DOI: 10.21917/ijsc.2018.0242

A PARTIAL RATIO AND RATIO BASED FUZZY-WUZZY PROCEDURE FOR CHARACTERISTIC MINING OF MATHEMATICAL FORMULAS FROM DOCUMENTS

G. Appa Rao[1], G. Srinivas[2], K. Venkata Rao[3] and P.V.G.D. Prasad Reddy[4]
[1]*Department of Computer Science and Engineering, GITAM Institute of Technology, India*
[2]*Department of Information Technology, Anil Neerukonda Institute of Technology and Sciences, India*
[3,4]*Department of Computer Science and Systems Engineering, Andhra University, India*

Abstract

Retrieval of mathematical text from data is a key predicament in present circumstances. To achieve this, we have considered three different algorithms viz., Sequence matcher, Levenshtein Distance and Fuzzy-Wuzzy. Two different variants of Fuzzy-Wuzzy are found applicable to this study out of four variants. Performance of these variants in retrieving mathematical texts, is calculated using efficiency measure, sensitivity analysis and time series exploration. Fuzzy-Wuzzy partial ratio algorithm scored better over the other variants on efficiency measure and sensitivity analysis.

Keywords:

Sequence Matcher, Levenshtein Distance, Fuzzy-Wuzzy, Partial Ratio

1. INTRODUCTION

The importance in mining of time series has been improved in the present years as majority of the technical papers are in print with mathematical formulae with time series. Due to high dimensionality and high relationship between data time series is very complex. The search for associated mathematical expression by the researchers is operational in technological aspect and cannot be completed in effect with text based search engine excluding apposite text keywords are known. The search for mathematical formulae is important and delicate as they restrain both constitutional and interpretation information. In general, the theoretical basis of the knowledge in numerous technical documents is generally represented with mathematical formulae. The established search engines Google and Yahoo works competently for text based information. But they are besieged in searching data with mathematical formulae.

Let us consider an example $\cos x+e^{\sin x}$, this formula includes three symbols *e*, sin and cos indicates exponential and trigonometric functions. In this equation the sin and cos terms contains some semantic meaning and cos and sin terms are structurally connected to exponential.

As already fairly a few schemes were anticipated for the extraction of mathematical formulae, in this paper we intended a new-fangled method for retrieving repetitive and non-repetitive formulae come out in the given text.

The later part of this paper is planned as follows. Section 2 appraises the associated work on mathematical retrieval systems. Section 3 pacts with the probable approach. Section 4 includes the experimental results. As a final point, the conclusion is obtainable in section 5.

1.1 EXTRACTING MATH FORMULAS

Commonly, MathML and Latex are used for writing Math formulae. The text file with Math Keywords is encumbered in our program in add on mode. Then the math keywords from laden text Document acknowledged in Reading Mode. Pre-processing consent to read the file and hoard every string in the list after removing the stop words from the file. The subsequent step is to load the math keyword document into the program and build a list which encloses all the keywords related to math. Evaluate the math keyword list with the keywords in the text document to excavate math keywords [1]-[6].

2. MOTIVATION AND CONTRIBUTION

Mathematical formula retrieval for problem solving by Samarasinghe and Hui [2] proposed document retrieval to help solve mathematical problems. In this approach they have used Kohonen's Self Organizing Maps for data clustering. They have presented the proficiency of the planned approach with other clustering techniques [1]-[4].

Feature Extraction and Clustering-based Retrieval for mathematical formulae [1] planned that in order to present the scientific knowledge mathematical formulae or expressions are essential. As mathematical formulae contain both semantic and structural information it is very tricky to retrieve related mathematical formulae. They proposed an efficient approach with help of approach for the retrieval of mathematical formulae. They proposed a new approach with three popular clustering algorithms k-Means, Self-Organizing Map (SOM), and Agglomerative Hierarchical Clustering (AHC) for formula retrieval.

Rao et al. [7] implemented Leveshtein distance and Sequence matcher to check string similarity on Mathematical texts and keywords. An evaluation of their performance was carried out based on the time taken for the retrieval of keywords from Mathematical texts. Sequence matcher performed with lesser false negatives when compared with Levenshtein distance.

