Australian Journal of Basic and Applied Sciences
2018 Sept; 12(9): pages 41-46
DOI: 10.22587/ajbas.2018.12.9.7
Research Article
AENSI Publications
Journal home page: www.ajbasweb.com

An Efficient Model for Clustering Source Code Documents for Feature Location

[1]A.S. Baby Rani and [2]A.R.Nadira Banu Kamal

[1]*Assoc. Prof. in Computer Science, Sri Meenakshi Govt. Arts College for Women(Autonomous), Madurai, Tamil Nadu, India*
[2] *Principal, Mohammed Sathak Hamid College of Arts and Science for Women, Ramanathapuram, Tamil Nadu, India*

Correspondence Author: A.S. Baby Rani, Assoc. Prof. in Computer Science, Sri Meenakshi Govt. Arts College for Women(Autonomous), Madurai, Tamil Nadu, India
Tel: 918754021617
E-mail: asrani18@yahoo.com

Received date: 12 June 2018, **Accepted date:** 4 September 2018, **Online date:** 13 September 2018

Abstract

Text mining is widely used for many Software Engineering tasks and clustering is applied to group the related documents for feature location. With the proliferation of software applications in the Agile Development environment, it is vital and challenging to document and maintain the source code artifacts for maintenance and support. An automatic document summarization that can aid for feature location can be helpful for many software engineering tasks. In this research work, the unsupervised machine learning technique, Latent Dirichlet Allocation (LDA) modeling is used to model the source code documents with the entire textual content as text corpus and k-Means clustering is applied to cluster the documents.The inherent relationship of the source code documents is harnessed by clustering on text corpus with formal contexts used in the source code documents and this model is mapped upon LDA model with cluster analysis measures to improve the clustering. Three open source software systems namely JEdit, ArgoUML and JabRef are used for the case study and the results are presented. This work provides a fully automated model to extract features and efficiently store in its meta-form.

Key words: Text Mining, Machine Learning, LDA, Clustering, Cluster Analysis, Feature Location.

INTRODUCTION

In the present scenario of agile and rapid application development environment, myriad applications are proliferating and it is highly challenging to document and maintain such applications. Software engineering (SE) community had employed before formal methods for source code comprehension and maintenance. Now, text mining techniques are found to be appropriate to make source code analysis, as the present technology is not confined to a particular technical platform. When there is no proper decomposition of source code or documentation available, clustering is the common text mining technique applied to group the related documents. But, the variability of the text content and instability of the clustering techniques, it is required to choose among the various clustering algorithms that is suitable for an application with the proper validation measures. Due to the high dimensional space of text documents, it is essential to reduce it by applying any one of the Information Retrieval methods like Vector Space Modeling, Latent Dirichlet Allocation modeling, Latent Semantic Analysis etc. In this research work, we used LDA, an unsupervised machine learning technique as a representational model and k-Means clustering for grouping the software documents. LDA model can be drawn from the different contents of the entire source code like comments, identifiers, classes, datatypes, methods and a combination of one or more of these. In this work, LDA model with the entire source code (except comments) is analysed and improvised with that of a model that comprise only the attributes or class names alone with the objective of deriving a meta-model that can be used for further analysis.

1.1 *Literature Review:*

Feature location is the fundamental activity in carrying out many SE tasks like program comprehension, bug localization, trace visualization, impact analysis, et al .In the earlier researches, formal methods were used for SE tasks and to mention a few, function point analysis was applied by Ramkrishna Chatterjee et al., (2001), structural dependency graph analysis by Robillard and Murphy (2002), and formal concept analysis by Kim Mens and Tom Tourwe (2005).Recently text mining concepts are gaining popularity among the SE research community as mentioned by Bogdan Dit, et al.,(2013) in their survey. Annibale Panichella, et al.,(2013) applied Latent Dirichlet Allocation modeling in their research work on finding the applicability text mining technique for feature location. In search of aspects or design patterns, where it is not compulsory to have module dependency, components of concern will be unnoticed with formal methods whereas text mining techniques can identify the components. Even though it is intuitive to apply the formal methods for a software system, text mining and Information Retrieval (IR) methods perform well in feature location. Also, the research study by Poshyvanyk et. al.,(2007a, 2007b), Bogdan Dit et al., (2013) reveals that combining text mining or IR method with formal methods yield better results than applying any of the methods independently.

