Result [1, 5].

2.1 Computing Techniques

- Dis(i,j) = score of best alignment from d11..d1i to d21…..d2j
- Dis (i–1, j–1) + d (d1i, d2j) //copy Dis (i–1, j) + 1 //insert
- Dis (i, j–1) + 1 //delete

Cost depend upon following factors:

- Dis (0, 0) = 0 cost // if both strings are same
- Dis (i, 0) = dis (i–1, 0) +1 = 0 // if source string is empty
- Dis (0, j) = dis (0, j–1) +1 = 0 // if target string is empty [6, 7].

3 Modified Levenshtein's Edit Distance Algorithm

Levenshtein's edit distance algorithm can be modified by removing the stop words. The words such as also, is, am, are, they, them, their, was, were etc. are ignored by search engine during processing are called stop words [4, 8].

Most search engines are programmed to eliminate such words while indexing or retrieving as the outcome of search query. Stop words are considered inappropriate for searching purposes because they occur commonly in the language for which the indexing engine has been tuned [4]. These words are dropped in order to save both time and space at the time of searching in the text documents. The words which are often used are is, am, are, they, them, also, the, of, and, to, in which are insignificant in IR and text mining. Stop words are removed to reduce due to following reasons:

- Each documents approximately consists 20–25% stop words.
- Efficiency of document is improved by removing the stop words.
- Stop words are not useful for searching or text mining
- To reduce indexing [9].

4 Performance Analysis

Levenshtein's edit Distance is not considered as an absolute value. If the first string is 'Race' and the second string is 'spaces', it's very unlikely that one of them is misspelled. However, if the first string is 'I have a pet' and the second string is 'I have a cat', the second string is probably misspelled. But in both cases, the Levenshtein's Distance is 2. The first value means that 2/3 of the characters are different, the second value tells us that the difference is little. Another method to calculate Levenshtein's edit distance algorithm

is matrix method. The Levenshtein's Edit Distance algorithm calculates the minimum edit operations that are needed to modify one document to obtain second document. A matrix is initialized measuring in the (m, n)-cell the Levenshtein's distance between the m-character prefix of one with the n-prefix of the other word [12, 13].

The following example will determine the use of Matrix method. Let the first string is PEON and second string is SPEND the minimum path is selected by comparing at each stage. The calculation process of the Levenshtein's distance between two strings of different length is based on the number of operations to transform first string to another and the edit distance between the substrings X1m = x1x2…xn and Ymn = y1 y2…yn is calculated as follows:

- $\mathrm{Dis(m,n)} = \mathrm{Dis(X1m, Y1n)}$
- $\mathrm{Dis(m,n)} = \min\{\mathrm{Dis(m-1,n)+1, Dis(m,n-1)+1, Dis(m-1,n-1)+Cost}\}$

$$\text{With } 0 \quad \text{if } \mathrm{X}\,m-1 = \mathrm{Y}n-1$$
$$1 \quad \text{else}$$

and the initializations are: $\mathrm{D(m,\varnothing)} = \mathrm{m}$ and $\mathrm{D(\varnothing,n)} = \mathrm{n}$, where $\varnothing$ represents the empty string [11].

In this way, we calculate the Levenshtein's distance between two strings is shown in lower right most block in Fig. 1 [13]. In above example there are different ways to replace "PEON" with "SPEND", but the minimum cost path is taken by this method is shown with arrows. The experimental details of Levenshtein's Edit distance in terms of space and time is follows. The inputs given in two Documents and number of words after removing stop words are calculated in all the documents by using Levenshtein's Edit distance formula is shown in Fig. 2. The time taken to calculate Levenshtein's distance with Stop words is shown in Fig. 3. The calculated time and cost represented in separately in view of easy understanding of graphs. The time taken to compare documents is calculated in milliseconds and later be converted into asymptomatic time by applying on some other algorithms.

		P	E	O	N
	0	1	2	3	4
S	**1**	1	2	3	4
P	**2**	1	2	3	4
E	**3**	2	1	2	3
N	**4**	3	2	2	2
D	**5**	4	3	3	3

Fig. 1. Edit distance between two strings

- D(m,n) = score of best alignment from s1..si to tm…….tn.

Here 51 Text length of Document A and B means the document size. The document size means it contains the defined number of words. The experiment is done by taking different document size from 50–1000.

Text Length of Document A	Text Length of Document B	Document A after removing stop words	Document B after removing stop words
51	62	27	38
103	90	59	53
203	192	124	119
395	410	242	233
798	750	470	474

Fig. 2. Text length of Document A and B with and without using stop words

Text Length of Document A	Text Length of Document B	Time taken to calculate Levenshtein's distance with Stop words (in milliseconds)
51	62	14
103	90	16
203	192	23
395	410	62
798	750	218

Fig. 3. Time taken to calculate Levenshtein's distance after removing stop words

The time taken to calculate Levenshtein's distance with Stop words is represented in Fig. 3. The Time taken to calculate Levenshtein's distance after removing Stop words is shown in Fig. 4.

