Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 19 (2015) 866 – 873

Procedia Technology

8th International Conference Interdisciplinarity in Engineering, INTER-ENG 2014, 9-10 October 2014, Tirgu-Mures, Romania

Software component clustering and classification using novel similarity measure

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

[a]*Department of Computer Science & Engineering, Kakatiya Institute of Technology, Warangal, India*
[b]*Department of Information Technology, VNR VJIET (Autonomous), Hyderabad, India*
[c] *Principal & Professor of Computer Science & Engineering, S.R.Engineering College (Autonomous), Warangal, India*

Abstract

The similarity measures such as Euclidean, Jaccard, Cosine, Manhattan etc present in the literature only consider the count of the features but does not consider the feature distribution and the degree of commonality. There is a significant research carried out for designing new similarity measures which can accurately find the similarity between any two software components. The distribution of component features in the software components has important contribution in evaluating their degree of similarity. This is the key idea for the design of the proposed measure. The main objective of this research is to first design an efficient similarity measure which essentially considers the distribution of the features over the entire input. We then carry out the analysis for worst case, average case and best case situations. The proposed measure is Gaussian based and preserves the properties of Gaussian function and can be used for clustering and classification of software components.

Peer-review under responsibility of "Petru Maior" University of Tirgu Mures, Faculty of Engineering

Keywords: software components; similarity; component vector; clustering.

1. Introduction

Clustering is one of the topics which have achieved a lot of practical importance from the researchers and also from the perspective of the software industry. The significance for clustering approach comes from the need of

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

Peer-review under responsibility of "Petru Maior" University of Tirgu Mures, Faculty of Engineering
doi:10.1016/j.protcy.2015.02.124

decision making such as classification, prediction, component search and retrieval and is thus widely used in many practical domains such as text classification, bioinformatics, medicine, image Processing. We can define Clustering as the process of grouping similar set of patterns together [20].

The input to software component clustering algorithm may be any set of entities or patterns or software components. The output of clustering algorithm will be a partition of cohesive groups. The representations or descriptions of clusters so formed shall be used in decision making such as which software component or pattern need to be selected. In view of software engineering, all the components within the same cluster have high cohesion and low coupling. Software component clusters can be treated as highly cohesive groups with low coupling which is the desired feature. 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 [20]. 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 method called Maximum Capturing is proposed for document clustering [19]. Maximum Capturing includes two procedures: 1. constructing document clusters and 2. Assigning cluster topics. The search complexity can be reduced by using the algorithm [21] where ever necessary as part of component retrieval.

2. Related Works

In [1] Ahlgren and Colliander study the similarity among documents in the context of science mapping. Their study results with the experimentation results showing that the citation only methods performed worse than the text only method. However their study did not consider the term frequency-inverse document frequency approach. In [2], Renata H. S. Reiser et.al proposes a fuzzy based exclusive or functions and discusses commutative and associative properties. The work of Marius and Liviu in [3] captures the stylistic similarity between texts. The authors consider the principal component analysis and statistical cluster analysis. In [4], authors represent the documents as topic maps. In other words, they use the concept of topic maps to improve the semantic similarity value to cluster documents. The similarity measure is obtained by computing correlation between document pairs. Liao and vemuri [5], applied the approach of clustering and classification to intrusion detection by using k-nearest neighbor. Their work is concerned with detecting if a program behaves normally or abnormally. In [7] the authors use a rank based approach for finding similarity. Ravi Shankar et.al studies problems that occur while clustering dynamic data [8]. The problem of handling dynamic data is that we need to consider both the history cost and quality of the cluster formed and is also an emerging research area. The main contribution to this paper is the design of novel similarity measure which has the property of Gaussian function. The Similarity measure defined is a function of commonality function [18, 24]. The Proposed similarity measure preserves the property of Gaussian function and considers the distribution of a feature over all software components in the input dataset.

3. Similarity Measure

The objective of the proposed similarity measure is to consider the common features and their distribution in the entire input dataset. We define the function Φ as a function of two component features by modifying the basic XNOR function. The function Φ maps its input to a output value of 0, 1 or U as defined in the Table.1.