The effectiveness of the applying LDA model depends on the parameters used, due to the variability of the contents. Mitchell and Mancoridis(2001a, 2001b) grouped the related components using clustering, the text mining technique, when there was no proper decomposition of documents available.In the literature, there are many clustering algorithms proposed as mentioned in the survey done by Fahad, et al.(2014) and attempts have been made to analyze and categorize them for a large number of applications.Clusters can be validated with the cluster validation measures as discussed by Yanchi Liu, et al.,(2010). The research study reveals that no clustering algorithm is suitable for all kinds of applications and hence text mining is combined with other formal methods for feature location. In this context, Formal Concept Analysis (FCA) is effectively used in many SE tasks like Feature Location using dynamic analysis by Eisenbarth, Koschke and Simon (2002), Software Product Line Engineering by Ra'Fat et al., (2013), code refactoring, impact analysis and more. In his work on software analysis, Gregor Snelting(2005) stated that the large size of feature space limits the construction of concept lattices for feature location in static analysis of source code. But with the latest technology

Citation: A.S. Baby Rani and A.R.Nadira Banu Kamal., An Efficient Model for Clustering Source Code Documents for Feature Location. Australian Journal of Basic and Applied Sciences, 12(9): 41-46. DOI: 10.22587/ajbas.2018.12.9.7

of big data framework, a parallel algorithm could be used to apply FCA on various clusters simultaneously rather working on the entire data as a whole. Hence, in this work, we try to cluster the documents with better distribution in various clusters so as to apply FCA on the clusters obtained. LDA is used to model the source code documents and K-Means clustering is applied on the resulting LDA model. With the objective of applying formal concept analysis on the clusters obtained, a cluster refinement algorithm is designed to improve the clustering using subspace clustering approach as explained by Anne Patrikainen and Marina Meila (2006).Three open source software systems namely JEdit, ArgoUML, JabRef are used for experimental study. Clusters are validated with the Silhouette Coefficient and weighted F-Measure.

The following sections are organized as follows. Section2 provides the necessary details about LDA and the measures used for analysis in this work. Section 3 describes the methodology and Section 4 explains about the Case Study undertaken. Section 5 discusses the results obtained and concludes with Section 6.

Background:

1.1 Latent Dirichlet Allocation Model:

Latent Dirichlet Allocation modeling is a generative probabilistic model to extract terms out of a corpus in an unsupervised machine learning approach based on *Naïve-Bayes* theorem. A detailed explanation of LDA is presented in the article written by D.M. Blei, et al., (2003). The basic idea is that documents are represented as random mixtures over latent topics where each topic is characterized by a probability distribution over terms or words. The co-occurrence of the terms derive the contextual meaning and are defined as topics.

LDA assumes the following generative process for each document $\mathbf{w}$ *in a corpus D:*

1. Choose $N \sim$Poisson(ξ).
2. Choose $\theta \sim$ Dir(α).
3. For each of the N word w_n ,

(a) Choose a topic $z_n \sim$Multinomial(θ)

(b) Choose a word w_n from $p(w_n| z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

w represents a document (a vector of words) where w = (w_1,w_2,…w_N)

α is the parameter of the Dirichlet distribution, technically α =(α_1, α_2,…, α_k), but unless otherwise noted, all elements of α will be the same.

z represents a vector of topics, where if the ith element of z is 1 then w draws from the ith topic.

β is a k x V topic by term probability matrix for each topic (row) and each term(column), where $\beta_{ij} = p(w_i = 1 | z_j = 1)$

1.2 Cluster Validation Measures:

1.2.1 Silhouette Coefficient:

The Silhouette Coefficient, as explained in their article by Annibale Panichella, et al., (2013), is computed for each document using the concept of centroids of clusters. Formally, let C be a cluster; its centroid is equal to the mean vector of all documents belonging to C:

$$\text{Centroid(C)} = \sum_{d_i \in C} d_i \, /|C| \tag{1}$$

Starting from the definition of centroids, the computation of the Silhouette coefficient consists of the following three steps:

- For document d_i, calculate the maximum distance from d_i to the other documents in its cluster. We call this value $a(d_i)$.
- For document d_i, calculate the minimum distance from d_i to the centroids of the clusters not containing d_i. We call this value $b(d_i)$.
- For document d_i, the Silhouette coefficient $s(d_i)$ is:

$$s(d_i) = \frac{b(d_i) - a(d_i)}{\max (a(d_i), b(d_i))} \tag{2}$$

The value of the Silhouette coefficient ranges between -1 and 1. A negative value is undesirable because it relates to the case where a(di) > b(di), i.e. a document in one cluster is closer to a document belonging to another cluster and hence it implies poor clustering.

