Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 17 (2013) 121 – 128

Procedia Computer Science

Information Technology and Quantitative Management (ITQM2013)

Document Clustering Using Hybrid XOR Similarity Function for Efficient Software Component Reuse

Vangipuram Radhakrishna[a]*,C.Srinivas[b],Dr.C.V.Guru Rao[c]

[a]Department of IT, VNR Vignana Jyothi Institute of Engineering and Technology,Bachupally,Hyderabad,INDIA
[b]*Associate Professor of CSE, Kakatiya Institute of Technology,Warangal,INDIA*
[c]*Professor of CSE,S.R Engineering College,Warangal,INDIA*

Abstract

In this paper a generalized approach is proposed for clustering a set of given documents or text files or software components for reuse based on the new similarity function called hybrid XOR function defined for the purpose of finding degree of similarity among two document sets or any two software components. We construct a matrix called similarity matrix of order n-1 by n for n document sets or software components by applying hybrid XOR function for each pair of document sets. We define and design the clustering algorithm which has its input as similarity matrix and output as a set of clusters formed dynamically as compared to other clustering algorithms that predefine the count of clusters and documents being fit to one of those clusters or classes finally. The approach carried out uses simple computations.

Selection and peer-review under responsibility of the organizers of the 2013 International Conference on Information Technology and Quantitative Management

Keywords : hybrid xor ;clustering; frequent itemsets; cluster

1. Introduction

Software reuse can be defined as the process of developing new software systems by making use of existing software components from other systems. Reuse eliminates the need for developing a software system from the scratch and reduces the cost of productivity. The component for reuse may be a requirement document or test document or modules to name a few. Software Component retrieval from the Software component repository has gained a significant importance from the researchers and also from industry perspective. The retrieval of any component from the repository requires a search algorithm that can retrieve the software component with the features specified by the user in the user query.

Clustering is a process which when applied to software components groups all the components with similar

* Corresponding author. Tel.: +919700684242; fax: +0-000-000-0000 .
E-mail address: radhakrishna_v@vnrvjiet.in.

1877-0509
Selection and peer-review under responsibility of the organizers of the 2013 International Conference on Information Technology and Quantitative Management
doi:10.1016/j.procs.2013.05.017

feature into one class and those with dissimilar features into another class. Clustering reduces the search time complexity as all the similar components are grouped into one class. Clustering is not any one specific algorithm that we can stick firm to, but it must be viewed as the general task to be solved.

Document clustering or text clustering is one of the main themes in text mining [3]. It refers to the process of grouping documents with similar contents or topics into clusters to improve both availability and reliability of text mining applications such as information retrieval, classifying text, summarizing document sets, etc.

2. Taxonomy

The problem of finding frequent itemsets is dealt widely in the literature [8]. Frequent item sets originate from association rule mining. Recently, it has being applied in the area of text mining for document categorization, document clustering. In [1] clustering a given set of text documents from neighbour set is proposed. In [2] the authors propose a methof for discovering maximum length frequent item sets.

In [6], the classification of text files or documents is done by considering Gaussian membership function and making use of it to obtain clusters by finding word patterns. Each cluster is identified by its word pattern calculated using Fuzzy based Gaussian membership function once clusters are formed.

A new method called Maximum Capturing is proposed for document clustering is proposed in [3]. Maximum Capturing includes two procedures as finding constructing document clusters and assigning cluster topics. In [10], algorithm to search for a pattern in a text is proposed which can be used to search for component of interest in the component repository.

3. Proposed Method

In this paper we address two issues

1. Clustering a given set of documents
2. Clustering Software components which may be modules or functions.

In the case of document clustering the idea is to first obtain frequent item sets for each document using any of the existing association rule mining algorithms. This involves elimination of stop words and stemming words from each document to reduce the dimensionality of a document as all the words of a document do not associate technical or semantic meaning and also it is not appropriate to consider whole word set as it increases the processing time of each document. The idea is to make use of only those word sets which can form candidate solutions in defining the clusters. This process is followed by finding the frequent item sets using any of the existing algorithms for association rule mining. We then form a boolean matrix with rows indicating documents and columns indicating unique frequent item sets from each document. This is further followed by the computation of a binary feature vector for each document pair, represented as a 2D array or 2D matrix by redefining the XOR function as hybrid XOR similarity function with slight modification in the function introducing high impedance variable as Z.

