Journal of Theoretical and Applied Information Technology
20th August 2015. Vol.78. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ONTOLOGY ASSISTED SEMI-SUPERVISED BUG REPORT CLASSIFICATION

[1]SAKTHI KUMARESH, [2]BASKARAN R

[1]Associate Professor, Dept. of Computer Science, MOP Vaishnav College for Women, Chennai, India

[2]Associate Professor Department of Computer Science and Engineering, Anna University, Chennai, India

E-mail: [1]sakthimegha@yahoo.co.in, [2].dr.baskaran10@gmail.com

ABSTRACT

Automated bug report clustering and classification plays a significant role in managing, assigning, and understanding the bug categories. The most challenging problem in bug report classification is the inadequate amount of labeled dataset. The proposed framework introduces an Ontology-assisted Semi-supervised Clustering Based Classification (OS-CBC) for bug reports amid a small size of the labeled dataset scenario. The proposed approach enriches the data set of the bug report using constructed Bug and Enriched Meta-feature Extraction (BEME) ontology. Semantic constraints based semi-supervised hierarchical clustering (Semantic-HAC) algorithm prioritizes the constraints for clustering the bug reports based on the BEME ontology. The cluster formation of bug reports depends on the transitive dissimilarity and ultrametric distance using ontology-based prioritized constraints. It extends the dataset (stretched) of the bug reports based on the maximum likelihood of the features in the cluster for labeling the unlabeled data. Moreover, the proposed approach categorizes the bug reports of stretched test set under the category of training set label using Multi-label Naive Bayes (MLNB) classifier. The classification technique focuses on the threshold based filtered weight of each term in the training set to improve the accuracy. The proposed OS-CBC approach significantly improves the classification accuracy of the bug reports.

Keywords: *Bug, Classification, Clustering, Ontology, Constraint, Semi-Supervised*

1. INTRODUCTION

A software bug is a common concern in software engineering and one needs to embark upon with it in a systematic way. In software systems, defect management is an important aspect to ensure the software system's reliability. Automated software defect management solves many issues related to software maintenance. Software repositories encompass the information of software bugs in the form of Extensible Markup Language (XML) or Hypertext Markup Language (HTML). A software bug or defect mainly consists of the title, description, and comments in textual format. The Bug Tracking System (BTS) manages the bug fixing process right from receiving the bug to the assignment. Software developers, testers, and end-users submit the bug reports to the bug repository. Bugzilla, Perforce, and JIRA are the open source bug trackers that allow the bug report from both developers and end-users. Bug triage is the process of assigning each bug report to the appropriate developer. An automatic bug tracking system tackles the issues of the labor-intensive, fault-prone, and time-consuming software bug process.

Bug report datasets available in the text summary format in which data analysis tool extracts the unknown structures and manages the vast datasets. Mining in bug reports improves the quality of bug report categorization. Mining based bug report analysis avoids the complexity of bug report identification. A bug report assignment to the software developers depends on the severity of bugs and experience of the developers. Mining techniques are used to identify the bug category of the bug reports. Bug report analysis through mining techniques assigns each class to an appropriate bug. Cluster analysis finds the group of identical objects that reveals homogeneous characteristics of bug reports in a group. A comprehensive survey discusses text clustering algorithms in the field of linked data and social networks [1]. The classification framework comprises of two phases namely, training, and the testing phase. Training data for the classifier creates the classification rules. The test set of classifier maps the test dataset of the bug report to appropriate classes depending on the training dataset. Real-world applications exploit the classification of tasks and require multi-class classifiers. A keyword extraction provides related

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

information to the word in the documents [2]. Ontology [3] provides the knowledge representation with the help of relationships and concepts of artifacts. It contains classes, properties, and the relationship of each term in a particular domain of knowledge. Ontology integrates the semantic knowledge to understand the related information and extracts the keyword [4]. It enhances the management, information organization, and understanding. It consists of classes and instances with the provisioning of machine learning semantics of the metadata. The significance of ontology is to improve the content intelligence further. The bug ontology framework integrates the knowledge of the bug report. The existing text classification [5] and the semi-supervised clustering [6] method exploits only submitted keywords in textual documents for performing classification and clustering. It degrades the performance of dissimilarity between clusters. To address this problem, the proposed approach employs ontology-based clustering and classification of bug reports.

In this paper, the proposed framework introduces Ontology-assisted Semi-supervised Clustering Based Classification (OS-CBC) approach to enhance the accuracy of bug classification. The proposed approach has four major phases such as Meta-feature generation, clustering, stretched training and test set generation, and classification. Preprocessing of the textual bug report is the initial stage to get the structured data. In the first phase, the proposed approach constructs BEME ontology for adding meta-features with each term of the dataset in the bug report. The second phase clusters the ontology-based labeled and unlabeled data using Semantic-HAC algorithm. It considers the prioritized triple-wise relative constraints using BEME ontology, dissimilarity matrix, and ultrametric distance for clustering the bug reports. In the third phase, the proposed approach extends the labeled and unlabeled dataset based on cluster attributes. The extension of the labeled and unlabeled dataset of the bug reports depends on the maximum likelihood of features in a cluster. In the training set, feature extension based on taxonomy is done for adding cluster features under the label of the training set. The extended form of the labeled and unlabeled data is known as stretched training and stretched test set. The final phase classifies the stretched test set under the stretched training set of bug reports using MLNB classifier. It classifies the test set threshold based on the filtered weight of the term in the training set and the number of times it occurs in the class. Finally, each test set of the bug related word has the labels. The experimental results reveal that the semantic constraint-based semi-supervised clustering based classification improves classification accuracy.

