Journal of Software Engineering and Applications, 2012, 5, 983-990
http://dx.doi.org/10.4236/jsea.2012.512113 Published Online December 2012 (http://www.SciRP.org/journal/jsea)

ML-CLUBAS: A Multi Label Bug Classification Algorithm

Naresh Kumar Nagwani[1], Shrish Verma[2]

[1]Department of Computer Science & Engineering, National Institute of Technology Raipur, Raipur, India; [2]Department of Electronics and Telecommunication Engineering, National Institute of Technology Raipur, Raipur, India.
Email: nknagwani.cs@nitrr.ac.in, shrishverma@nitrr.ac.in

Received October 4th, 2012; revised November 6th, 2012; accepted November 17th, 2012

ABSTRACT

In this paper, a multi label variant of CLUBAS [1] algorithm, ML-CLUBAS (Multi Label-CLassification of software Bugs Using Bug Attribute Similarity) is presented. CLUBAS is a hybrid algorithm, and is designed by using text clustering, frequent term calculations and taxonomic terms mapping techniques, and is an example of classification using clustering technique. CLUBAS is a single label algorithm, where one bug cluster is exactly mapped to a single bug category. However a bug cluster can be mapped into the more than one bug category in case of cluster label matches with the more than one category term, for this purpose ML-CLUBAS a multi label variant of CLUBAS is presented in this work. The designed algorithm is evaluated using the performance parameters F-measures and accuracy, number of clusters and purity. These parameters are compared with the CLUBAS and other multi label text clustering algorithms.

Keywords: Software Bug Mining; Software Bug Classification; Bug Clustering; Classification Using Clustering; Bug Attribute Similarity; Multi Label Classification

1. Introduction

Classification algorithms in text mining can be categorized using the class labels assigned to each category. Using class labels as classification parameters there are two major categories of classification algorithms, the first category is single label algorithms and the other category is multi label classification algorithms. In multilabel categorization a text document may belong to one or more number of categories. In single-label classification, each document is mapped to exactly one category. Software bugs contains most of the important information as text. The algorithm CLUBAS (CLassification of software Bugs Using Bug Attribute Similarity) is presented in [1] for creating the categorized groups of similar bugs using the textual similarity of bug attribute. CLUBAS is a single label text clustering based classification algorithm, where one bug cluster can be mapped to exactly one bug category. In this paper a multi label variant of CLUBAS, namely ML-CLUBAS (Multi Label-CLassification of software Bugs Using Bug Attribute Similarity) is presented.

2. The ML-CLUBAS Algorithm

In this section, the pseudo code and working of the ML-CLUBAS algorithm is presented. ML-CLUBAS is segmented into the five major steps just like CLUBAS. All the steps are same as CLUBAS except the ***step* 4** (***Mapping Clusters to Classes***), which transforms the CLUBAS algorithm to ML-CLUBAS algorithm. Like CLUBAS, it takes two parameter for performing the bug classification *i.e*. textual similarity threshold value (τ) and number of frequent terms in cluster label (**N**). The initial step in the *Extract Data*, where the bug records from a particular bug repository is retrieved and stored in the local system. The next step is *Pre-processing Step*, where the software bug records available locally in HTML or XML file formats are parsed and bug attributes and their corresponding values are stored in the local database. After this the stop words elimination and stemming is performed over the textual bug attributes summary (title) and description, which are used for creating the bug clusters. In the following step (*Clustering*), the pre-processed software bug attributes are selected for textual similarity measurement. Cosine similarity technique from symmetric [2] Java API (Application Programming Interface) is used for measuring the weighted similarity between a pair of software bugs. For all software bug pairs the weighted similarities are calculated and stored, using which the clusters are created. The clusters are created as follows—initially one cluster is created with a random bug, then if the similarity value for a paired bugs with this bug is less than the similarity threshold value (τ), then both of the bugs are mapped to the same clusters, otherwise the new cluster is created and the bug which is not belonging to the cluster is

mapped to new cluster. This similarity threshold value is one of the important parameters for the ML-CLUBAS algorithm. If the value of τ (similarity threshold value) is the high, then high similarity between the software bug attributes is expected for clustering and vice-versa.