The algorithm for document clustering has its input as documents with frequent item sets and output as set of clusters formed dynamically. The approach followed is a tabular approach

3.1 Algorithm for Document Clustering

Document_Clustering (Document Set, Frequent item sets)
Begin of Algorithm
Step1:
For each document D do
Begin

Step1. Remove stop words and stemming words from each document.
Step2. Find unique words in each document and count of the same.
Step3. Find frequent itemsets of each document
End for

Step 2: Form a word set W consisting of each word in frequent item sets of each document.

Step 3: Form Dependency Boolean Matrix with each row and column corresponding to each document and each word respectively

For each document in document set do
Begin
For each word in word set to
Begin
If (word W_k in Word set W is in document D_i)
Begin
Set $D[D_i, W_k] = 1$
Else
Set $D[D_i, W_k] = 0$
End if
End for
End for

Step 4: Find the Feature vector similarity matrix by evaluating similarity value for each document pair applying Hybrid XOR Function defined in table 1 to obtain the matrix with feature vectors for each document pair.

Step 5: Replace the corresponding cells of matrix by count of number of zeroes in tri state feature vector.

Step 6: At each step, find the cell with maximum value and the document pairs containing this value in the Similarity matrix. Group such document pairs to form the clusters. Also if document pair (I, J) is in one cluster and document pair (J, K) is in another cluster, form a new cluster containing (I, J, K) as its elements.

Step 7: Repeat Step6 until no documents exist or we reach the stage of first minimum value leaving zero entry.

Step 8: Output the set of clusters obtained.

Step 9: label the clusters by considering candidate entries.

End of algorithm

We define the Similarity function **S** as a function of documents A and B which is a tri state function as shown below in the truth table of Table 1.

Table 1. Truth Table of hybrid XOR Similarity Function

A	B	S(A.B)
0	0	Z
0	1	1
1	0	1
1	1	0

Hypothesis-1

If a frequent item set exists in the document, then the cell value of the matrix corresponding to D $[d_i,w_k]$ is made 1 else the cell value is made as zero where i varies from 1 to n is index of each document and k is index of each frequent item.

4. Case Study

4.1 Case study of Document Clustering

For sake of simplicity we consider document sets each treated as a transaction of frequent itemsets. We can also consider software components with high features for clustering so that they can be reused efficiently. Consider the document sets with the frequent item sets obtained after mining using any one of the association rule mining algorithms as shown below.

Table 2. Documents and Corresponding Frequent item sets

DOCUMENTS	FREQUENT ITEMSETS
DOCUMENT 1	{ ENCRYPT, NEURAL NETWORKS, CLUSTER}
DOCUMENT 2	{SVM , MINING, CLUSTER}
DOCUMENT 3	{ SVM, NEURAL NETWORKS, MINING, CLUSTER}
DOCUMENT 4	{ENCRYPT, NEURAL NETWORKS, MINING, CLUSTER}
DOCUMENT 5	{SVM , CLUSTER}
DOCUMENT 6	{ENCRYPT, NEURAL NETWORKS, MINING}
DOCUMENT 7	{ENCRYPT, SVM, NEURAL NETWORKS}
DOCUMENT 8	{SVM, NEURAL NETWORKS}
DOCUMENT 9	{NEURAL NETWORKS, MINING, CLUSTER}

We now construct a Boolean matrix with rows indicating each document and column corresponding to each unique frequent item from set of frequent item sets of all documents sets respectively.

Table 3. Boolean matrix Representation of Table.2

	ENCRYPT	NEURAL NETWORKS	CLUSTER	SVM	MINING
D1	1	1	1	0	0
D2	0	0	1	1	1
D3	0	1	1	1	1
D4	1	1	1	0	1

D5	0	0	1	1	0
D6	1	1	0	0	1
D7	1	1	0	1	0
D8	0	1	0	1	0
D9	0	1	1	0	1

We form a matrix D [n-1, n] for n documents and consider only the upper triangular region. The cells of the matrix are filled by applying the similarity function S for which each document pair forms the input as shown below in table 4.

