ORIGINAL RESEARCH PAPER

Volume-8 | Issue-2 | February-2019 | PRINT ISSN No 2277 - 8179

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH

EMERGE TOWARDS LEVENSHTEIN'S DISTANCE ALGORITHM FOR ESTIMATION OF TEXTUAL SIMILARITIES

Computer Science

Neelesh Channawar Gondwana University, Gadchiroli, Maharashtra, India

Dr. S. B. Kishor Gondwana University, Gadchiroli, Maharashtra, India

ABSTRACT

In modern days Plagiarism is a severe offence continuously happening in research and Academic field. Plagiarism is nothing but a method of approving other thoughts and research and ideas without proper appreciation. In plagiarism one can mention and steal other hard work as its own idea and research which is a serious misdeed. The problem of plagiarism is increasing day by day because of the powerful search engines which are available and having all the resources and information regarding any concept. To avoid and detect plagiarism in text matching lot of detecting tools and algorithms are present in market. Levenshtein distance algorithm is a very famous and efficient algorithm among all the plagiarism detection technique. In this paper we will show the working function and ability to detect plagiarism in similar textual strings. Levenshtein distance is a very simple measure which can be an effective string approximation tool. Here we will provide a comprehensive overview about Levenshtein distance algorithm which is an efficient algorithm for detecting plagiarism. The tentative idea will illustrate that Levenshtein algorithm is useful in complex plagiarism detection.

KEYWORDS

Plagiarism, Levenshtein distance algorithm, Textual Plagiarism, similarity between documents, string matching

INTRODUCTION

Plagiarism is a very serious dilemma in literary environments and in education systems. Since everyone has access to the Internet, it is easy to use as a source of information. However, copies of files from the Internet can be considered as plagiarism: Anything found on the Internet can come from books, research or articles. This content can lead to serious legal issues, such as copyright infringement. Many types of software were searched. Anti-Theft software has many types of information, and you can use the database such as articles, books or research, internet, or file contact files. The advent of the digital age has greatly increased the number of digital resources available on the global Internet. Creating such digital resources can now be quickly and easily stored and distributed. The rapid development of such digital companies has increased the possibility of copyright and theft. To solve this problem, researchers have been experimenting with various language thefts since 1990, starting with the digital literature recognition system [1]. However, the identification process was initiated in the 1970s to detect fraudulent software for each time [2]. However, many methods and tools are available online to determine the identity of a theft.

Plagiarism can be done from one source where someone will copy the text or reduce the text to maximum extent or plagiarism can be done from many sources where content will be taken in larger extent from different text [3]. Data Comparison is a process where we have to calculate the differences and similarities in different strings so as to remove the strings and data objects. Objects relate to program code, algorithms, computer files, text versions, or complex data structures [4]. Some problems arise in the software for finding plagiarism in articles is due to some programming concepts. The reasons for the similarity between programs can be classified as follows: One of the strong reasons is plagiarism. Appearance includes metrics, text, detailed features and some grammar suggestions and Semantics, program execution, input and output, share information, program dependency and graphical resemblance[5][6][7].

Many applications include spelling [8] when determining string similarity. Search dialects [9], the accuracy of pronunciation and affinity between DNA analysis Structure [10] or Web Mining [11], to name a few. There are many ways to do this Specifies the existence of a string, such as a string, in another string. Knuth-Morris-Pratt [12] or Boyer Moore's algorithm [13]. However, it is often necessary to find a fact as another substring, but a measure of similarity between them is a standard example. This situation is the case of plagiarism detection, in which plagiarized text is susceptible some words have been changed, other words have been deleted, other words have been exchanged, and so on. In this case, the algorithm is like Rabin-Karp [14] did not meet the required results. Therefore, a different approach is needed and some form of approximate string matching must be applied. It is worth mentioning that there are three related issues: (a) a string that conforms to the "worry free" symbol, and (b) a string Match k does not match and (c) matches k difference. The first one contains one Matches "any" Meta characters, the second allows matching up to k characters. In other words, the third required mode and the text work distance is less than or equal to k [15, 16, 17].

There are two ways present to compare documents like: introducing one single file for online plagiarism check and the second one is matching two file for a comparative check. Following steps are [7]:

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.
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 analysed.
Step 8. If the sentence has not been found, another website is loaded until results were analysed. The two algorithms used for the detection of Text is Levenshtein's distance method [18, 19].