The next step (*Cluster Label Generation*) is to generate the cluster labels using the frequent terms present in the bugs of a cluster. In this step the summary (title) and descriptions of all the software bugs belonging to a particular clusters are aggregated and frequent terms present in this aggregate text data is calculated and the **N** (where **N** is the number of frequent terms in labels and is an user supplied parameter) top most frequent terms are assigned to the clusters as the cluster labels. Mapping of the cluster labels to the bug categories using the taxonomic terms for various categories is carried out next (*Mapping Clusters to Classes*). In this step, the taxonomic terms for the entire bug categories are pre-identified and cluster label terms are matched with these terms. Matching of the terms indicates the belongingness of clusters to the categories. Here in ML-CLUBAS one bug cluster can belong to more than one bug category depending on the taxonomic term and cluster label match. On every match of these terms, the bug cluster can belong to the bug categories. The last step (*Performance Evaluation and Output Representation*) is generating the confusion matrix, using which various performance parameters like precision, recall, and accuracy is calculated. The precision and recall can be combined together to calculate f-measure, the formulas for these parameters is mentioned in the next section. Finally the cluster information is visualized and represented as the output of the ML-CLUBAS.

ALGORITHM ML-CLUBAS

Returns: a) Clusters consisting of similar bugs,
b) Categories of each cluster.
Arguments: τ—Similarity Threshold,
N—Number of frequent terms in cluster labels.

***Step* 0 (*Extract Data*):**
0a) Generate the numbers set **R** for bug data sources (bug-id range, randomly etc.).
0b) For each number $\mathbf{m} \in \mathbf{R}$, append it to the bug repository URL.
0c) Using URL programming, extract the HTML or XML page for the bug with the bug-id value as **m**.

***Step* 1 (*Pre-processing Step*)**:
for-each bug record retrieved from the bug repository.
1a) Parse and extract the bug attributes from each bug file.
1b) Eliminate the **stop words** from bug summary, description and comments.
1c) Apply **stemming** to the textual attributes bug summary, description and comments.

***Step* 2 (*Clustering*)**:
2a) For each pair of bugs B_i and B_j, calculate the textual similarity between the attributes summary and description, using the similarity weights W_S and W_D such that the similarity value is normalized to 1, *i.e.* $\mathbf{W_S + W_D = 1}$.
2b) $\mathbf{Sim\ (B_i, B_j) = W_S \times Sim\ (B_{i\text{-}summary}, B_{j\text{-}summary}) + W_D \times Sim\ (B_{i\text{-}description}, B_{j\text{-}description})}$.
2c) IF $\mathbf{Sim\ (B_i, B_j) > \tau}$ **THEN** Assign $\mathbf{B_i}$, $\mathbf{B_j}$ to same cluster **ELSE** Create a new cluster and Assign $\mathbf{B_j}$ to this cluster.

***Step* 3 (*Cluster Label Generation - Using Frequent Terms for a Cluster*)**:
For each cluster $\mathbf{C_i}$, get the lists of bugs belonging to this cluster.
3a) Extract the summary and description of these bugs.
3b) Concatenate this textual data to form the cluster text data.
3c) Calculate the **N** frequent terms $\mathbf{\{T_{i1}, T_{i2} \cdots T_{iN}\}}$ from each cluster text data, and assign them to these clusters as cluster labels.
3d) $\mathbf{Label\ (C_i) \leftarrow \{T_{i1}, T_{i2} \cdots TiN\}}$.

***Step* 4 (*Mapping Clusters to Classes*)**:
4a) For each cluster $\mathbf{C_i}$, get each term $\mathbf{T_{iK}}$ in the $\mathbf{Label\ (C_i)}$ (cluster label) and match it with the bug taxonomic terms. The match indicates the belongingness of cluster in that bug category. If matching occurs with taxonomic terms of two or more number of categories, then the cluster will belong to all of these categories.

***Step* 5 (*Performance Evaluation and Output Representation*)**:
5a) Generate the confusion matrix.
5b) Calculation of the performance parameters **Accuracy**, **Precision**, **Recall**, **F-Measure**, **Number of Clusters** and **Entropy**.
5c) Visualized representation of bug clusters and its labels.

3. Classifier Performance Evaluation

The accuracy and performance of prediction models for classification problem is typically evaluated using a confusion matrix. Various performance measures like accuracy and F-measure are derived from the confusion matrix. The formula's for the parameters is covered in the CLUBAS [1]. From clustering quality comparison point of view two parameters are important *i.e*. number of clusters and entropy.

Entropy

Entropy is the amount of information by which the knowledge about the classes increases, when clusters are increased. Entropy tends to increase with the number of clusters, it reaches maximum to $\log_2(N)$, where N is the number of clusters. Entropy is a measure of uncertainty associated with the random variables. Lower value of entropy indicates the better quality of clusters. In an ideal situation, if the software bug has a one to one mapping with a cluster, then the value of entropy will be zero [3,4]. Entropy is defined as follows:

$$\text{entrpoy} = -\sum P_i \log_2 P_i \quad (1)$$

where P_i is the probability of a document being in i^{th} cluster.