Table 1. Function Φ defined by modifying basic XNOR function

C_{im}	C_{jm}	$\Phi(C_{im}, C_{im})$
0	0	U
0	1	0
1	0	0
1	1	1

Here, C_{im} and C_{jm} are binary variables indicating the presence or absence of the m^{th} component feature in i^{th} and

j^{th} components respectively. The presence of a component feature is defined equal to 1 and its absence is defined as 0.

We define the component vector for any two software components C_i and C_j as

$$CV\ [C_i, C_j] = [\Phi(C_{i1}, C_{j1}), \Phi(C_{i2}, C_{j2}), \Phi(C_{i3}, C_{j3}) \ldots\ldots\ldots\ldots \Phi(C_{ik}, C_{jk})] \quad (1)$$

where $\Phi(C_{ik}, C_{jk})$ is function defined in table.1

The similarity measure for two software components is given by the equation below

$$CSIM = \frac{(1+\mu)}{(1+\Delta)} \quad (2)$$

where

$$\mu = \frac{\sum_{k=1}^{k=m} N(C_{ik}, C_{jk})}{\sum_{k=1}^{k=m} D(C_{ik}, C_{jk})} \quad (3)$$

$$N(C_{ik}, C_{jk}) = \begin{cases} 1 - e^{-\gamma^2} & ; \Phi(C_{ik}, C_{jk}) = 1 \\ -e^{\gamma^2} & ; \Phi(C_{ik}, C_{jk}) = 0 \\ 0 & ; \Phi(C_{ik}, C_{jk}) = U \end{cases} \quad (4)$$

with

$$\gamma = \frac{\Phi(C_{ik}, C_{jk})}{\sigma_k} \quad (5)$$

here , σ_k = standard deviation of k^{th} component feature in all components of the input.

and

$$D(C_{ik}, C_{jk}) = \begin{cases} 0 & ; \text{ if } \Phi(C_{ik}, C_{jk}) \text{ is U} \\ 1 & ; \text{if } \Phi(C_{ik}, C_{jk}) \text{ is not U} \end{cases} \quad (6)$$

Here, d_{ik} indicates presence or absence of the k^{th} feature in i^{th} document.

The values of $N(C_{ik}, C_{jk})$ and $D(C_{ik}, C_{jk})$ are used to measure the contribution of each feature in finding similarity.

3.1 Validation of the Proposed Measure

3.1.1 Best Case Scenario

For the best case situation, C_1 = {1, 1, 1, 1, 1…….m} and C_2 = {1, 1, 1, 1, 1…….m}.

The component Vector, CV_{12} is represented as CV_{12} = <1, 1, 1, 1, 1……..m>. The value of μ is computed using eq.4 and eq.6 as shown below

$$\mu = \frac{N(C_{i1}, C_{j1}) + N(C_{i2}, C_{j2}) + N(C_{i3}, C_{j3}) + \cdots . + N(C_{im}, C_{jm})}{D(C_{i1}, C_{j1}) + D(C_{i2}, C_{j2}) + D(C_{i3}, C_{j3}) + \cdots + D(C_{im}, C_{jm})}$$

$$= \frac{\left(1 - e^{-\gamma_1{}^2}\right) + \left(1 - e^{-\gamma_2{}^2}\right) + \left(1 - e^{-\gamma_3{}^2}\right) \ldots\ldots\ldots . . \left(1 - e^{-\gamma_m{}^2}\right)}{(1 + 1 + 1 \ldots\ldots\ldots . mtimes)}$$

$$= \frac{(1 + 1 + 1 \ldots mtimes) - \left(e^{-\gamma_1{}^2} + e^{-\gamma_2{}^2} + e^{-\gamma_3{}^2} \ldots . mtimes\right)}{m}$$

….] (7)

For the best case situation the values of σ_k for k = 1 to m , approaches zero. This makes the values of $e^{-\gamma_1{}^2}, e^{-\gamma_2{}^2}, e^{-\gamma_3{}^2} \ldots\ldots . e^{-\gamma_m{}^2}$ equal to 0.