3. SEQUENCE MATCHER AND LEVENSHTEIN DISTANCE

Sequence Matcher is an agile class for match up pairs of sequences of any type, so long as the sequence components are analogous. The essential thought is to find the best ever neighboring matching sub string that contains useful elements. The similar idea is then continually applied to the segments of the successions towards right and towards the left of the similar

subsequence. This does not give up irrelevant edit sequences but does be inclined to capitulate matches that "look right" to people. Sequence Matcher gives the ratio in between 0 to 1 after evaluating two strings. When the assessment ratio after comparison between two strings is greater than 0.7 then it will be considered as keyword and keyword will be stored in the data set [8].

The Levenshtein Distance is used to assess the likeness between source string and objective string. The essential ideas of Levenshtein Distance are broadly used in areas like computer science, computational linguistics, bioinformatics, molecular biology, DNA analysis. It can also be used in evaluating the similarity of melodies or rhythms in music. The Levenshtein distance has extensively permeated in our day to day life. Levenshtein distance or edit distance is used for of spell checking and error correction in a program or in an application. The amalgamation of Levenshtein distance or edit distance with Trie index locates linked words faster. In order to transform one word into another word Levenshtein distance refers to the number of single character operations such as insertion, replacement or deletion. The edit distance between "cat" and "rat" is one, since substituting the character 'c' by 'r' the word "cat" can be converted to "rat".

4. FUZZY STRING MATCHING

Fuzzy string matching also describes fairly precise String Matching, which is a process of finding strings that approximately match a given pattern. Edit distance is used to calculate the proximity of match in terms of edit distance, which is the number of primordial operations necessary to translate the string into an accurate match. Primeval operations are usually: insertion, deletion and substitution. Fuzzy String Matching can have diverse realistic applications. Archetypal examples are spell-checking, text re-use detection, spam filtering, as well as quite a few applications in the bioinformatics domain like matching DNA sequences. Fuzzy-Wuzzy library used for testing string resemblance between two words or sentences and gives the ratio between 0 and 1. If the ratio is more nearer to 1 then we can say that particular words are well matched. If it is nearer to 0 then we can say both are irrelevant to each other.

There are four popular types of fuzzy matching logic supported by Fuzzy-Wuzzy package:

- Ratio: uses pure Levenshtein Distance based matching
- Partial Ratio: matches based on best substrings
- Token Sort Ratio: tokenizes the strings and sorts them alphabetically before matching
- Token Set Ratio: tokenizes the strings and compare the intersection and remainder

4.1 FLOWCHART OF ACTUAL PROCESS

The Fig.1 shows the actual procedure of retrieving Math formulae with proposed Fuzzy-Wuzzy (Partial Ratio) and Fuzzy-Wuzzy (Ratio).

4.1.1 Procedure of Fuzzy-Wuzzy based Retrieval of Mathematical Formulae:

Step 1: Open a file with mathematical text in append mode.

Step 2: Perform a read operation on the file and preprocess the data by splitting the lines of text into Mathematical Functions and removing the stop words.

Step 3: Create an Apriori Text File with predefined math functions and open it in read mode.

Step 4: Using fuzzy-wuzzy partial ratio or ratio function, check the matching of mathematical functions obtained from Step 2 with the predefined math functions in Apriori file.

a. If an exact match of a particular keyword is found then it will be printed as math keyword.

b. If not a keyword in text file is checked for its semantic string similarity with keywords in the Apriori Math file.

i. If the similarity ratio is greater than 0.75 on testing, then the particular keyword will be recognized as math keyword and appended into Apriori file.

ii. The process continues till the end of file

Step 5: This procedure is repeated again with New Text files containing mathematical functions.