4. Implementation

Implementation is done using open source object oriented programming language Java, and MySql is taken as local data base management system, Weka [5] API is used for implementing the stemming and other classification algorithms for comparison. The multi map data structure is also used for calculations and storing the clusters information at run time.

4.1. Datasets and Sampling

The random software bug records are selected from four open sources online software bug repositories namely, Android [6], JBoss-Seam [7], Mozilla [8] and MySql [9]. Random sampling technique is used and the sample size of 1000 is taken for the experiments from these four repositories for the comparison of the classifiers.

4.2. Pre-Processing

After the software bug records are extracted and made available at local system, and then pre-processing of these records is performed. The pre-processing takes places in three stages: parsing, elimination of stop words and stemming [1].

4.3. Mapping Bug Clusters to Categories

The categorical terms are generated from the software bug clusters labels. The technique of generating these taxonomic terms from various bug repositories is given in [1]. Cluster labels are generated by computing the frequent terms present in the bug clusters, then these terms are matched against the taxonomic terms of various bug categories presented in [1], whenever there is a match of cluster label term and taxonomic term that indicates the category of the bug cluster, here in this case a bug cluster can belong to more than one bug category. Experiments are performed from binary classification (two bug classes) to eleven number of bug classes. ***Ten*** bug classes are mentioned in [1], if a software bug does not fall on these ***ten*** categories, then it is mapped to a special category "**Other**", *i.e.* the ***eleventh*** category of bug classes. In ML-CLUBAS a bug may fall on one or more categories from these ***eleven*** bug categories.

5. Comparison of CLUBAS, ML-CLUBAS and Other Similar Algorithms

The comparison of ML-CLUBAS is first performed with CLUBAS using the performance parameters accuracy and F-measure. Since up to cluster generation stage both CLUBAS and ML-CLUBAS are same, so same number of bug clusters are generated by both of these algorithms. ML-CLUBAS is further compared with the other multi label text clustering algorithms Lingo and STC using the parameters accuracy, F-measure, number of clusters and entropy. Lingo is proposed by Osinski *et al.* [10,11] for clustering search results, which uses the method of algebraic transformations of the term-document matrix and frequent phrase extraction using suffix arrays. Lingo is a popular web clustering algorithm and is commonly used for clustering the web search results. Grouper [12,13] is a snippet-based clustering engine. The main feature of Grouper is the introduction of a phrase-analysis algorithm called STC (Suffix Tree Clustering). The STC algorithm groups the input texts according to the identical phrases they share.

5.1. Accuracy

The result of the parameter accuracy for the algorithm and various repositories is shown in **Figure 1**. In ML-CLUBAS algorithm, a single cluster is mapped to more than one bug category which causes higher values of true negative and false positives in the confusion matrix. And because of higher values of true negative and false positive values there are drop in accuracy and F-Measure values in ML-CLUBAS algorithm. From the experimental results and graphs, it is clearly shown that algorithm CLUBAS performs better than any of the multi label text algorithms in terms of accuracy and F-Measure. The relationship between the number of classes and accuracy of