This means the above equation (7) reduces to

$$\mu = \frac{m-0}{m} = \frac{m}{m} = 1 \quad (8)$$

In this case, the similarity measure is

$$CSIM = \frac{(1+\mu)}{(\lambda+1)} = \frac{(1+1)}{(1+1)} = 1. \quad (9)$$

The value of CSIM = 1 indicates both the software components are most similar to each other.

3.1.2 Worst Case Scenario

For the worst case C_1 = {0, 0, 0, 0, 0…….m} and C_2 = {0, 0, 0, 0, 0…….m}. The component Vector, CV_{12} is represented as CV_{12} = <U, U, U, U……..m times>. The value of μ is computed using eq.4 and eq.6 as shown below

$$\mu = \frac{N(C_{i1}, C_{j1}) + N(C_{i2}, C_{j2}) + N(C_{i3}, C_{j3}) + \cdots . + N(C_{im}, C_{jm})}{D(C_{i1}, C_{j1}) + D(C_{i2}, C_{j2}) + D(C_{i3}, C_{j3}) + \cdots + D(C_{im}, C_{jm})}$$

$$= \frac{(0 + 0 + 0 \ldots m\ times)}{(0 + 0 + 0 . . m\ times)} \quad (10)$$

Such a situation does not arise practically though we assume theoretically because the at least one of the features will be present. However, if N=D=0 for all component features, we make it -1 indicating the negative maxima.

In this case, the similarity measure is

$$CSIM = \frac{(1+\mu)}{(\lambda+1)} = \frac{(-1+1)}{(1+1)} = 0. \quad (11)$$

The value of CSIM = 0 indicates that the two software components are least similar to each other or dissimilar w.r.t each other.

3.1.3 Average Case Scenario

For the average case situation C_1 = {1, 0, 1, 0, 1…….m times} and C_2 = {0, 1, 0, 1, 0…….m times}. The component Vector, CV_{12} is represented as CV_{12} = <1, 1, 1, 1……..m times>. The value of μ is computed using eq.4 and eq.6 as shown below

$$\mu = \frac{N(C_{i1}, C_{j1}) + N(C_{i2}, C_{j2}) + N(C_{i3}, C_{j3}) + \cdots . + N(C_{im}, C_{jm})}{D(C_{i1}, C_{j1}) + D(C_{i2}, C_{j2}) + D(C_{i3}, C_{j3}) + \cdots + D(C_{im}, C_{jm})}$$

$$= \frac{\left(-e^{-\gamma_1{}^2}\right) + \left(-e^{-\gamma_2{}^2}\right) + \left(-e^{-\gamma_3{}^2}\right) \ldots\ldots\ldots \left(-e^{-\gamma_m{}^2}\right)}{(1 + 1 + 1 \ldots\ldots\ldots mtimes)}$$

$$= \frac{-\left(e^{-\gamma_1{}^2} + e^{-\gamma_2{}^2} + e^{-\gamma_3{}^2} \ldots . mtimes\right)}{m}$$

Assuming the values of $e^{-\gamma_1{}^2}, e^{-\gamma_2{}^2}, e^{-\gamma_3{}^2} \ldots . e^{-\gamma_m{}^2}$ are all the same, then we have the above equation reduced to

$$\mu = \frac{-me^{-\gamma^2}}{m} \tag{12}$$

The value of $-e^{-\gamma^2}$ lies between 0 and -1 making the domain of similarity between (-1, 0). The negative similarity value is used to bias the presence-absence feature in the dataset.

In the case, the similarity measure is

Case 1: $\gamma = 0$.

The value for similarity measure denoted by CSIM is now given by

$$\text{CSIM} = \frac{(1-e^{-\gamma^2})}{(1+1)} = \frac{(1-1)}{(1+1)} = 0 \tag{13}$$

Case 2: $\gamma = \infty$.

Practically it is not infinite but for a particular very high value of γ the value of $e^{-\gamma^2}$ approaches zero. Then the value for S is

$$\text{CSIM} = \frac{(1-e^{-\gamma^2})}{(1+1)} = \frac{(1-0)}{(1+1)} = 0.5 \tag{14}$$

From the above two cases, the value of similarity measure in the average case is between always between 0 and 0.5 for the average case.