1.1 Contributions

- The proposed framework introduces ontology-assisted clustering based classification model for improving the accuracy in bug report classification.
- The proposed approach presents the BEME ontology for extracting the keyword of bug report dataset with additional features in the form of classes and instances.
- This proposed approach suggests semantic constraints based semi-supervised hierarchical technique for clustering the software bug reports. It considers the ontology-based priority of triple-wise relative constraints, and ultrametric distance matrix based elements on the constraints for clustering the ontology established labeled and unlabeled dataset.
- It also proposes the clustering based classification using MLNB classifier. The proposed approach extends the ontology established labeled and unlabeled dataset using cluster features. MLNB classifier categorizes the bug reports based on stretched test sets under stretched training set labels. The proposed categorization employs the weight of each term in the stretched training set and number of occurrences for identifying the probability of the term in the category.

2. RELATED WORK

Bugzie [7] assigns bug reports to the expertise of right developers depending on the Fuzzy set-based modeling. Automatic bug triaging of Bugzie approach fixes the bug with multiple technical terms of a bug report and the capability of developers. A hybrid approach classifies the bug reports using text mining and data mining techniques to predict the bug automatically. It consists of two stages, the textual bug report classification and the machine learning of future extraction. The data grafting merges the extracted features with the selected feature of the bug report and provides the output to machine learner [5]. The work [8] exploits Naive Bayesian classifier to categorize the textual bug reports using terms of the document. The feature extraction of classification depends on the presence or absence probability of a term in the document. The work [9] classifies the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

defects in software system automatically. It categorizes the text and code features of the defects based on the multi-class classification.

2.1 Classification and Clustering

Automated orthogonal defect classification (AutoODC) [10] enhances Relevance annotation framework. It automatically classifies the software system defects using textual features of defect reports. The semi-supervised text classification approach enriches the Naive Bayes (NB) classifier using expectation-maximization. Bug-triage employs semi-supervised classification method for avoiding the deficiency of the labeled bug report [11]. A string kernel [12] classifies the text documents based on the sub-sequence length of the feature. Kernel-based learning system text categorization exploits the Support vector machine. Clustering based classification (CBC) approach considers both labeled and unlabeled data of the dataset. Initially, CBC clusters the labeled data. It labels the unlabeled data depends on clusters. The trained set of expandable labeled data is the input for classifier for improving the accuracy of classification [13].

Hierarchical Agglomerative Clustering [14] enhances the constraints based clustering. It employs clustering constraints ordering format to improve the similarity of constraints in a hierarchical manner. Predicting priority via a multi-faceted factor analysis automatically prioritizes the bug reports that depend on machine learning. The priority level focuses multiple factors such as temporal, textual, related-report, author, product, and the severity of the bug reports. The extracted features of various factors are the input for ThresholdinG and Linear Regression to ClAssifY Imbalanced Data (GRAY) classification engine [15]. Graph partitioning algorithm [16] clusters the textual information of bug reports. It exploits three clustering algorithms such as normalized cut (summary), size regularized cut (description), and k-means algorithm (summary and description).

Ontology-based text clustering improves the clustering performance. The clustering method depends on the distance measure of each word using the ontology. Distance measure calculation depends on the Euclidean distance and two k-means clustering algorithms [17]. Semi-supervised hierarchical clustering [6] considers the triple-wise relative constraints and ultrametric dendrogram distance. It exploits constrained optimization technique and transitive dissimilarity based technique for achieving an effective accuracy of semi-supervised clustering. A comparative analysis in [18] evaluates the clustering algorithms of agglomerative clustering and partition approaches. A constrained agglomerative hierarchical clustering model solves the problem of the initial merging error in agglomerative clustering method.

The existing techniques for text classification are supervised learning models. The proposed approach exploits a Naive Bayesian learning method for classifying the textual bug reports due to the natural adaptation of multi-label classification. The constrained agglomerative hierarchical clustering approach improves the clustering performance and the high complexity of the data.

3. PROPOSED METHODOLOGY

The proposed framework is the ontology assisted clustering and classification algorithm. Fig.1 illustrates the bug report processing stages such as a pre-processing, meta-feature generation using bug ontology, BEME ontology-based prioritization for clustering, extended training, and test set creation and filtered weight based classification of software bugs.

The bug tracking system manages the bug reports and enables end-users or software developers submit the bug reports. Bug reports contain more information, for instance, one-line bug summary, data and time of a bug, bug location, priority and severity of software defects. The proposed framework constructs the bug ontology based on the significance of the bug reports.