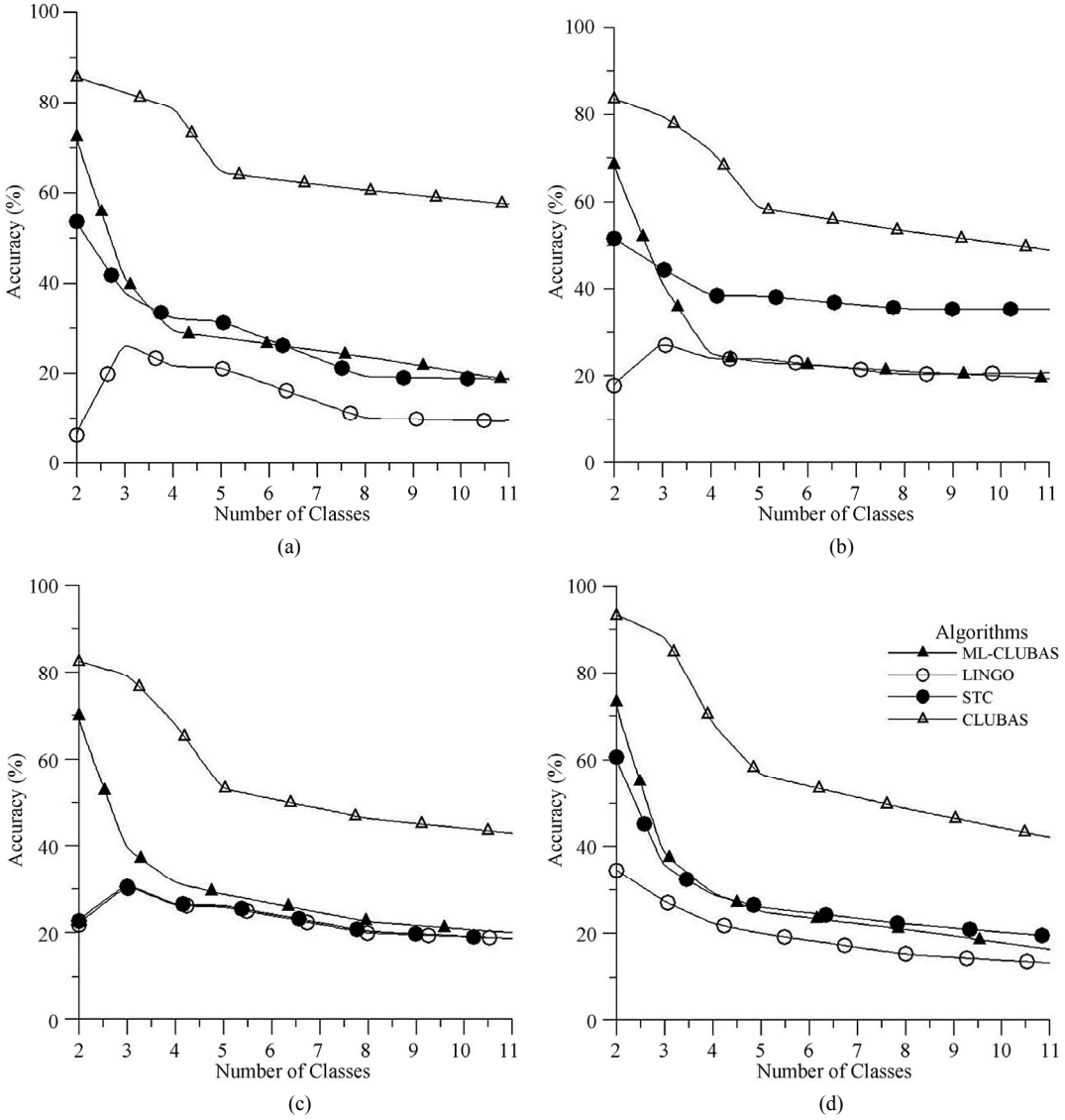

Figure 1. Accuracy in % for different number of classes in text clustering based classification techniques for (a) Android (b) JBoss-Seam (c) Mozilla (d) Mysql Bug Repository.

the multi label text clustering algorithm is shown in figure. With increase in number of classes, drop in accuracy values is observed. Accuracy wise ML-CLUBAS performs much better than both Lingo and STC algorithms for all the bug repositories taken in experiment except JBoss-Seam repository. In case of JBoss-Seam bug repository STC gives higher accuracy than ML-CLUBAS and Lingo, the reason behind this is analyzed from manual section of JBoss-Seam repository. From manual inspection it is observed that JBoss-Seam bug repository consist of less textual information (less amount of text in textual attributes of bug) than the other bug repositories.

5.2. F-Measure

The relationship between the F-Measure and number of classes is plotted in **Figure 2**. With increase in number of classes in bug classification, there is a minor drop observed in all the multi label text clustering algorithms. Except JBoss-Seam, for rest of the bug repositories there is a similar behavior of algorithms in terms of F-Measure. In case of JBoss-Seam bug repository, STC algorithm gives maximum values than other text clustering algorithms ML-CLUBAS and Lingo.

5.3. Number of Clusters

Number of clusters generated for different bug repositories is depicted in **Figure 3** for different number of bug samples taken for experiment. A sample size of 200, 300, 500, 800 and 1000 is taken for the experiment. From figure, it is shown that the number of clusters created