For measuring the distance between documents squared Euclidean distance is used. The overall measure of the quality of clustering C = {C_1,C_2,…C_k} can be obtained by computing the mean Silhouette Coefficient of all documents as follows.

$$S(C) = \frac{1}{n} \sum_{i=1}^{n} s(d_i) \tag{3}$$

1.2.2 F-Measure:

To compute F-[1]Measure, each cluster is considered as the result of a query and each class is the desired set of documents for a query which is available in the gold set[1]. Now, the recall and precision of clusters can be calculated for each class. More specifically, for cluster j and class i,

Recall(i, j) = n_{ij}/n_i

Precision(i, j) = n_{ij} / n_j

Where n_{ij} is the number of members of class i in cluster j.

$$F(i, j) = \frac{2 * Recall(i,j) * Precision(i,j)}{Precision(i,j) + Recall(i,j)} \tag{4}$$

The overall value for the F-Measure is computed by taking the weighted average of all values for the F-Measure given by

$$F = \sum_i \frac{n_i}{n} \max (F(i,j)) \tag{5}$$

F-Measure for assessing the cluster quality is computed with the benchmarks available in the link[1] which is provided by Annibale Panichella, et al., (2013). The higher value of the F-Measure signifies the goodness of the clusters formed.

Methodology:

First, the source code is sliced into documents at method level granularity. The target text of the source code documents for analysis shall be the source code text (excluding keywords and special symbols), class and method names, comments, attributes, identifiers or a combination of these different sources. In this work, text corpus with the entire source code is used for clustering (Cluster1) the source code documents. With the message passing mechanism of the methods, attributes/classes used in the methods including formal parameters will dictate an intrinsic relationship among the methods with common attributes and hence text corpus with attributes alone is used as another source of clustering (Cluster2). The clusters formed by the entire source code is improved by the latter as explained below. A bipartite graph is devised with the clusters of Cluster1 as one set of nodes and are mapped on to the nodes or clusters of Cluster2 when the clusters contain documents in common. A cluster refinement algorithm is implemented to improve the clustering by applying the projection on the bipartite graph and subspace clustering. The cluster validation measures indicate the refined clusters are better than the clusters originally formed and these clusters shall be used as meta data for further information access and analysis. Schematic diagram of the proposed model is shown in Fig. 1.

Citation: A.S. Baby Rani and A.R.Nadira Banu Kamal., An Efficient Model for Clustering Source Code Documents for Feature Location. Australian Journal of Basic and Applied Sciences, 12(9): 41-46. DOI: 10.22587/ajbas.2018.12.9.7

1.3 Text Corpus:

Java source files (.java, .jar) of software systems under study are pooled out from the project archive. Text corpus is prepared by slicing the java program files at method level granularity. Text corpus for the other clustering is done by extracting the class/datatype names of formal parameter listing and declarative statements from the method body.

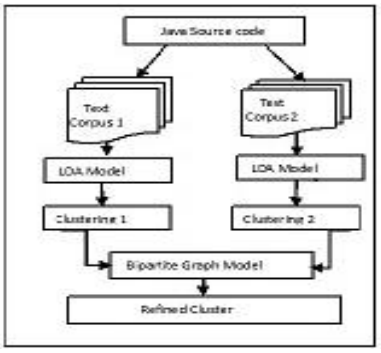

Fig. 1: Proposed Model.

3.2 Preprocessing:

The extracted files are preprocessed using the modified stop word removal algorithm to remove the keywords and common datatypes along with the prepositions and conjunctions existing in the source code documents. All special characters like {,},$,#, et. al., are removed. Then, stemming is applied to transform the words in different forms to its common root with Porter's Stemming algorithm. All the unique words (stemmed) in various documents are collected and indexed which is used to compute the term frequency vector for each document. All the datatypes/ classes except primitive datatypes like int,boolean,char, etc. and common class like String are discarded and classes/datatypes are separately indexed.

