Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 31 (2014) 1044 – 1050

Procedia Computer Science

2nd International Conference on Information Technology and Quantitative Management, ITQM 2014

Clustering and Classification of Software Component for Efficient Component Retrieval and Building Component Reuse Libraries

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

[a]*Kakatiya Institute of Technology and Science, Warangal,INDIA*
[b]*VNR Vignana Jyothi Institute of Engineering and Technology,Hyderabad,INDIA*
[c] *Professor of CSE & Principal, S.R.Engineering College ,Warangal,INDIA*

Abstract

A Software Repository is a collection of library files and function codes. Programmers and Engineers design develop and build software libraries in a continuous process. Selecting suitable function code from one among many in the repository is quite challenging and cumbersome as we need to analyze semantic issues in function codes or components. Clustering and Mining Software Components for efficient reuse is the current topic of interest among researchers in Software Reuse Engineering and Information Retrieval. A relatively less research work is contributed in this field and has a good scope in the future. In this paper, the main idea is to cluster the software components and form a subset of libraries from the available repository. These clusters thus help in choosing the required component with high cohesion and low coupling quickly and efficiently. We define a similarity function and use the same for the process of clustering the software components and for estimating the cost of new project. The approach carried out is a feature vector based approach.

Keywords : component; reuse; similarity; cluster ; mining ; repository

1. Introduction

The Universality and Ubiquitous nature of the internet made the possibility to access a huge amount of virtual unlimited information in the digital text format by the humans [17]. This gave a new definition for data mining creating a new set of possibilities for mining text information.

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

doi:10.1016/j.procs.2014.05.358

Consequently the text oriented derivation of data mining called as text mining has been gaining a lot of practical significance as the available data grows at higher rate than can be perceived or handled by the humans. In this context, the problem of clustering is one topic that gained more practical importance from the researchers and more specifically from the perspective of the software industry.

From software engineering perspective, the requirement for clustering comes from the need for software component classification, software component clustering, analysing project behavior for predicting the project cost behavior, performing software component search and for the component retrieval from software library or software repository. Clustering is also widely used in many practical domains such as text classification, bioinformatics, medicine, image processing to name a few.

The component clusters obtained from clustering process may be viewed as the highly cohesive groups with low coupling which is the desired feature. Software component clustering may be defined as the process of grouping similar set of software patterns together [1]. The input to the clustering algorithm can be a set of software entities or software patterns or software requirement documents or any set of software components. The result of the software component clustering process is a set of clusters having high cohesive nature w.r.t other software clusters. The descriptions or representations of the clusters formed may be used for the decision making for selecting a software component or software pattern of interest. The interesting feature or property of the clustering process is that all the software components within a single cluster share common properties in some sense and patterns in different clusters are dissimilar. From perspective of software engineering, all the components within same cluster have high cohesion and low coupling. 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.

Clustering algorithms may unsupervised or supervised [1]. In unsupervised clustering the partitions are viewed as the unlabelled patterns or components. Supervised clustering algorithms label the patterns which can be used to classify the components for decision making. Hence the partitions obtained by clustering process may be labeled or unlabeled.

A new method called Maximum Capturing is proposed for document clustering [4].Maximum Capturing includes two procedures: 1.Constructing document clusters and 2. Assigning cluster topics. The search complexity can be reduced by using the algorithm [11] where ever necessary as part of component retrieval.

2. Related Works

The problem of finding frequent itemsets is first initiated from [9] which use frequent items to find association rules in large transactional databases. In [2] clustering a given set of text documents from neighbor set is proposed. In [3] the authors propose a method for discovering maximum length frequent item sets. In [7], the classification of text files or documents is done by considering Gaussian membership function and d 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 functions once clusters are formed. In this paper the idea is to first obtain frequent item sets for each document using existing association rule mining algorithms either by horizontal or vertical approach. Once we find frequent itemsets in each document then we form a Boolean matrix with rows indicating documents and columns indicating unique frequent items from each document. This is followed by the computation of a ternary feature vector for each document pair, represented as a 2D array or 2D matrix by redefining the XNOR function as hybrid XNOR logic with slight modification in the function introducing high impedance variable as Z. The idea of maximum capturing is taken as the base framework for clustering [4]. The authors perform clustering using XOR similarity function [15].

3. Proposed Work

3.1 Similarity Measure

To perform clustering, we define a similarity measure which may be used to find the similarity between any pair of software components or software patterns or software documents as given in Table.A. The software components may be software requirement specification files or software product files or program codes. The proposed similarity measure Sim (E1, E2) is a function of any two attributes E1 and E2. We consider E1 and E2 as the attributes present in two different files F1 and F2 respectively.

Table A. Proposed Similarity Measure

E1	**E2**	**Sim(E1,E2)**
Absent	**Absent**	**Neglect (say Z)**
Absent	**Present**	**0**
Present	**Absent**	**0**
Present	**Present**	**1**

The input for component clustering algorithm is a set of software components with properties predefined and the output is a set of highly cohesive components with low coupling feature.