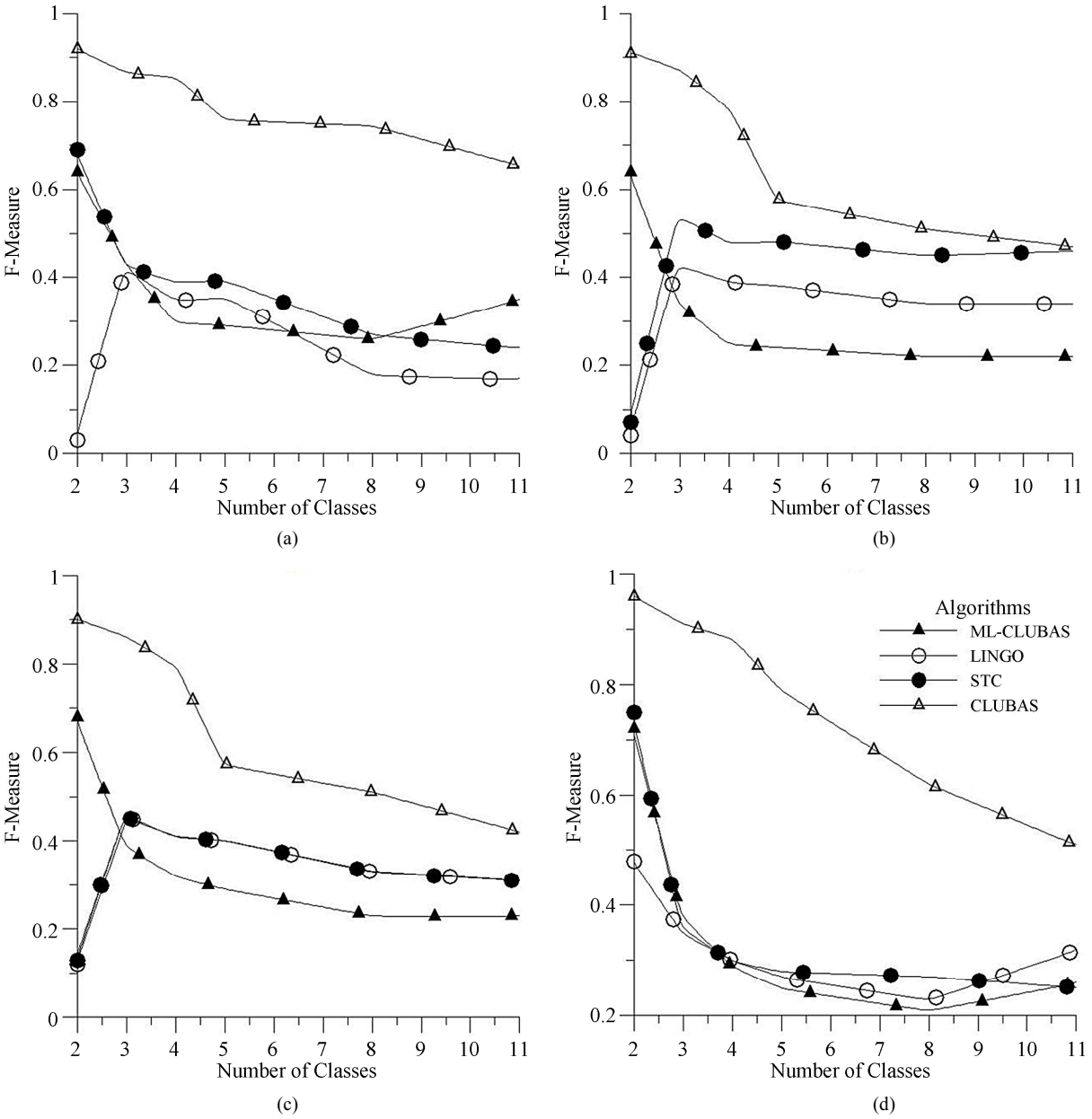

Figure 2. F-Measure in % for different number of classes in text clustering based classification techniques for (a) Android (b) JBoss-Seam (c) Mozilla (d) Mysql Bug Repository.

increases with the number of software bugs. This is because as the number of bugs increases the more number of bugs are discovered which are not falling in any of the existing clusters, in other words the dissimilar bugs are entering into the system, which causes forming of the newer bug clusters.

Both CLUBAS and ML-CLUBAS creates same number of bug clusters, since up to the bug cluster step, the mechanism of the algorithms is same. After creation of bug clusters only the implementation of CLUBAS and ML-CLUBAS differs. Lingo creates maximum number of clusters (with less number of bugs in the clusters) for all the repositories than ML-CLUBAS and STC. STC always creates less and fixed number of clusters, because of its tree data structure. STC algorithm generates less number of clusters, up to 2000 bug samples it generates 16 (2^4) clusters because it follows a tree based structure to generate the clusters. Lingo generates more clusters than STC, but less than the CLUBAS algorithm for the same number of bugs. Lingo creates clusters by identifying key phrases in text, whereas CLUBAS generates clusters using textual similarity information in the text collection of software bug attributes. The reason behind the less number of clusters in Lingo and STC is more