The length of Document A and the length of same document after removing the stop words is shown in Fig. 5. Where case A1 represents the complete text length of Document A and case A2 represents the length of Document A after removing the stop words.

Similarly, in Fig. 6. case, B1 represents the complete text length of Document B and case B2 represents the length of Document B after removing the stop words. Of the Levenshtein's distance between the words "PEON" and "SPEND", the distance is three as shown in Fig. 1.

The matrix completes the blocks from the top most corner of left to the lower right corner. Each move vertically or horizontally corresponds to insertion or a deletion and substitution respectively. Each operation is initially set to costs.

1. The diagonal move costs one, if the two characters in the row and do not match and one if they do. The cost is locally minimizes by each block.

Text Length of Document A	Text Length of Document B	Time taken to calculate Levenshtein's distance after removing Stop words (in milliseconds)
51	62	7
103	90	5
203	192	12
395	410	30
798	750	119

Fig. 4. Time taken to calculate Levenshtein's distance after removing stop words

- case TW1 represent the time with stop words and TWO represents the time taken to calculate Levenshtein's distance without stop words.

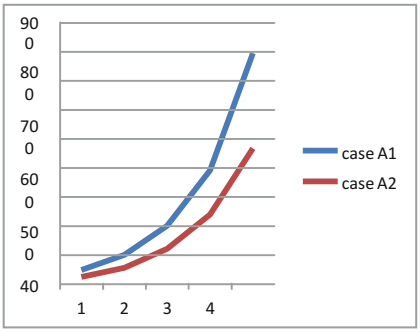

Fig. 5. Text length of document A with and without using stop words

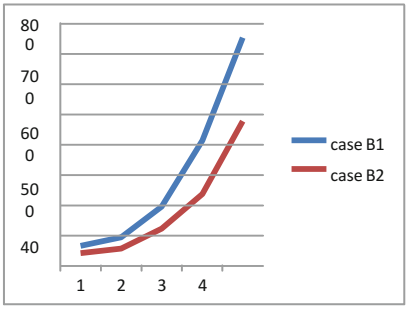

Fig. 6. Text length of document B with and without using stop words

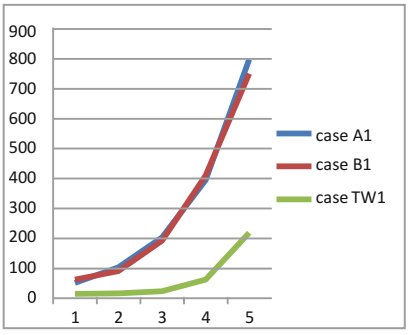

Fig. 7. Time taken to calculate Levenshtein's distance with stop words

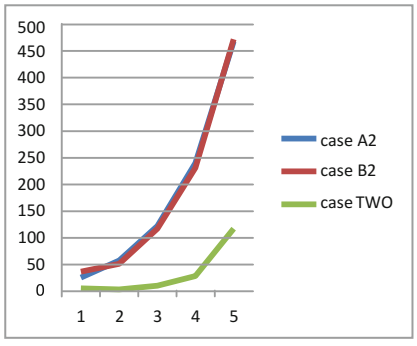

Fig. 8. Time taken to calculate Levenshtein's distance without stop words

The time taken to calculate Levenshtein's edit distance with Stop words is shown in Fig. 7. where Case A1 represents the complete text length of Document A and case B1 represents the complete text length of Document B. The case TW1 is the time taken by algorithm to compare the lengths of Document A and Document B.

The Fig. 8. represents the comparison of Documents after removing the Stop words. Case A2 signifies the length of Document A after removing the stop words, case B2 represents the length of Document B after removing the stop words and The final comparison of the times taken by Levenshtein's in different documents with and without stop words. In Fig. 9. case TW1 is time taken to calculate Levenshtein's distance with stop words and case TWO is time taken to calculate edit distance after removing the stop words in the text Documents.

The cost calculated by Levenshtein's Algorithm of two dissimilar documents is shown in Fig. 10.