4. Case Study

For simplicity, we show the trace of computation by considering 9 components with feature set consisting of 5 component features as in Table.2. The Table.2 below shows the component-feature matrix in the binary form.

Table 2. Component-feature matrix in Binary Form

	Feature-1	Feature-2	Feature-3	Feature-4	Feature-5
Component-1	1	1	1	0	0
Component-2	0	0	1	1	1
Component-3	0	1	1	1	1
Component-4	1	1	1	0	1
Component-5	0	0	1	1	0
Component-6	1	1	0	0	1
Component-7	1	1	0	1	0
Component-8	0	1	0	1	0
Component-9	0	1	1	0	1

The sample computation of similarity values of component-1 w.r.t components 2 trough 9 is only shown below because of space constraint. The value of Δ is assumed as 1 for the purpose of biasing the similarity measure. Here Nr and Dr indicates Numerator and Denominator of the function μ respectively. In section 4.1 each Component-i is represented by i for simplicity and ease of use.

4.1 Component-Component Similarity computation

<1-2>: < 0, 0, 1, 0, 0>
Nr = -0.0273 -0.0058+ 0.9817-0.0273-0.0273 = 0.8940
Dr = 5
SMF= (0.1788+1)/(1+1)= 0.5894

<1-3>: <0, 1, 1, 0, 0>
Nr = -0.0273 + 0.9942+0.9817-0.0273-0.0273 =1.8940
Dr = 5
SMF = 0.6894

<1-4>: < 1, 1, 1, U, 0>
Nr =0.9727 +0.9942+0.9817+0-0.0273 = 2.9213
Dr = 4
SMF= (0.7303+1)/(1+1)=0.8652

<1-5>: < 0, 0, 1, 0, U>
Nr = -0.0273-0.0058+0.9817-0.0273+0=0.9213
Dr = 4
SMF= (0.2303+1)/(1+1)=0.6152

<1-6>: < 1, 1, 0, U, 0>
Nr =0.9727 +0.9942-0.0183 +0-0.0273=1.9213
Dr = 4
SMF= (0.4803+1)/2=0.7402

<1-7>: < 1, 1, 0, 0, U>
Nr = 0.9727 +0.9942-0.0183-0.0273+0=1.9213
Dr = 4
SMF= (0.4803+1)/2 = 0.7402

<1-8> = <0, 1,0, 0, U>

Nr = -0.0273+0.9942-0.0183-0.0273+0=0.9759
Dr = 4
SMF= 0.6220

<1-9> =<0, 1, 1, U, 0>
Nr = -0.0273+0.9942+0.9817+0-0.0273=1.9213
Dr = 4
SMF= 0.7402

Thus Component-1 is more similar to the Component-4 with value of 0.8652 or 86.52% similarity. The final set of Clusters formed after applying algorithm in [18, 20-24] are Cluster-1: < 1, 2, 3, 4, 6, 9>; Cluster-2: <7, 8>; Cluster-3: <5>.

5. Conclusion

The objective of this research is to propose a software component similarity measure to find the similarity between any two software components so that the components may be reuse. The proposed measure considers the distribution of the features over the entire set of components so as to make it applicable for clustering and classification of software components. The components may be any software requirement documents, program modules. The similarity measure designed is analyzed and validated for the worst case, average case and best case situations. The proposed measure is Gaussian based and hence is more accurate as it considers the distribution of component feature in each software component of the input set.