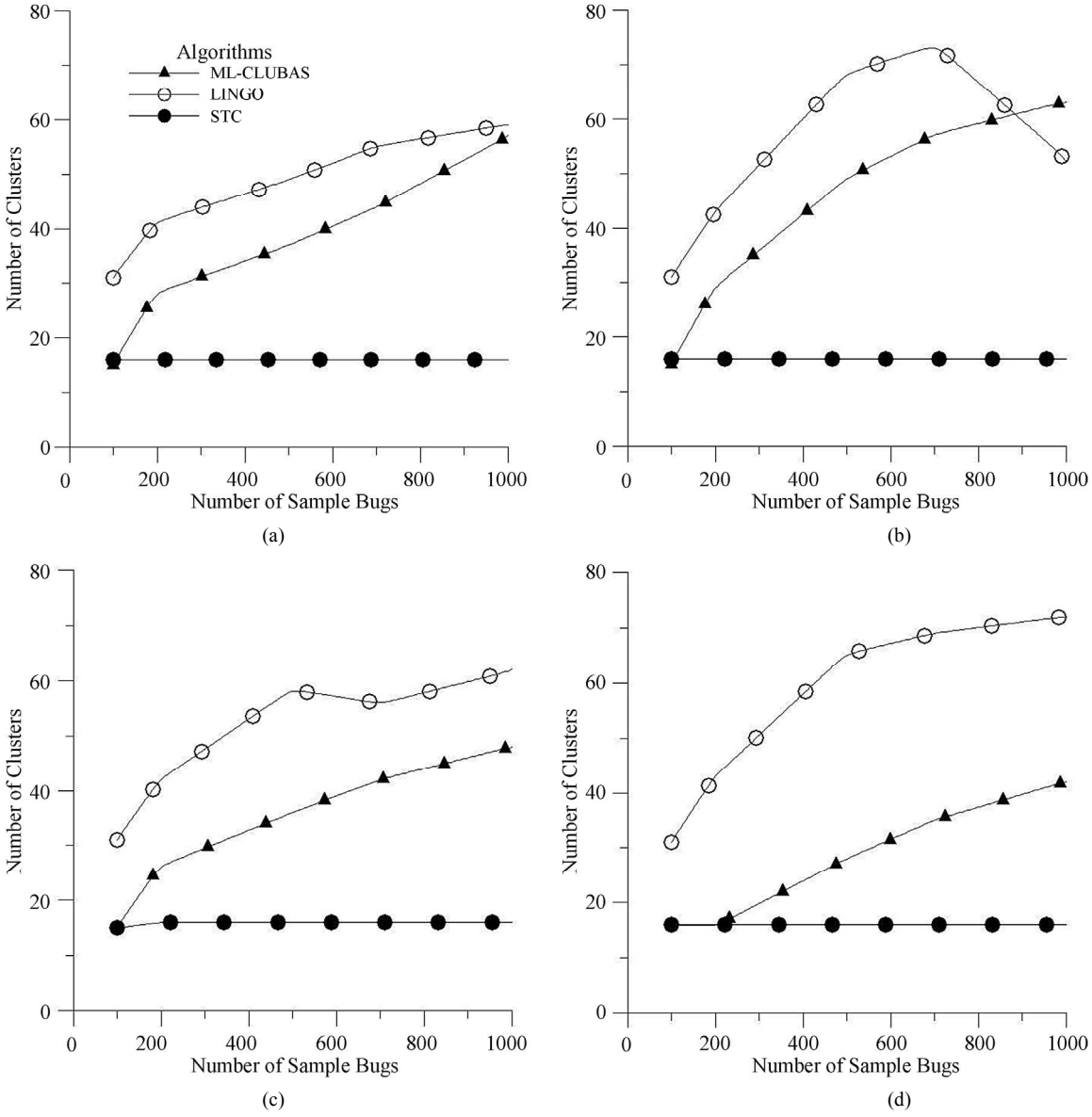

Figure 3. Number of clusters for multi label text clustering algorithms for different repositories.

number of software bugs are ignored in clusters and treated as outliers in Lingo and STC whereas less number of bugs is identified as outliers in CLUBAS. Around 7% bugs are identified as outliers in CLUBAS, whereas around 13% - 15% bugs are identified as outliers in Lingo and STC.

5.4. Entropy

The graph plotted for the corresponding entropy values in **Figure 4**. Figure indicates the entropy values calculated for different bug repositories at different sampling point. Lower value of entropy indicates the better quality of clusters. From figure it is observed that, for none of the repositories Lingo is able to produce the acceptable values (the ideal value for entropy is zero). For MySql bug repository the values are slightly better than other two repositories. Entropy wise ML-CLUBAS performs better than Lingo but not from STC algorithm, since Lingo creates more number of clusters than ML-CLUBAS algorithm. From the experimental results it is observed that almost in all the cases there is a log leaner relationship between the number of clusters and entropy values for multi label text clustering algorithms. Entropy wise, STC is the best algorithm than since it creates less number of clusters. However, ML-CLUBAS gives acceptable entropy values in the practical scenario, since it is difficult to get fewer clusters for text data using textual similarity mechanism for cluster creation.