Table 4. Feature Vector Representation of document set

	D1	D2	D3	D4	D5	D6	D7	D8	D9
D1	x	{1,1,0,1,1} = 1	{1,0,0,1,1}=2	{0,0,0,Z,1}=3	{1,1,0,1,Z=1	{0,0,1,Z,1}=2	{0,0,1,1,Z}=2	{1,0,1,1,Z}=1	{1,0,0,Z,1=2
D2	x	x	{Z,1,0,0,0}=3	{0,0,0,0,1}=4	{Z,Z,0,0,=2	{1,1,1,1,0}=1	{1,1,1,0,1}=1	{Z,1,1,0,1}=1	{Z,1,0,1,0}=2
D3	x	x	x	{1,0,0,1,0}=3	{Z,1,0,0,1}=2	{1,0,1,1,0}=3	{1,0,1,0,1}=2	{Z,0,1,0,1}=2	{Z,0,0,1,0}=3
D4	x	x	x	x	{1,1,0,1,1}=1	{0,0,1,0,0}=4	{0,0,1,1,1 }=2	{ 1,0,1,1,1}=1	{ 1,0,0,Z,0}=3
D5	x	x	x	x	x	{1,1,1,1,1}=0	{1,1,1,0,Z}=1	{Z,1,1,0,Z}=1	{Z,1,0,1,1}=1
D6	x	x	x	x	x	x	{0,0,Z,1,1}=2	{1,0,Z,1,1}=1	{1,0,1,Z,0}=2
D7	x	x	x	x	x	x	x	{1,0,Z,0,Z}=2	{1,0,1,1,1}=1
D8	x	x	x	x	x	x	x	x	{Z,0,1,1,1}=1

Once we obtain the above table with feature vectors for each document pair then we replace the corresponding cells of matrix by count of number of zeroes in tri state feature vector as given in table 5.

Table 5. Similarity Matrix with Feature Vector Replaced by Count of 0s.

	D1	D2	D3	D4	D5	D6	D7	D8	D9
D1	x	1	2	3	1	2	2	1	2
D2	x	x	3	4	2	1	1	1	2
D3	x	x	x	3	2	3	2	2	3
D4	x	x	x	x	1	4	2	1	3
D5	x	x	x	x	x	0	1	1	1
D6	x	x	x	x	x	x	2	1	2
D7	x	x	x	x	x	x	x	2	1
D8	x	x	x	x	x	x	x	x	1

Step1: find the first maximum value from the matrix and target only those cells having this value to form initial cluster.

Table 6. Content of Similarity Matrix showing step1

	D1	D2	D3	D4	D5	D6	D7	D8	D9
D1	x	1	2	3	1	2	2	1	2
D2	x	x	3	0	2	1	1	1	2
D3	x	x	x	3	2	3	2	2	3
D4	x	x	x	x	1	0	2	1	3
D5	x	x	x	x	x	0	1	1	1
D6	x	x	x	x	x	x	2	1	2
D7	x	x	x	x	x	x	x	2	1
D8	x	x	x	x	x	x	x	x	1

Find max value from the above table which is 4 here and target those cells as they form the best candidate solutions.

Stage1: (2, 4) and (4, 6) have val as 4. So form cluster as (2, 4, 6).

Step 2: Find the next max value from the above table which is 3 here and target those cells as they form the best candidate solutions. Now cluster {2, 4, 6} is dynamically changed to {1, 2, 3, 4, 6, 9} and is no more a separate cluster.

Table 7. Content of Similarity Matrix showing step2

	D1	D2	D3	D4	D5	D6	D7	D8	D9
D1	x	1	2	3	1	2	2	1	2
D2	x	x	z	0	2	1	1	1	2
D3	x	x	x	3	2	3	2	2	3
D4	x	x	x	x	1	0	2	1	z
D5	x	x	x	x	x	0	1	1	1
D6	x	x	x	x	x	x	2	1	2
D7	x	x	x	x	x	x	x	2	1
D8	x	x	x	x	x	x	x	x	1

Find the next max value from the above table which is 3 here and target those cells as they form the best candidate solutions.

Stage2: consider only un-clustered document set {1, 3, 5, 7, 8, 9} and search for value 3 in corresponding columns. (1,4)-(3,4)-(3,6)-(3,9) : So form cluster {2,4,6,1,3,9} as new Cluster. Set the values as zero.
Cluster 1: {1, 2, 3, 4, 6, 9}.