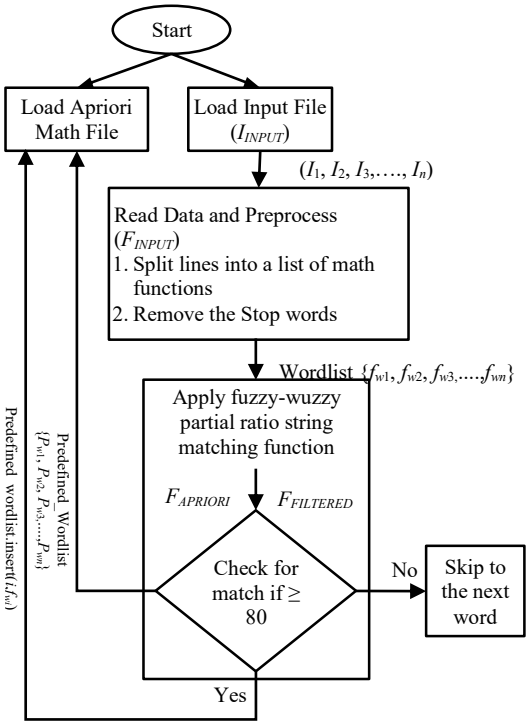

Fig.1. Procedure of Fuzzy-Wuzzy based retrieval of mathematical formulae

Algorithm Fuzzy-Wuzzy

Begin:

Step 1: Consider a Input File (F_{INPUT}) with mathematical text lines ($I_1, I_2, I_3, \ldots, I_n$)

Step 2: Consider a Apriori Math File ($F_{APRIORI}$) with *Predefined_Wordlist* {$P_{w1}, P_{w2}, P_{w3}, \ldots, P_{wn}$}

Step 3: Load Apriori Math File ($F_{APRIORI}$) in Append Mode ($a+$)

Step 4: Load Input File (F_{INPUT}) in Read Mode (r)

Step 5: Read data and Preprocess (F_{INPUT})

Split ($I_1,I_2,I_3,\ldots,I_n$) into a list of math functions {f_1, f_2, $f_3,\ldots,f_n$}
listfrom *File* := (F_{INPUT}).split(" ")
Remove the Stop words from {$f_1, f_2, f_3,...,f_n$}
stop_words :=set(*stopwords*.*words*('english'))
stop_words :=[*x*.*lower*() for *x* in *stop_words*]
return ($F_{FILTERED}$) with wordlist {$f_{w1}, f_{w2}, f_{w3},..., f_{wn}$})

Step 6: Apply function *fuzz_partial_ratio* ($F_{APRIORI}$, $F_{FILTERED}$)
Match {P_{w1}, P_{w2}, P_{w3},..., P_{wn}} with {$f_{w1}, f_{w2}, f_{w3},...,f_{wn}$}
For f_{wi} in $F_{FILTERED}$:
For P_{wi} in $F_{APRIORI}$:
if (*fuzz_partial_ratio*(f_{wi},P_{wi}) > 75 && f_{wi}.*isalpha*() && *len*(f_{wi}) ≥ 3) :
if(f_{wi} not in $F_{APRIORI}$) :
Append to $F_{APRIORI}$:=*Predefined_wordlist.insert*(*i*, f_{wi})
i := *i*++
end if
else:
Skip $f_{wi} := f_{wi+1}$
end if
end loop
end loop

Step 7: Repeat Step 6 till satisfactory

End

4.2 TIME ANALYSIS OF TWO DIFFERENT APPROACHES

Time analysis is the time taken to retrieve the matched formula. The RAM and processor speed plays a major role in the retrieval time. In this paper the experimentation for calculating the retrieval time is performed with 4 GB RAM and I3 Processor system.

Table.1. Time for retrieving matched formulae from the document with Sequence Matcher, Fuzzy-Wuzzy (Partial ratio), Fuzzy-Wuzzy (Ratio)

Number of formula	Sequence Matcher	Levenshtein Distance	Fuzzy-Wuzzy (Partial ratio)	Fuzzy-Wuzzy (Ratio)
40	3.042	3.28	2.8	3.89
30	3.046	2.91	3.402	3.5028
30	3.23	3.34	3.15	3.474
30	7.23	3.35	1.632	2.774
20	2.7	6.76	2.731	3.822
20	2.61	2.84	3.15	3.4971
20	2.74	17.32	2.872	4.246
20	2.75	15.42	2.75	3.802
40	5.59	2.62	2.925	3.38
30	3.09	3.22	2.951	3.641
30	3.03	2.97	4.078	3.459
20	2.74	3.84	3.198	3.5
30	15.2	2.8	3.026	3.905
20	2.82	2.93	18.512	4
20	2.8	17.3	2.929	3.637
30	3.21	3.24	2.991	3.71
30	4.23	3.39	3.503	3.081
30	4.62	2.95	3.378	3.632