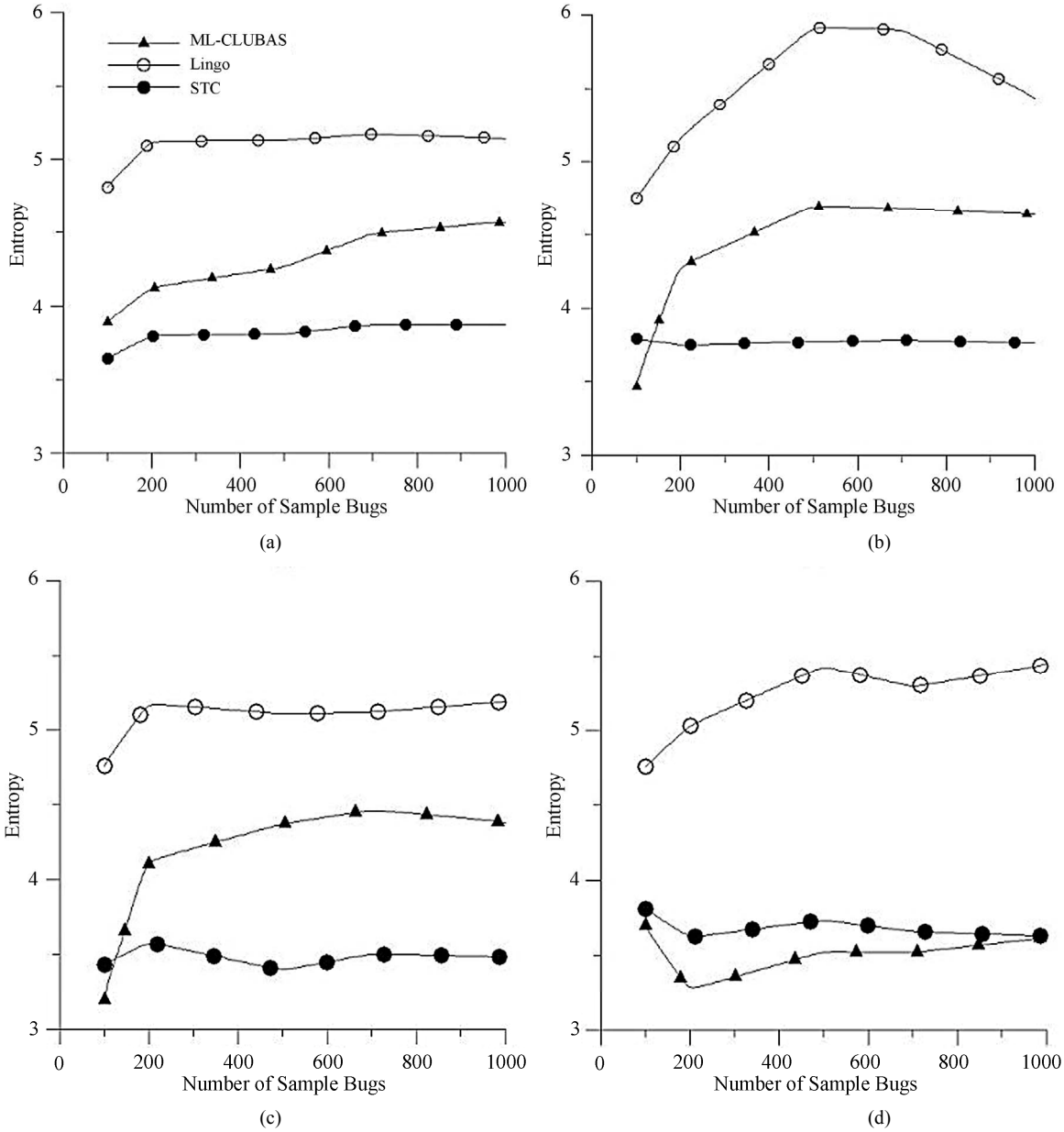

Figure 4. Entropy values for multi label text clustering algorithms for different repositories.