II. LEVENSHTEIN DISTANCE

Plagiarism detection method is only one way to secure the plagiarised text in a research paper or in a article or in a book. Levenshtein distance is a very efficient detection method. The name of the Levenshtein distance algorithm comes from the Russian scientist Vladimir Levenshtein. The Levenshtein distance algorithm is a very popular high performance plagiarism detection algorithm for computer science. Today, the algorithm is primarily used to search for plagiarized text in required fields.

The algorithm can be defined as a measure of the similarity between two strings, called source string(s) and destination string (t). Spacing is the number of deletions, inserts, or substitutions required to convert s to t [20]. As the Levenshtein distance increases, the strings became more different. We can take the source string is the input, and the target string is one of the entries in the given text.

In Levenshtein's distance, edit operations required to show the distance between two documents which show the minimum distance to transform one text document into the other. The following edit operations are used by Levenshtein's distance algorithm to modify one document into another [7], the operations are-

a. Insertion
b. Deletion
c. Substitution

The Levenshtein's distance between given strings depend on three basic operations to replace one string to another.

Below are the examples: Suppose any two strings are given
I. Sight - tight (substitution of't' for's')
ii. Temple -Temples (insert operation introducing 's' at end)
iii. God- Go (Deletion of 'd' at the end)

The three main conditions in Levenshtein distance algorithm are as follow [20],
I. The cell immediately above plus 1: d[i-1, j]+1
II. The cell immediately to the left plus 1: d[i, j-1]+1
III. The cell diagonally above and to the left plus the cost: d [i-1, j-1] +cost.

The Levenshitan distance is evaluated by the similarity between the source and destination chains. The basic concepts of Levensatin Distance are widely used in computer science, mathematical linguistics, biological sources, molecular biology, DNA analysis and other fields. It can be used to measure the similarities of music or music. The elimination of Levenshtein is common in our daily lives. Levenshtein removes or removes spell checking or editing to repair a program or application. Another possible use of Levenshatan distance is the recognition of language and the identification of plagiarism.

III LEVENSHTEIN DISTANCE ALGORITHM

Levenshtein distance algorithm is a very efficient plagiarism detection method. Levenshtein is common in our daily lives. Levenshtein removes or removes spell checking or editing to repair a program or application. The algorithm can be defined as a measure of the similarity between two strings. We can take the source string is the input, and the target string is one of the entries in the given text.

The Levenshtein distance algorithm is as follows-
Let there are two strings named as s and t. 's' is the source string and 't'is the destination string[21].

Step 1: Initialization
a) Set n to be the length of s, set m to be the length of t.
b) Construct a matrix containing 0..m rows and 0..n columns.
c) Initialize the first row to 0..n,
d) Initialize the first column to 0..m.

Step2: Processing
a) Examine s (i from 1 to n).
b) Examine t (j from 1 to m).
c) If s[i] equals t[j], the cost is 0. d) If s[i] doesn't equal t[j], the cost is 1.
e) Set cell d[i,j] of the matrix equal to the minimum of:
i) The cell immediately above plus 1: d[i-1,j] + 1.
ii) The cell immediately to the left plus 1: d[i,j-1] + 1.
iii) The cell diagonally above and to the left plus the cost: d[i-1,j-1] + cost.

Step 3: Result
Step 4: Stop
Step 2 is repeated till the d[n,m] value is found

IV ISSUES AND CHALLENGES

Based on our survey, we have found that over the past two decades, various methods and tools have been developed to enable rapid and accurate detection of plagiarism. Levenshtein distance algorithm solved the following major problems: (i) excellent syntax and semantic feature extraction, (ii) monolingual and multilingual plagiarism detection processing, and (iii) inclusion of plagiarism into sources of text and program data. Or does not contain a quoted code. However, with the rapid development of digital technology to support replication, storage, and distribution, some important issues and research challenges remain unattended. In this section, we will examine some of the issues and challenges that computer and language researchers face [22].

1. There is another way to find textual data and source code to ensure proof of accuracy and all content. Therefore, this is an important topic.
2. At present, there is no guarantee that the proximity metric of the plagiarized text segment is detected with great accuracy in intrinsic and extrinsic detection frames.
3. Developing a multi-language plagiarism checker that works without external references but guarantees high precision is a daunting task.
4. Developing a repository based on references to author fingerprint management is complete and accurate.
5. Developing a plagiarized inspector to accept user comments and generate detailed plagiarism reports (similarity detection between 1% and 99%) and the right source is a major issue.