3.3 LDA Modeling:

The high dimensional space of the source code is reduced using the unsupervised machine learning, probabilistic modeling called LDA modeling with the appropriate smoothing parameters α, β & k which is explained in section 2.1. Fast Gibbs sampling algorithm is applied using GibbsLDA++ (2003) on the term-by-document matrix which reduces the term-by-document space of the text corpus into topic-by-document space. It also provides the vector of topic assignment to each term and the topic-by-term matrix that constitutes the probability distribution of terms over a particular topic.

3.4 K-Means Clustering:

The related documents are clustered with k-Means clustering using the topic-by-document representation of the source code documents for both text corpora. The cluster size can be chosen randomly and in this experimental study cluster size of 200 is used.The resulting clustering is validated with the internal validation measure, Silhouette Coefficient, that exhibits a very high value for singleton clusters or clusters with few documents. The experimental evaluation of document distribution in various clusters brings out the large clusters i.e. clusters with more documents which indicates a poor clustering. With the observation of very large clusters and many singleton clusters, a cluster refinement algorithm is proposed to improve the clustering using a bipartite graph model.

3.5 Bipartite Graph Model:

A Bipartite graph is constructed by defining a set of nodes with Cluster1 and another set of nodes with Cluster2. Each node is defined by the cluster elements(documents in that cluster) and total number of elements in the cluster. The total number of documents in each node is defined as its density. Edges are added between nodes when it is having documents in common and edge weight is defined by the total number of similar documents. The density measure of the nodes is used for finding clusters that need refinement. The projection on the bipartite graph can easily move a document of singleton cluster into another appropriate cluster.

3.6 Cluster Refinement Algorithm:

Let $C = \{C_1, C_2, \ldots, C_k\}$ be the group of clusters formed from a clustering and n_i be the no.of documents in the cluster C_i.
$S = \{ C_i \mid C_i \in C , n_i = 1 \}$ be the set of Singleton clusters.
$L = \{ C_i \mid C_i \in C , n_i > \mu \}$ be the set of Large clusters where μ is the average size of a cluster.

Step 1. Merge the Singleton Clusters S with nearest neighbor:

For each cluster C_i in S,

1. Find out the document ***d*** in C_i
2. Find the nearest neighbor ***x*** of ***d*** with the projection on bipartite graph.
3. Add the document ***d*** into the cluster that contains ***x***.
4. Remove the cluster Ci

Step 2. Apply subspace clustering for the Large clusters L:

1. Find out the set of documents {***d***} in L.
2. Remove the features that are not present in {***d***}.
3. Eliminate the insignificant features of the document with term quality variance as mentioned by Berry(2004).
4. Reduce the feature space by removing the common features i.e. the features present in almost all the documents.
5. Divide the set of documents into two disjoint subsets D_{L1} and D_{L2} by projecting on the prevalent and non-prevalent features respectively. Prevalence refers to the presence of feature in the large number of documents.
6. Apply K-Means (or any other) clustering on D_{L1} and D_{L2}.

By following the Steps 1 & 2, a refined cluster with better document distribution will be obtained.

Case Study:

Three open source software systems[1] namely JEdit, ArgoUML and JabRef are taken for empirical evaluation. The characteristics of the Systems following preprocessing are given in Table 1.

Citation: A.S. Baby Rani and A.R.Nadira Banu Kamal., An Efficient Model for Clustering Source Code Documents for Feature Location. Australian Journal of Basic and Applied Sciences, 12(9): 41-46. DOI: 10.22587/ajbas.2018.12.9.7

Table 1: System Characteristics.

System	Size KLOC	No. of Methods	No. of Terms	No. of Attributes
JEdit	104	6347	3203	503
ArgoUML	149	11000	12005	1233
JabRef	27	4755	3681	500

In the research wok of Annibale Panichella, et al. (2013), they identified that the effectiveness of LDA modeling depends on the parameters chosen for [α β,k].In finding optimal number of topics in applying LDA modeling,G.Maskeri , S.Sarkar and K.Heafield (2008), suggested from their experimental study that 300 to 400 topics are sufficient to model the source code documents .Hence, the text corpus is modeled with fine tuned LDA parameters that are shown in Table 2.