5.5. Computation Time

The computation time for creating the software bug classification using the text clustering algorithms (Lingo and STC) takes only five seconds up to 1000 software bugs records, whereas the algorithm ML-CLUBAS takes slightly higher computation time than Lingo and STC algorithms. The computation time calculated is around 20 seconds up to 1000 software bugs on the same machine using the same software bugs records. This is because measuring pair-wise attributes similarity and then applying clustering and label generation requires lot of calculations, which requires slightly more time than Lingo and STC algorithms. For 1000 bugs the maximum time taken is about 3.5 seconds and the maximum time taken for 100 bugs is about 1.2 seconds. The experiments are performed over a machine with CPU as 2.0 GHz and 2 GB of RAM (Random Access Memory).

6. Threats to Validity

The limitation of the work is same as with CLUBAS [1]. The experiments are performed a number of time to validate the results still there is a possibility that parameter values may slightly differ from the claimed results due to the randomness (random samples taken for the experiments). The other limitation of the work can be derived from the Zipf's power distribution law [14,15]. It

states that most of users use limited number of words (terms) frequently in the documents. Zipf's law supports the algorithm CLUBAS, and ML-CLUBAS algorithms since both are derived from the frequent terms concept, however in few cases where the developers or testers from different places can use the different set of vocabularies to specify the bug information, in that case, the accuracy values may drop slightly.

7. Conclusion & Future Scope

CLUBAS is a single label classification algorithm, where each bug cluster belongs to a single bug category. In this work, a multi label variant of CLUBAS, ML-CLUBAS is present with pseudo-code where a single bug cluster can be mapped to more than one bug category. The comparison of ML-CLUBAS with CLUBAS and other text clustering algorithm Lingo and STC is also presented. From results it is observed that since bug clusters are mapped to more than one category in ML-CLUBAS, which causes more values in TN (True Negative) and FP (False Positive) and hence less accuracy and F-measure than CLUBAS. From the comparison with Lingo and STC, it is found that accuracy wise algorithm ML-CLUBAS performs better. From cluster entropy wise STC is the best algorithm, however ML-CLUBAS gives the acceptable entropy values.

REFERENCES

[1] N. K. Nagwani and S. Verma, "CLUBAS: An Algorithm and Java Based Tool for Software Bug Classification Using Bug Attributes Similarities," *Journal of Software Engineering and Applications*, Vol. 5, No. 6, 2012, pp. 436-447. doi:10.4236/jsea.2012.56050

[2] S. Chapman, "Simmetrics, Java Based API for Text Similarity Measurement," 2011. http://www.dcs.shef.ac.uk/~sam/simmetrics.html.

[3] C. D. Manning, P. Raghavan and H. Schuitze, "Introduction to Information Retrieval," 2008. http://nlp.standford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html

[4] H. Li, K. Zhang and T. Jiang, "Minimum Entropy Clustering and Applications to Gene Expression Analysis," *Proceedings of IEEE Computational System Bioinformatics Conference*, Stanford, August 2004, pp. 142-151.

[5] I. H. Witten, E. Frank, L. E. Trigg, M. A. Hall, G. Holmes and S. J. Cunningham, "Weka (Waikato Environment for Knowledge Analysis)," 2011. www.cs.waikato.ac.nz/ml/weka

[6] "Android Bug Repository," 2011. http://code.google.com/p/android/issues

[7] JBoss-Seam, "Bug Repository," 2011. https://issues.jboss.org/browse/JBSEAM

[8] "Mozilla Bug Repository," 2011. https://bugzilla.mozilla.org

[9] MySql, "Bug Repository," 2011. http://bugs.mysql.com

[10] S. Osinski, J. Stefanowski and D. Weiss, "Lingo: Search Results Clustering Algorithm Based on Singular Value Decomposition," *Proceedings of the International Intelligent Information Processing and Web Mining Conference*, Zakopane, 17-20 May 2004, pp. 359-368.

[11] S. Osinski, "An Algorithm for Clustering of Web Search Results," Master's Thesis, Poznań University of Technology, Poznań, 2003.

[12] O. Zamir and O. Etzioni, "Grouper: A Dynamic Clustering Interface for Web Search Results," *Computer Networks*, Vol. 31, No. 11-16, 1999, pp. 1361-1374. doi:10.1016/S1389-1286(99)00054-7

[13] O. Zamir and O. Etzioni, "Web Document Clustering: A Feasibility Demonstration," *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval* (*SIGIR*), Melbourne, 24-28 August 1998, pp. 46-54.

[14] W. Li, "Random Texts Exhibit Zipf's-Law-Like Word Frequency Distribution," *IEEE Transactions on Information Theory*, Vol. 38, No. 6, 1992, pp. 1842-1845. doi:10.1109/18.165464

[15] W. J. Reed, "The Pareto, Zipf and Other Power Laws," *Economics Letters*, Vol. 74, No. 1, 2001, pp. 15-19. doi:10.1016/S0165-1765(01)00524-9