CONCLUSION

Plagiarism in the texts is a crime. The university community is now the most common area where plagiarism often takes place. Most of the time, plagiarism occurs on common features like inserting, deleting or synonyms of a word. However, this type of work substitution requires a lot of string comparison. In this research paper we provide a broad overview of the levenshtein distance algorithm for plagiarism. We show the working form of algorithm existed in textual data and source code. Despite the introduction of many methods and tools over the past two decades; we believe that there are still many problems and challenges that need to be addressed. Finally, we present a series of studies aimed at developing complete and correct proofreaders for monolingual and multilingual text data as well as source code.

REFERENCES

1. S. Brin, J. Davis, H. Garcia-Molina, Copy detection mechanisms for digital documents, in: ACM SIGMOD Record, Vol. 24, ACM, 1995, pp. 398-409. 24 Parker, et al., Computer algorithms for plagiarism detection.
2. Kharat, R., et al.: Semantically detecting plagiarism for research papers. Int. J. Eng. Res.
3. Caroline, L., et al.: Plagiarism is Easy, But also Easy to Detect. MPublishing, University of
4. Burkhardt, S., Kärkkäinen, J.: One-Gapped q-Gram filters for Levenshtein distance. In:Apostolico, A., Takeda, M. (eds.) Combinatorial Pattern Matching. CPM 2002. LectureNotes in Computer Science, vol. 2373, pp. 225–234.Springer,Heidelberg (2002). https://doi.org/10.1007/3-540-45452-7_19
5. Hubert, C.: A contextual normalized edit distance. In: 2008 IEEE 24th International Conference on Source Data Engineering Workshop, ICDEW 2008 (2008).
6. Shama Rani and Jaiteg Singh, Enhancing Levenshtein's edit Distance Algorithm for Evaluating Document similarity, ICAN 2017, CCIS 805, pp. 72–80, 2018. https://doi.org/10.1007/978-981-13-0755-3_6.
7. M. Wypych (2002) Stochastic Spelling Correction of Texts in Polish, Institute of Linguistics, Adam Mickiewicz University, Poznań, Poland; "Speech and Language Technology. Volume 6", Poznań.
8. J8. Nerbonne, W. J. Heeringa, P. Kleiweg (1999) Edit distance and Dialect Proximity, in: D. Sankoff and J. Kruskal (eds.), Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequency Comparison, CSLI, Standford, pp. v-xv.
9. http://www2.toki.or.id/book/AlgDesignManual/BOOK/BOOK5/NODE204.HTM. Scherbina, Application of Levenstein Metric to Web sage Mining, Institute for System Programming, Russian Academy of Science, Proceedings of the 7th BIS Conference, Poznan University of Economics Press, to appear.
10. D. E. Knuth, J. H. Morris Jr., V. R. Pratt (1977) Fast pattern matching in strings, SIAM J. Comput., 6(1), 323-350.
11. R. S. Boyer, J. S. Moore (1977) A fast string searching algorithm, CACM, 20, 762-772.
12. R. M. Karp, M. O. Rabin (1987) Efficient randomized pattern-matching algorithms, IBM J. Res. Dev., 31(2), 249-260. H. Wright, Approximate String Matching using Within-Word Parallelism, Computer Science University of Montana.
13. http://www.cs.mu.oz.au/~mjl/thesis/node17.html
14. http://www2.toki.or.id/book/AlgDesignManual/BOOK/BOOK2/NODE46.HTM.
15. Aouragh, S.I.: Adaptating Levenshtein distance to contextual spelling correction. Int. J. Comput. Sci. Appl. 12(1), 127–133 (2015).
16. Andoni, A.: Approximating edit distance in near linear time. In: Proceedings of the 41stAnnual ACM Symposium on Theory of Computing (STOC 2009), Bethesda, MD, USA, pp. 199–204 (2009).
17. Nikhil Ghode, Shubham Jadhav, Sampada Moon, Ashmina Khan, Shrutika Bhalkar ,Detecting Plagiarism In Academics Using Levenshtein Distance Algorithm And Semantic Similarity, International Journal on Future Revolution in Computer Science & Communication Engineering ISSN: 2454-4248 Volume: 4 Issue: 3,pp. 471-473
18. Rishin Haldar and Debajyoti Mukhopadhyay, Levenshtein Distance Technique in Dictionary Lookup Methods: An ImprovedApproach,https:// www.researchgate.net/ publication/48180222.
19. Hussain A Chowdhury , Dhruba K Bhattacharyya, Plagiarism: Taxonomy, Tools and Detection Techniques.