Table 2: LDA Configuration parameters.

System	α		β		K - No. of Topics	
	Cluster1	Cluster2	Cluster1	Cluster2	Cluster1	Cluster2
JEdit	0.5	0.5	0.5	0.5	400	300
ArgoUML	0.125	0.5	0.05	0.5	300	300
JabRef	0.125	0.5	0.1	0.5	300	300

K-Means clustering is applied on the LDA model to cluster the documents. A bipartite graph is constructed with Cluster1 and Cluster2. The clusters formed are inspected and the histogram chart shows there are many singleton clusters. The document distribution in various clusters are shown in Fig. 2,3,& 4 for JEdit, ArgoUML and JabRef respectively. From the cluster analysis, it has been observed that there are many singleton clusters and few very large clusters. Hence, cluster refinement algorithm as explained in section 3.6 is used to refine the clusters further.

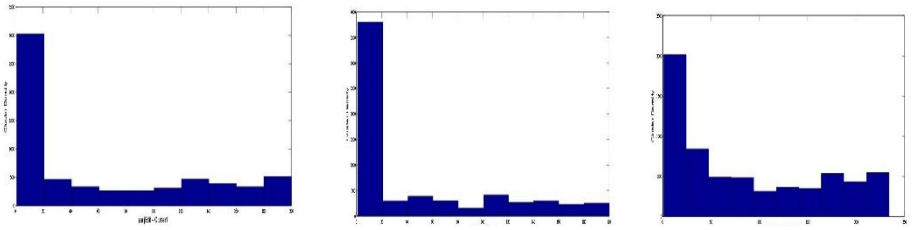

Fig. 2: Histogram Chart of Clustering of JEdit.

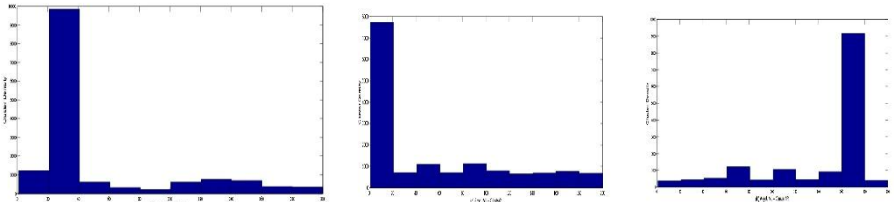

Fig. 3: Histogram Chart of Clustering of ArgoUML.

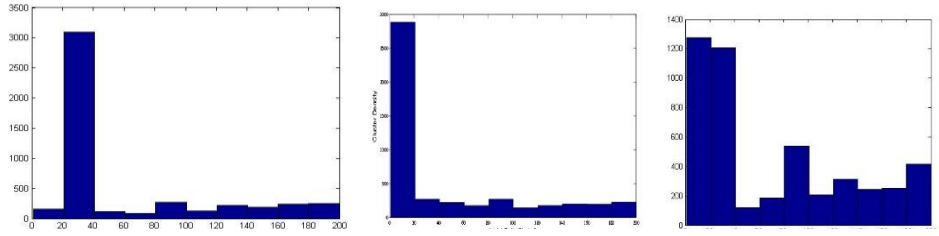

Fig. 4: Histogram Chart of Clustering of JabRef.

RESULTS & DISCUSSION

The boxplot of the distribution of mean Silhouette Coefficients of Cluster1, Cluster2 and the refined cluster named as Cluster1R of the software systems under study are shown in Fig. 5,6,&7. In JEdit, Cluster1 shows a high positive value in 3rd and 4th quartile due to singleton clusters or clusters with very few documents. In Cluster1R of JEdit, even though the variability is high, mean silhouette coefficients signify even distribution of documents over the various clusters. In Cluster2, a higher value of lower whisker and a larger quadrant of 3rd quartile, proves that the clusters formed for Cluster2 is better than Cluster1 and Cluster1R. Still, a higher positive value ranging from 0.8 to 1 indicates the existence of singleton clusters. Similarly, in ArgoUML and JabRef, Cluster1R exhibits an even distribution of

Citation: A.S. Baby Rani and A.R.Nadira Banu Kamal., An Efficient Model for Clustering Source Code Documents for Feature Location. Australian Journal of Basic and Applied Sciences, 12(9): 41-46. DOI: 10.22587/ajbas.2018.12.9.7

documents over Cluster1 and it has very few singleton clusters. The lower whisker of the boxplot of Cluster2 is relatively higher than that of Cluster1 and Cluster1R for all the three software systems. This shows that the clusters of Cluster2 (text corpus drawn from datatypes/classes of the documents) exhibit good coherence.