3.2 Algorithm for Clustering

The algorithm may be used to cluster software requirement documents or program codes and this work is an extension of [15, 17]. If the dimensionality is very high, then to reduce the dimension of the documents we may eliminate stop words and stemming words by forming a list of stop words separately and storing in a file. When clustering English documents we may use porter stemmer algorithm. In case, if the dimensionality of software documents is too large to handle, then to further reduce the dimension we may apply Singular value decomposition as given in step1 of the algorithm.

Algorithm. Algorithm for Component Clustering.

```
Input: Document set, frequent items.
Output: set of clusters.
Begin of Algorithm
Step1:
    For each document D do
    Begin
      Step1.1   Remove stop words and stemming words from each document.
      Step1.2   Find unique words in each document and count of the same.
      Step1.3   Find reduced dimension set by applying Singular Valued Decomposition to all the files.
    End for
Step 2:  Form a word set W consisting of each word in reduced item set of each document of step1.
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 do
          Begin
           If (word wk in Word set W is in document Di)
             Begin
             Set D[Di, wk] = 1
 Else
             Set D[Di, wk] = 0
  End if
End for
       End for
Step 4:  Find the Feature vector similarity matrix by evaluating similarity value for each component pair applying
         similarity measure defined in table 1 to obtain the matrix with feature vectors for each component or 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 document pairs containing this value in the matrix.
        Group such document pairs to form clusters. Also if document pair (X,Y) is in one cluster and document pair
        (Y, Z) is in another cluster, form a new cluster containing (X, Y, Z) as its elements.
Step 7:  Repeat Step6 until no components or 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
```

4. Case Study- Clustering Software Projects to Estimate Project Cost

Consider the sample parameters for the software project based on the programming language used, platform, software type, existing plan, and development model [16]. If we need to cluster or classify the set of projects based on these parameters then we can use the proposed algorithm to achieve the task.

Let the parameters contain the domain as

Programming Language = {C: 1, C++:2, JAVA: 3, COBOL: 4, C##:5}

Platform= {Personal Computer: 1, Mobile: 2, Mainframe: 3}

S/W TYPE = {Application: 1, Embed: 2}

PLAN = {Yes: 1, No: 0}

Development Model = {Waterfall: 1, Iterative: 2, Agile: 3, Spiral: 4, Prototype: 5}

We form the matrix with rows indicating the projects and columns indicating the factors considered for the project evaluation using direct approach.

Table 1. Matrix showing the combination of Project VS Parameters.

	PL	Platform	S/W type	Plan	Modern Practice	Dev. Model	Priority	Risks	Release type
Project 1	C	PC	application	no	yes	waterfall	low	less	1.0
Project 2	C++	PC	application	yes	yes	iterative	low	Less	1.0
Project 3	Java	Mobile	embedded	no	yes	agile	high	medium	1.0
Project 4	C	Mobile	embedded	no	yes	Spiral	low	less	1.0
Project 5	C	Mobile	embedded	no	yes	Prototype	low	less	1.0
Project 6	Cobol	mainframe	application	yes	no	Iterative	medium	medium	2.0
Project 7	Java	Mobile	embedded	no	yes	Spiral	low	less	2.0
Project 8	C#	Mobile	embedded	no	yes	agile	low	less	2.0

Table 2. Matrix obtained after index generation.

Project Vs Attribute	**PL**	**Platform**	**S/W type**	**Plan**	**Modern Practice**	**Dev.Mode**	**Priority**	**Risks**	**Release type**
Project 1	1	1	1	2	1	1	1	1	1.0
Project 2	2	1	1	1	1	2	1	1	1.0
Project 3	3	2	2	2	1	3	2	2	1.0
Project 4	1	2	2	2	1	4	1	1	1.0
Project 5	1	2	2	2	1	5	1	1	1.0
Project 6	4	3	1	1	2	2	3	2	2.0
Project 7	3	2	2	2	1	4	1	1	2.0
Project 8	5	2	2	2	1	3	1	1	2.0
Project 9	3	2	2	2	1	4	1	1	1.0

Table 3. Similarity Matrix obtained from Table.2.

	P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**
P1	X	6	2	6	6	1	4	4	5
P2	X	X	2	4	4	4	3	3	4
P3	X	X	X	5	5	1	5	3	4
P4	X	X	X	X	8	0	7	6	8
P5	X	X	X	X	X	0	6	6	7
P6	X	X	X	X	X	X	1	1	1
P7	X	X	X	X	X	X	X	7	8
P8	X	X	X	X	X	X	X	X	6
P9	X	X	X	X	X	X	X	X	X

Stage 1: Choose the cell with maximum value from Table.3. Here (P_4,P_5), (P_4, P_9), (P_7, P_9) have value as 8.

Group these cells to form an initial cluster containing projects (P_4, P_5, P_7, P_9).

We thus have CLUSTER-1: {P_4, P_5, P_7, P_9}.