4.3 SENSITIVITY MEASURE

Sensitivity is used to measure the ratio of actual math keywords that are exactly identified from the text file, supplied as an input. This can be expressed as:

$$\text{Sensitivity } (S) = \frac{n(Tp)}{n(Tp)+n(Fn)} \quad (1)$$

where, $n(Tp)$ is the number of True Positives and $n(Fn)$ is the number of False Negatives, where, $n(r)$ is the number of formulae retrieved, $n(f)$ is the total number of formulae and $n(U_r)$ is the Number of unwanted formulae retrieved

Table.2. Over all Sensitivity Measure with Sequence Matcher, Levenshtein Distance and Fuzzy-Wuzzy

	Sequence Matcher			Levenshtein Distance			Fuzzy-Wuzzy		
Samples with 20 formulae	***n(Tp)***	***n(Fn)***	***S***	***n(Tp)***	***n(Fn)***	***S***	***n(Tp)***	***n(Fn)***	***S***
Permutations	19	1	95%	17	3	85%	20	0	100%
Limits	20	0	100%	17	3	85%	20	0	100%
Sigma	20	0	100%	16	4	80%	20	0	100%
Square	19	1	95%	16	4	80%	20	0	95%
Trigonometric	19	1	95%	17	3	85%	19	1	95%
Factorial	18	2	90%	17	3	85%	19	1	95%
Differentiation	18	2	90%	17	3	85%	19	1	95%
Exponential	14	6	70%	17	3	85%	19	1	95%
Integral	20	0	100%	18	2	90%	20	0	100%
Logarithmic	20	0	100%	20	0	100%	20	0	100%
Overall	187	13	93.5%	172	28	86%	196	4	98%

4.4 EFFICIENCY

Efficiency is measured as the number of formulae retrieved from the number of the number of formulae in the training document. The sequence matcher retrieves maximum number of formulae in lesser time compared to Levenshtein Distance as it retrieves more unwanted formulae. The Proposed Fuzzy-Wuzzy method retrieves more number of related formulae from the test document that are matched with training document. The efficiency of proposed method Fuzzy-Wuzzy is more compared to Sequence matcher and Levenshtein Distance. The efficiency range with Fuzzy-Wuzzy method is almost from 95-100% as shown in Table.3.

Efficiency Measure (E_M) is calculated using the below equation:

$$E_M = \frac{n(r)}{n(f)} \times 100 \quad (2)$$

Table.3. Overall Efficiency Measure with Sequence Matcher, Levenshtein Distance and Fuzzy-Wuzzy

	Sequence Matcher			**Levenshtein Distance**			**Fuzzy-Wuzzy**		
Samples with 20 formulae	***n*(*r*)**	***n*(U_r)**	***E*M**	***n*(*r*)**	***n*(U_r)**	***E*M**	***n*(*r*)**	***n*(U_r)**	***E*M**
Permutations	19	-	95%	17	3	85%	20	-	100%
Limits	20	-	100%	17	4	85%	20	-	100%
Sigma	20	-	100%	16	3	80%	20	-	100%
Square	19	-	95%	16	4	80%	20	-	100%
Trigonometric	19	-	95%	17	4	85%	19	-	95%
Factorial	18	2	90%	17	3	85%	19	1	95%
Differentiation	18	3	90%	17	3	85%	19	1	95%
Exponential	14	-	70%	17	4	85%	19	1	95%
Integral	20	-	100%	18	3	90%	20	-	100%
Logarithmic	20	-	100%	20	7	100 %	20	1	100%

5. COMPARATIVE STUDY

The recital of fuzzy-wuzzy is calculated in this section. Primarily the string comparison is performed with only one dataset file, after preprocessing and executing with fuzzy-wuzzy a new dataset will be attained. The amount of the data set will amplify after consecutive string comparisons. The planned approach identifies math related keywords along with some unnecessary words. If any math associated keyword is present in the tested document it is dynamically inserted into our math dataset if it does not exist before, next time while loading file the recently updated dataset is used for string matching. The efficiency of proposed approach is measured in terms of time analysis. String comparison with fuzzy-wuzzy is accomplished with exact matching and completed with less time than sequence matcher and Levenshtein distance in terms of Time Analysis is shown in Fig.2 and Fig.3. The Comparative study in terms of Efficiency and Sensitivity is presented in Fig.4, and Fig.5.