Table 3: Cluster Validation Measures.

System	No. of Queries	Cluster 1		Cluster 2		Cluster 1 (Refined)	
		Mean Silhouette Coefficient	Weighted F-Measure	Mean Silhouette Coefficient	Weighted F-Measure	Mean Silhouette Coefficient	Weighted F-Measure
JEdit	150	0.333	0.027	0.3132	0.02	0.0793	0.0294
ArgoUML	91	0.2485	0.04	0.1578	0.011	0.0369	0.042
JabRef	39	0.4736	0.037	0.4008	0.018	-0.0019	0.038

The clusters are validated with the internal and external cluster validation measures and tabulated in Table 3.

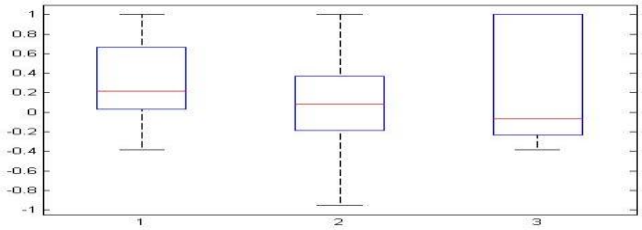

Fig. 5: Boxplots of Mean Silhouette Coefficient- JEdit (Cluster1,Cluster1R,Cluster2).

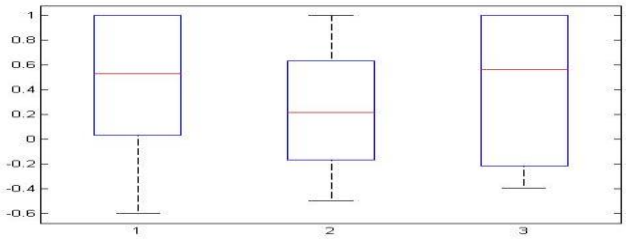

Fig. 6: Boxplots of Mean Silhouette Coefficient- ArgoUML (Cluster1,Cluster1R,Cluster2).

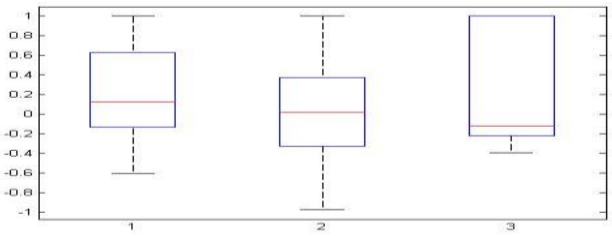

Fig. 7: Boxplots of Mean Silhouette Coefficient- JabRef (Cluster1,Cluster1R,Cluster2).

In the literature, the aggregate of mean Silhouette Coefficients over all clusters is used for validating cluster quality. In this study of cluster analysis, it reveals that the distribution of mean silhouette coefficients shall validate better than its aggregate value since there is no correlation between silhouette coefficient and weighted F-Measure. The boxplot of mean silhouette coefficient of refined cluster signifies the improvement of cluster quality and document distribution over the clusters.

The weighted F-Measure shows a little improvement over the original clustering but it is promising that reformulation of clusters does not decline the existing quality. Also, clusters with reasonable size and better distribution can be modeled into concept lattices and formal concept analysis can be applied for feature analysis and feature location.

Conclusion:

Feature location, one of the basic tasks in Software Engineering is being done with the text mining techniques. In the absence of proper decomposition of components, clustering is used to group the related documents. Researches show tuning the parameters and honing the clustering shall improve its efficiency in locating features. This work gives a direction to generate better clustering that can be used as meta model for many SE tasks. We intend to apply formal concept analysis on this meta model for feature location in the next work.

REFERENCES

Anne Patrikainen and Marina Meila, 2006. Comparing subspace clusterings, IEEE Transactions on Knowledge and Data Engineering, 18(7).