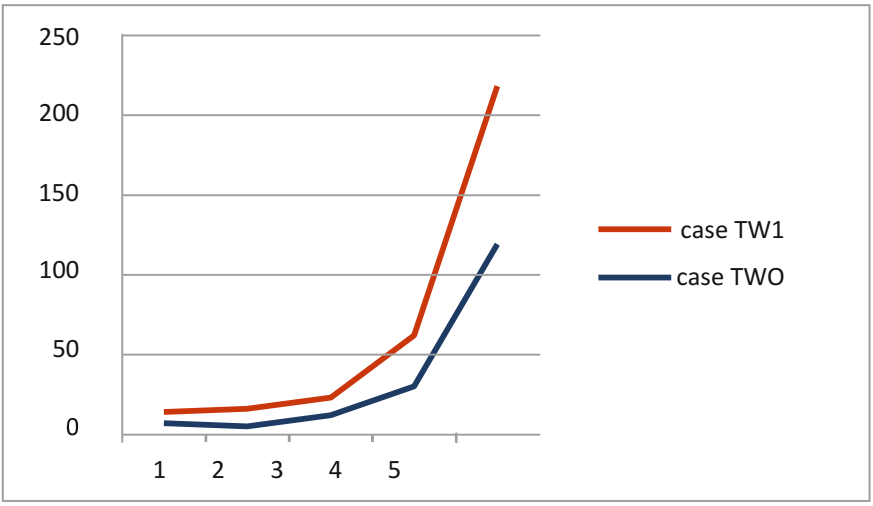

Fig. 9. Time taken to calculate before and after removing stopwords

Text length of Document A	Text length of Document B	Cost with Stopwords (in characters)	Cost after removing Stop words (in characters)
51	62	257	202
103	90	534	418
203	192	1011	794
395	410	1907	1469
798	750	2483	1859

Fig. 10. Cost to replace one document to another

5 Conclusion

The documents with different text length 50, 100, 200, 400, 800 is taken to calculate the Levenshtein edit distance and the time need to compare both documents by using Levenshtein edit distance algorithm. This is observed that each document consists 20–30% stop words, which are not useful for any calculation. Therefore, it is observed that if 20% stop words are removed from any text document, 50% time can be reduced to calculate the Levenshtein's edit distance. The Comparison of time with stop words and after removing stop words from the documents of different text length is shown in Fig. 9.

References

1. Gueddah, H., et al.: Introduction of the weight edition errors in the Levenshtein distance. IJARAI Int. J. Adv. Res. Artif. Intell. **1**(5) (2012)
2. Dang, Q.T.: Determining restricted Damerau-Levenshtein edit-distance of two languages by extended automata. In: International Conference on Computing and Communication
3. Burkhardt, S., Kärkkäinen, J.: One-Gapped q-Gram filters for Levenshtein distance. In: Apostolico, A., Takeda, M. (eds.) Combinatorial Pattern Matching. CPM 2002. Lecture Notes in Computer Science, vol. 2373, pp. 225–234. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45452-7_19
4. Hubert, C.: A contextual normalized edit distance. In: 2008 IEEE 24th International Conference on Source Data Engineering Workshop, ICDEW 2008 (2008)
5. Aouragh, S.I.: Adaptating Levenshtein distance to contextual spelling correction. Int. J. Comput. Sci. Appl. **12**(1), 127–133 (2015)
6. Danny, H.: A unified algorithm for accelerating edit distance computation via text compression. In: Symposium on Theoretical Aspects of Computer Science year (city)
7. Ndiaye, M., Faltin, A.V.: Correcteur Orthographique Adapté à Apprentissage du Français. Revue Bulag no. 29, pp. 117–134 (2004)
8. Mitton, R.: Ordering the suggestions of a spellchecker without using context. Nat. Lang. Eng. **15**(2), 173–192 (2009)
9. Damerau, F.J.: A technique for computer detection and correction of spelling errors. Commun. Assoc. Comput. Mach. **7**, 171–176 (1964)
10. Kobzdej, P.: Parallel application of Levenshtein's distance to establish similarity between strings. Front. Artif. Intell. Appl. **12**(4) (2003)
11. Yujian, L., Bo, L.: A normalized Levenshtein's distance metric. IEEE Trans. Pattern Anal. Mach. Intell. **29**(6), 1091–1095 (2007)
12. Haldar, R., Mukhopadhyay, D.: Levenshtein distance technique in dictionary lookup methods: an improved approach. Web Intell. Distrib. Comput. Res. Lab
13. Andoni, A.: Approximating edit distance in near linear time. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC 2009), Bethesda, MD, USA, pp. 199–204 (2009)
14. Oberreuter, G., Velasquez, J.D.: Expert system with applications: text mining applied to plagiarism detection. Web Intelligence Consortium Chile Research Centre, Department of Industrial Engineering, Chile (2013)
15. Kharat, R., et al.: Semantically detecting plagiarism for research papers. Int. J. Eng. Res. Appl. (IJERA) **3**(3), 077–080 (2013). ISSN: 2248-9622, https://www.ijera.com

16. Caroline, L., et al.: Plagiarism is Easy, But also Easy to Detect. MPublishing, University of Michigan Library, Ann Arbor, vol. 1 (2006)
17. Journal of technology management for growing economies (JTMGE) (2015). http://tmgejournal.com/abstract.php?id=513. Accessed 14 Jan 2017