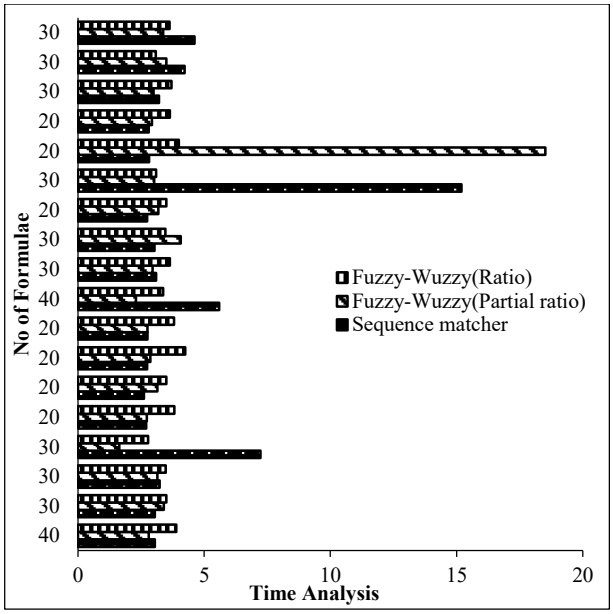

Fig.2. Comparison between Sequence Matcher, Fuzyy-Wuzzy (Partial Ratio) and Fuzzy-Wuzzy (Ratio) in terms of Time analysis

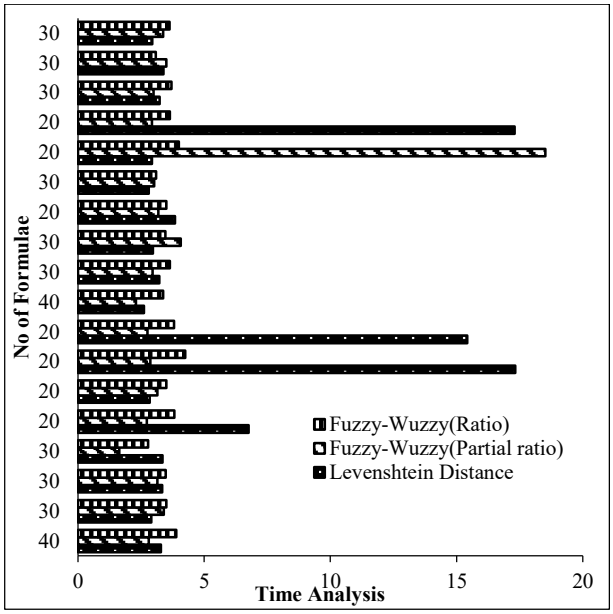

Fig.3. Comparison between Levenshtein Distance, Fuzyy-Wuzzy (Partial Ratio) and Fuzzy-Wuzzy (Ratio) in terms of Time analysis

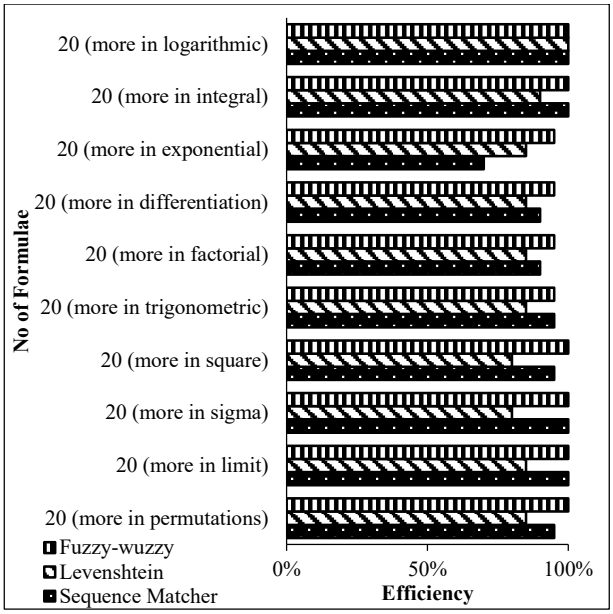

Fig.4. Comparison between Levenshtein Distance, Fuzyy-Wuzzy and Sequence Matcher in terms of Efficiency