Step 3: Find the next max value from the above table which is 2 here and target those cells as they form the best candidate solutions.

Table 8. Content of Similarity Matrix showing step3

	D1	D2	D3	D4	D5	D6	D7	D8	D9
D1	x	1	2	x	1	2	2	1	2
D2	x	x	x	x	2	1	1	1	2
D3	x	x	x	x	2	x	2	2	x
D4	x	x	x	x	1	x	2	1	x
D5	x	x	x	x	x	0	1	1	1
D6	x	x	x	x	x	x	2	1	2
D7	x	x	x	x	x	x	x	2	1
D8	x	x	x	x	x	x	x	x	1

Find the next max value from the above table which is 2 here and target those cells as they form the best candidate solutions.

Stage3: consider only un-clustered document set {5, 7, 8} ad search for value 2 in corresponding columns. Here (7, 8) has 2. So form cluster {7, 8} as new Cluster. Set the values as zero or x.
Cluster 2: {7, 8}

Step 4: Find the next max value from the above table which is 1 here and target those cells as they form the best candidate solutions.

Table 9. Content of Similarity Matrix showing step4

	d1	d2	d3	d4	d5	d6	d7	d8	d9
D1	x	1	2	x	1	2	2	1	2
D2	x	x	x	x	2	1	1	1	2
D3	x	x	x	x	2	x	2	2	x
D4	x	x	x	x	1	x	2	1	x
D5	x	x	x	x	x	0	1	1	1
D6	x	x	x	x	x	x	2	1	2
D7	x	x	x	x	x	x	x	x	1
D8	x	x	x	x	x	x	x	x	1

Stage4: consider only un-clustered document set {5} and search for value 1 in corresponding columns. Here (5, 7), (5, 8), (5, 9) are all 1s. But this is next minimum value after zero if we consider initial table values before clustering. Hence 5 can't be similar to any of those documents and we must place it as a separate cluster {5}.

The clusters finally formed are as shown below in the following figure.

O/P: Set of clusters

Cluster-1: {1, 2, 3, 4, 6, 9}
Cluster-2 : { 7, 8}
Cluster-3 : { 5}

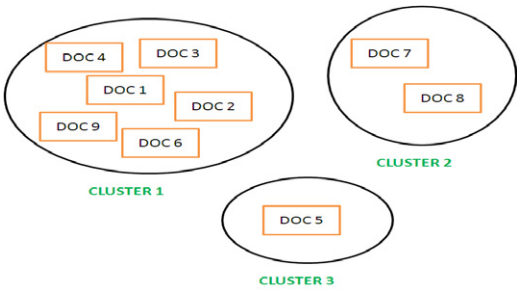

Fig. 1. Set of Clusters formed after applying the algorithm

Once the clusters are formed we can label the clusters for the purpose of identification by using candidate item sets approach followed in [3] or pass it to SVM to classify. As search operation is the bottleneck here we use the pattern search algorithm [10] to find if a given word is in a set of documents.

4.2 Component clustering Using Hybrid XOR Similarity function

Consider the five components C_1, C_2, C_3, C_4, C_5 with functional descriptions and weights assigned as VH, H, M, VL, L and need to be clustered to place all similar components in the repository. The table below are self explanatory and we can see (C_1, C_4) form one cluster. It can also be verified that these are similar to reference component R_1 and may be replaced by the stored reference component if required or stored in to the repository.

Table 10. Sample Source Components with functional properties and Stored Reference Components in repository

Component	Features	
C1	Push Element	H
	Pop Element	H
C2	Array Size	VL
	Queue Size	L
C3	Top Stack	M
	POP Stack	H
C4	Push Element	H
C5	Array Size	L
	Queue Size	M

Reference elements			
R1		R2	
Push Element	M	Array Size	M
Pop Element	H	Queue Size	L
		Top Stack	VL

Table 11. Source Components with functional properties in matrix form where U- indicates undefined.