Citation: A.S. Baby Rani and A.R.Nadira Banu Kamal., An Efficient Model for Clustering Source Code Documents for Feature Location. Australian Journal of Basic and Applied Sciences, 12(9): 41-46. DOI: 10.22587/ajbas.2018.12.9.7

Annibale Panichella, Bogdan Dit, Rocco Oliveto, Massimilano Di Penta, Denys Poshynanyk, Andrea De Lucia, 2013. How to effectively use topic models for Software Engineering tasks? An approach based on Genetic Algorithms ,IEEE ICSE pp: 522-531.

Michael W. Berry , 2004. Survey of Text Mining, Springer, pp: 81-83.

Blei, D.M., A.Y. Ng, M.I. Jordan, 2003. Latent Dirichlet Allocation, The Journal of Machine Learning Research, 3: 993–1022.

Bogdan Dit, Meghan Revelle, Malcom Gethers, Dennys Poshyvanyk, 2013. Feature Location in source code: A Taxonomy and Survey Wiley Journal of Software, Evolution and Process , 25(1): 53-95.

Eisenbarth, T., R. Koschke, D. Simon, 2002. "Locating Features in Source Code—Case Studies," available at http://www.bauhausstuttgart.de/bauhaus/papers/

Fahad, A., N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A.Y. Zomaya, S. Foufou, A. Bouras, 2014. A survey of clustering algorithms for Big Data: Taxonomy and Empirical analysis, IEEE Transactions on Emerging Topics in Computing, 2(3): 267-279.

Gibbs, L.D.A., 2003. http://gibbslda.sourceforge.net/. Int. Conf. on Research and Development in Information Retrieval, Toronto , Ontario, Canada, pp: 433-434.

Kim Mens, Tom Tourwe, 2005. Delving source Code with Formal Concept Analysis, Elsevier Journal of Computer Languages, Systems and Structures, 31(3-4): 183-197.

Martin, P., Robillard and Gail C. Murphy, 2002. Concern Graphs: Finding and Describing Concerns Using Structural Program Dependencies , Proc. of the 24th International Conference on Software Engineering.

Maskeri, G., S. Sarkar, K. Heafield, 2008. Mining Business Topics in Source Code using Latent Dirichlet Allocation, In Proc. 1st India Software Engineering Conference, Hyderabad, India, pp: 113-120.

McMillan, C., M. Grechanik, D. Poshyvanyk, Chen Fu, 2011. Exemplar: A Source Code Search Engine for Finding Highly Relevant Applications, IEEE Transactions on Software Engineering, 38(5).

Mitchell, B.S., S. Mancoridis, 2001. Comparing the decompositions produced by software clustering algorithms using similarity measurements, IEEE International Conference on Software Maintenance

Mitchell, B.S., S. Mancoridis, 2001. CRAFT: A Framework for evaluating Software Clustering Results in the absence of Benchmark Decompositions , IEEE Eighth Working Conference on Reverse Engineering

ThePorter Stemming algorithm, http://tartarus.org/~martin/PorterStemmer/.

Poshyvanyk, D., Y. Gael-Gueheneuc, A. Marcus, G. Antoniol, V. Rajlich, 2007. Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval, IEEE Trans. on Softw. Eng., 33(6): 420–432.

Poshyvanyk, D., A. Marcus, 2007. Combining Formal Concept Analysis with Information Retrieval for Concept Location in Source Code, IEEE International Conference on Program Comprehension.

Ramkrishna Chatterjee, Barbara G. Ryder, William A. Landi, 2001. Complexity of Points-To Analysis of Java in the Presence of Exceptions , IEEE Transactions on Software Engineering , 27(6) : 481-512

Ra'Fat Ahmad Al-Msie'Deen, Abdelhak-Djamel Seriai, Marianne Huchard, Christelle Urtado, Sylvain Vauttier, 2013. Mining features from the object-oriented source code of Software Variants by combining lexical and structural Similarity, 14th International Conference on Information Reuse and Integration, Las Vegas, NV,United States, pp: 586-593.

Gregor Snelting, 2005. Concept Lattices in Software Analysis , Springer,"Formal Concept Analysis" pp: 272-287.

Yanchi Liu, Zhongmou Li, Hui Xiong, Xuedong Gao, Junjie Wu, 2010. Understanding of Internal Clustering Validation Measures, IEEE International Conference on Data Mining.