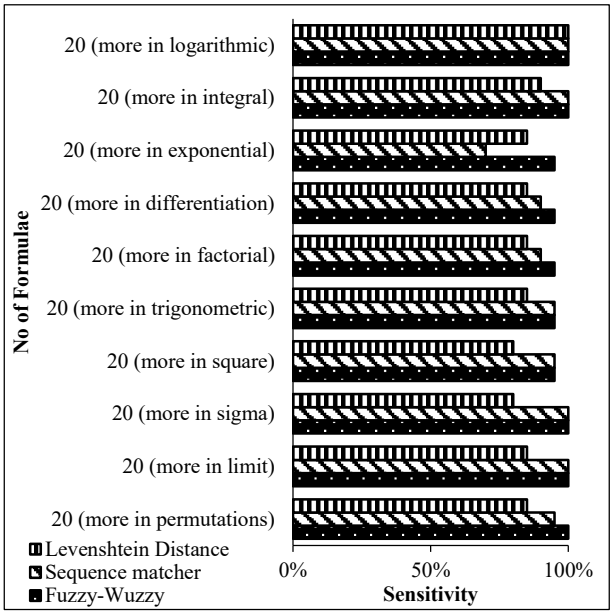

Fig.5. Comparison between Levenshtein Distance, Fuzyy-Wuzzy and Sequence Matcher in terms of Sensitivity

6. CONCLUSION

In this article we have made an evaluation of three different string matching algorithms namely, Levenshtein, Sequence Matcher and Fuzzy-Wuzzy to retrieve approximate matches of Mathematical formulae from texts. There are four variants of Fuzzy-Wuzzy of which two variants are found applicable to this study. The partial ratio based fuzzy-wuzzy retrieves the string exactly and in lesser time than ratio based Fuzzy-Wuzzy technique. Through our observations, we found that sequence matcher does not retrieve all strings, Levenshtein's distance retrieves false negatives whereas fuzzy-wuzzy partial ratio retrieves most of the mathematical formulae from the text. To assert this, we have made a comparison taking into regard, efficiency measure, sensitivity measure and Time series exploration. Fuzzy-Wuzzy performed better on efficiency measure and sensitivity measure and Sequence matcher scored high on time series exploration.

REFERENCES

[1] Kai Ma, Siu Cheung Hui and Kuiyu Chang, "Feature Extraction and Clustering-based Retrieval for Mathematical Formulas, *Proceedings of 2nd International Conference on Software Engineering and Data Mining*, pp. 372-377, 2010.

[2] Sidath Harshanath Samarasinghe and Siu Cheung Hui, "Mathematical Document Retrieval for Problem Solving", *Proceedings of International Conference on Computer Engineering and Technology*, pp. 583-587, 2009.

[3] J. Misutka and L. Galambos, "Mathematical Extension of Full Text Search Engine Indexer", *Proceedings of 3rd International Conference on Information and Communication Technologies: From Theory to Applications*, pp. 1-6, 2008.

[4] B.R. Miller and A. Youssef, "Technical Aspects of the Digital Library of Mathematical Functions", *Annals of Mathematics and Artificial Intelligence*, pp. 121-136, 2003.

[5] H. Zhang and M.S. Lin, "An Evolutionary K-means Algorithm for Clustering Time Series Data", *Proceedings of International Conference on Machine Learning and Cybernetics*, pp. 1282-1287, 2004.

[6] M. Kohlhase. "Markup for Mathematical Knowledge", *Proceedings of an Open Markup format for Mathematical Documents*, pp. 13-23, 2006.

[7] G. Appa Rao, K. Venkata Rao, P.V.G.D. Prasad Reddy and T. Lava Kumar, "An Efficient Procedure for Characteristic Mining of Mathematical Formulas from Document", *International Journal of Engineering Science and Technology*, Vol. 10, No. 3, pp. 152-157, 2018.

[8] G. Appa Rao, G. Srinivas, K. Venkata Rao and P.V.G.D. Prasad Reddy, "Characteristic Mining of Mathematical Formulas from Document-A Comparative Study on Sequence Matcher and Levenshtein Distance Procedure", *International Journal of Computer Sciences and Engineering*, Vol. 6, No. 4, pp. 400-403, 2018.