	Push Element	Pop Element	Array Size	Queue Size	Top Stack	Pop Stack
C1	H	H	U	U	U	U
C2	U	U	VL	L	U	U
C3	U	U	U	U	M	H
C4	H	U	U	U	U	U
C5	U	U	L	M	U	U

Table 12. Source Components with functional properties in matrix form after applying similarity function

	C1	C2	C3	C4	C5
C1	X	6Z	6Z	1	6Z
C2	X	X	6Z	6Z	6Z

C3	X	X	X	6Z	6Z
C4	X	X	X	X	6Z
C5	X	X	X	X	X

(C_1, C_4) forms one cluster.
C_2, C_3, C_5 are each in different cluster.

5. Conclusion

In this paper an attempt is made to study the problem of clustering and finally we come up with a unique approach carried out to cluster a set of given documents or text files or software components based on the new similarity function called hybrid XOR function defined for the purpose of finding degree of similarity among two document sets. We show the construction of similarity matrix of the order n-1 by n for n document sets generated by applying the hybrid XOR function for each pair of document or component sets. The Proposed algorithm has the input as similarity matrix and output being set of clusters formed dynamically as compared to other clustering algorithms that predefine the count of clusters and documents being fit to one of those clusters or classes finally. The approach can be justified as it carries out very simple computational logic and efficient in terms of processing. The approach can be extended to classify using classifiers and applying fuzzy logic in future.

References

[1] Congnan Luo, , Yanjun Li, Soon M. Chung. Text document clustering based on neighbors , Data & Knowledge Engineering (68) ,2009,1271–1288
[2] Tianming Hu,Sam Yuan Sung, Hui Xiong, Qian Fu. Discovery of maximum length frequent itemsets, Information Sciences (178), 2008,69–87
[3] Wen Zhanga,, Taketoshi Yoshida, Xijin Tang, Qing Wang. Text clustering using frequent itemsets, Knowledge-Based Systems 23 (2010) 379–388
[4] Wen Zhanga,Taketoshi Yoshida, Xijin Tang. A comparative study of TF*IDF, LSI and multi-words for text classification. Expert Systems with Applications 38 (2011) 2758–2765
[5] Vincent Labatut and Hocine Cherifi. Accuracy Measures for the Comparison of Classifiers, ICIT 2011 The 5th International Conference on Information Technology
[6] Jung-Yi Jiang et.al A Fuzzy Self-Constructing Feature Clustering Algorithm for Text Classification, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 23, NO. 3, MARCH 2011
[7] Melita Hajdinjak, Andrej Bauer. Similarity Measures for Relational Databases, Informatica 33 (2009) 143–149
[8] R. Agrawal, T. Imielinski, A. Swami. Mining association rules between sets of items in very large databases, Proceedings of the ACM SIGMOD Conference on Management of data, 1993, pp. 207–216
[9] F. Beil, M. Ester, X.W. Xu, Frequent term-based text clustering, in: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002, pp. 436–442
[10] Radhakrishna.V,C.Srinivas, C.V.Guru rao. High Performance Pattern Search algorithm using three sliding windows , International Journal of Computer Engineering and Technology , Volume 3,issue 2, 2012 , pages 543-552. Impact factor 3.85
[11] Yi Peng, Gang Kou. A Descriptive frame work for the field of data mining and knowledge discovery , International Journal of Information Technology and Decision making, Volume 7, No.4, 2008,Pages 639-682,Impact Factor 3.139
[12] Kou,G and Lou,C. Multiple Factor Hierarchical Clustering Algorithm for Large Scale Web Page and Search Engine Clickstream Data, Vol 197,Issue 1,Page 123-134, Annals of Operation Search,2012.
[13] Khuzaima.S.Daudjee . Organizing Reusable Software Repositories through Hierarchical Clustering .1994
[14] S. Tangsripairoj and M. H. Samadzadeh. Organizing and visualizing software repositories using the growing hierarchical self Organizing map. Journal of Information Science and Engineering, Vol 22, 283-295,2006.
[15] Brain S. Mitchell , Spiros Mancoridis. Comparing the Decompositions Produced by Software Clustering Algorithms by Similarity Clustering,Proceedings of the IEEE International Conference on Software maintenance,744-753 , 2001.
[16] M.Turan and Zehra.C. Clustering and Dimensionality Reduction to Determine Important Software Quality Metrics.
[17] Sunil Kumar and Rajesh Bhatia. Object UseCases Through MDL. Proceedings of the IEEE International symposium on Computer and Information sciences, 2007.