Stage 2: Choose the cell with next maximum value from Table.4. Here (P_1, P_2) are having value 6. Group these cells to form a cluster containing projects (P_1, P_2).

Table 4. Reduced Similarity Matrix obtained from Stage 1.

	P1	**P2**	P3	P6	P8
P1	X	**6**	2	1	4
P2	X	X	2	4	3
P3	X	X	X	1	3
P6	X	X	X	X	1
P8	X	X	X	X	X

We thus have CLUSTER-2: {P_1, P_2}

Stage 3: Choose the cell with next maximum value from Table.5. Here (P_3, P_8) are having value 3. Group these cells to form a cluster containing projects (P_3, P_8).

Table 5. Reduced Similarity Matrix obtained from Stage 2.

	P3	P6	P8
P3	X	1	**3**
P6	X	X	1
P8	X	X	X

So the cluster formed is CLUSTER-3: {P_3, P_8}

Stage 4 : The final cluster formed is CLUSTER-4: {P_6}

Table 6. Reduced Similarity Matrix obtained from Stage 3.

	P6
P6	X

If we have a new project with parameters specified, we can test to which cluster the new project is similar and then can estimate the nature of the project to perform cost estimation.

So the clusters formed are

CLUSTER-1: {P_4, P_5, P_7, P_9}.

CLUSTER-2: {P_1, P_2}

CLUSTER-3: {P_3,P_8}

CLUSTER-4 : {P_6}

5. Conclusion

In this paper, we define a new similarity function to compute the similarity between any two software components or software requirement documents. An algorithm to cluster a set of software components is outlined which uses the proposed similarity function to find the similarity among any two software entities. The input to algorithm is a similarity matrix and the output is the set of cohesive clusters. The algorithm is applied to set of projects with predefined parameters and clustering process is carried out from which we can estimate the cost of the project from the properties of previously carried out projects. In future, the approach can be extended to classify the components using classifiers by applying fuzzy logic. The search complexity can be reduced by using the algorithm [11] where ever necessary as part of component retrieval.

References

1. V.Susheela Devi, M. Narasimha Murthy. Text Book on Pattern Recognition. An Introduction. University Press.
2. Congnan Luo, Yanjun Li, M. Chung. Text document clustering based on neighbours, Data & Knowledge Engineering, 2009; 68:1271–1288.
3. Tianming Hu, Sam Yuan Sung, Hui Xiong, Qian Fu. Discovery of maximum length frequent itemsets, Information Sciences, 2008; 178: 69–87.
4 Wen Zhanga,, Taketoshi Yoshida, Xijin Tang, Qing Wang. Text clustering using frequent itemsets, Knowledge-Based Systems, 2010; 23: 379–388
5. Wen Zhanga, Taketoshi Yoshida, Xijin Tang. A comparative study of TF*IDF, LSI and multi-words for text classification. Expert Systems with Applications, 2011; 38: 2758–2765.
6. Vincent Labatut and Hocine Cherifi. Accuracy Measures for the Comparison of Classifiers, ICIT 2011 The 5th International Conference on Information Technology.
7. Jung-Yi Jiang et.al A Fuzzy Self-Constructing Feature Clustering Algorithm for Text Classification. IEEE Transactions on Knowledge and Data Engineering, 2011; 23(3).
8. Melita Hajdinjak, Andrej Bauer. Similarity Measures for Relational Databases, Informatica, 2009; 33:143–149.
9. 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: 207–216.
10. 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: 436–442.
11. Radhakrishna.V, C.Srinivas, C.V.GuruRao. High Performance Pattern Search algorithm using three sliding windows, International Journal of Computer Engineering and Technology, 2012; 3: 543-552.
12. V.Susheela Devi, M. Narasimha Murthy. Text Book on Pattern Recognition. An Introduction. University Press.
13. Salim kebir, Abdelhak-djamel seriai, Sylvain Chardigny. Comparing and Combining Genetic and Cluster Algorithms for Software Component Identification, in the Proceedings of the ACM Fifth International Conference on Computer Science and Software Engineering, 2012: 1-8.
14. Ronaldo.C.Veras, Silvio R.L.Meira, Adriano L.I. Oliveira, Bruno J.M.Melo. Comparitive study of clustering techniques for the organisation of software repositories, in the proceedings of 19th IEEE International Conference on Tools with Artificial Intelligence.
15. Vangipuram Radhakrishna, C.Srinivas, C.V.GuruRao. Document Clustering Using Hybrid XOR Similarity Function for Efficient Software Component Reuse. Procedia Computer Science, 2013; (17): 121-128.
16. Divya Kashyap, A. K. Misra. Software development cost estimation using similarity difference between software attributes. ISDOC '13: Proceedings of the 2013 International Conference on Information Systems and Design of Communication.
17. Chintakindi Srinivas, Vangipuram Radhakrishna, C.V. Guru Rao. Clustering Software Components for Program Restructuring and Component Reuse Using Hybrid XNOR Similarity Function. Procedia Technology, 2014 ; (12): 246-54.