References

[1] Per Ahlgren, Cristian Colliander.Document–document similarity approaches and science mapping: Experimental comparison of five approaches, Journal of Informatics 2009; 3:49-63.
[2] Benjamin C. Bedregal, Renata H. S. Reiser, Gracaliz P. Dimuro. Xor-Implications and E-Implications: Classes of fuzzy implications based on fuzzy Xor, Electronic Notes in Theoretical Computer Science. URL: www.elsevier.nl/locate/entcs
[3] Marius Popescu, Liviu P. Dinu. Comparing Statistical Similarity Measures for Stylistic Multivariate Analysis, Proc. of Intl. Conference RANLP 2009 - Borovets, Bulgaria, p. 349–354
[4] Muhammad Rafi, Mohammad Shahid Shaikh.An improved semantic similarity measure for document clustering based on topic maps, http://arxiv.org/ftp/arxiv/papers/1303/1303.4087.pdf
[5] Yihua Liao, V. Rao, Vemuri. Use of K-Nearest Neighbor classifier for intrusion detection, Computers and Security 2002, 21(5): 439-48.
[6] William Webber, Alistiar Moffat, Justin Zobel. “Similarity Measure for Indefinite Rankings, ACM Transactions on Information Systems 2010; 28(4): 1–34.
[7] Ravi Shankar, Kiran G V R, Vikram Pudi. Evolutionary Clustering using Frequent Itemsets, proceedings of ACM StreamKDD’10, July 25, 2010, Washington, DC, USA.
[8] Sung-Hyuk Cha. Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions, International Journal of Mathermatical Models and methods in applied sciences 2007; 1(4):300-307.
[9] Raman Arora, Maya R. Gupta, Amol Kapila, Maryam Fazel. Similarity-based Clustering by Left-Stochastic Matrix Factorization, Journal of Machine Learning Research 2013; 14(7):1715- 1746
[10] Selim Mimaroglu, Dan A. Simovici. Clustering and Approximate Identification of Frequent Item Sets, American Association for Artificial Intelligence, 2007.
[11] Sumit Goswami, Mayank Singh Shishodia. A Fuzzy Based approach to Text Mining and Document Clustering, International Journal of Data Mining & Knowledge Management Proc. 2013; 3(3):43-52.
[12] Si Quang Le, Tu Bao Ho. “ An association-based dissimilarity measure for categorical data”, Pattern Recognition Letters, Volume 26, 2005, Pg. 2549–2557.
[13] Sheau-Ling Hsieh, Wen-Yung Chang, Chi-Huang Chen, and Yung-Ching Weng. “Semantic Similarity Measures in the Biomedical Domain by Leveraging a Web Search Engine”, IEEE Journal of Biomedical and Health Informatics, Vol 17, No. 4, July 2013.
[14] Ning Liu et.al. “Learning Similarity Measures in Non-orthogonal Space”, Proceedings of ACM CIKM’04, November 8-13, 2004, Washington D.C., U.S.A.
[15] Yung-Shen Lin, Jung-Yi Jiang, and Shie-Jue Lee. “ A Similarity Measure for Text Classification and Clustering”, IEEE Transactions on Knowledge and Date Engineering,2013.
[16] 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.
[17] 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.

[18] 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.
[19] Wen Zhanga,, Taketoshi Yoshida, Xijin Tang, Qing Wang. Text clustering using frequent itemsets, Knowledge-Based Systems 23 (2010) 379–388
[20] V.Susheela Devi, M. Narasimha Murthy. Text Book on Pattern Recognition. An Introduction. University Press.
[21] 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.
[22] Chintakindi Srinivas, Vangipuram Radhakrishna, C.V. Guru Rao. Clustering and Classification of Software Component for Efficient Component Retrieval and Building Component Reuse Libraries. Procedia Computer Science, 2014; (31): 1044-1050.
[23] Chintakindi Srinivas, Vangipuram Radhakrishna, and C. V. Guru Rao. 2013. Clustering Software Components for Component Reuse and Program Restructuring. In Proceedings of the Second International Conference on Innovative Computing and Cloud Computing (ICCC '13). ACM, New York, NY, USA, Pages 261 , 6 pages. DOI=10.1145/2556871.2556933 http://doi.acm.org/10.1145/2556871.2556933.
[24] Vangipuram Radhakrishna, Chintakindi Srinivas, and C. V. GuruRao. 2014. A modified Gaussian similarity measure for clustering software components and documents. In Proceedings of the International Conference on Information Systems and Design of Communication (ISDOC '14). ACM, New York, NY, USA, 99-104. DOI=10.1145/2618168.2618184 http://doi.acm.org/10.1145/2618168.2618184