The proposed approach enriches the UltraTran [6] approach to clustering the bug reports. UltraTran degrades the performance due to the weak similarity of clustering constraints. The existing semi-supervised clustering algorithm fails to consider the vital constraint identification for clustering. The enriched structure of the UltraTran clustering method is a Semantic-HAC algorithm using ontology-based constraint prioritization. The proposed approach extends the text classification approach [5] using clustering based extended training and testing set. The proposed classification approach uses an MLNB classifier for multi-label classification.

3.1 Preprocessing

Preprocessing of the textual bug report is the initial stage in clustering and classification. It provides the relative technical information about a textual bug report from bug repositories. Tokenization, stop-word removal, and stemming are the pre-processing steps. Tokenization splits the textual bug document into constituent words while

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

removing numbers, punctuation marks, and weak correlation terms of bug reports. Hence, it extracts the keywords from bug documents and performs the stop-word removal process after the completion of tokenization of the document for providing technical terms.

Stemming is the process of modifying the related words to root words. For instance, "modify" is the fundamental form of similar words such as, "modification", "modified", "modifies".

3.2 Bug Ontology

Ontology provides related information to understand the technical strength of the word. OntoDM [19] discusses the ontology for the data mining domain. Ontology construction mainly depends on the factors of concepts, properties, instances, axioms, and relationships for a particular domain. Domain experts construct the ontology based on the ontology construction tools such as Protege and OntoEdit. Ontology development has the two necessary stages. In the first, it receives the information about domain knowledge in terms of properties, concepts, and relationships. In the second, it constructs the hierarchy structure in the form of classes, subclasses, and instances. The proposed bug ontology construction exploits the knowledge base driven ontology construction [20].

The proposed approach constructs the bug ontology to ensure the relative extraction of the given bug report. Bug and Enriched Meta-feature Extraction (BEME) ontology consists of the classes and instances of bug information. The proposed approach enriches the dataset with the support of BEME ontology for adding additional features of the dataset based on the semantic information. The dataset includes labeled and unlabeled dataset of the bug reports. BEME ontology automatically extracts the relative information of the bug report. It integrates the keyword of the bug report with new features to enhance the real strength of the term.

The construction of BEME ontology depends on the concepts, instances, properties, and axioms of software bug reports. Bug ontology representation languages are Web Ontology Language (OWL), Resource description framework (RDF), and simple ontology HTML extension (SHOE). BEME ontology forms the hierarchy with the related terms of bug reports information in the bug repositories. The constructed bug ontology establishes the relevant keywords in various forms such as class, subclass, and equivalent class. BEME ontology extracts the dataset keywords based on semantic relativity. Bug ontology retrieves the information related to the keywords for enriching the bug report dataset.

3.3 Clustering

Clustering stage forms the group of preprocessed bug attributes based on the similarity measurement of each term in a text document. Cluster formation fully depends on the weighted similarity between each software bug attribute. Dataset of the bug report contains inadequately labeled data and unlabeled data of attributes. A clustering technique receives input from both labeled and unlabeled data for bug reports. The proposed approach employs the agglomerative model of the hierarchical clustering [14] for clustering the constraints. It clusters the features of the bug report, depending on the similarity constraints and the distance between the terms. The proposed ontology-based prioritization of constraints addresses the problem of dissimilarity among clustering constraints.

3.3.1 Constraint prioritization

The proposed approach addresses the problem of constraint importance for identification using ontology-based constraint prioritization. Semantic-HAC algorithm prioritizes the semantic constraints of the bug report based on the constructed BEME ontology. Root word includes constraints at various distances in the hierarchical form. The proposed approach considers triple-wise constraints and each constraint has the three elements of the bug ontology. A constraint has the highest priority for clustering if two elements of a constraint are a direct child (equal distance) from the root element. The constraint priority depends on the similar features between the bug reports. It provides the high bonding information of constraints to form clusters. Initially, a constructed BEME ontology consists of related information of each keyword in the bug report. Hence, the proposed approach exploits the BEME ontology for prioritizing the constraints.

3.3.2 Cluster Formation

The proposed algorithm depends on semi-supervised clustering to form the cluster using an ultrametric distance. It forms the cluster based on the ultrametric distance between the BEME ontology-based prioritized constraints. C_1 and C_2 are the nearest two clusters to be merged using an agglomerative hierarchical clustering method. Ultra-metric distance consideration, groups the two nearest clusters to form a single cluster using the agglomerative hierarchical clustering method. The

Journal of Theoretical and Applied Information Technology
20th August 2015. Vol.78. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

selected two nearest clusters satisfy the following condition,

$$d(C_1,C_2) \leq \min (d(C_1,C_2), d(C_2,C_3)) \quad (1)$$

$$\min(d(C_1,C_2),d(C_2,C_3)) \leq d(C_1 \cup C_2,C_3) \ \forall a=1,2,3 \quad (2)$$

The combining order of the constraints to form a cluster depends on the ultrametric distance and transitive dissimilarity of each keyword in the bug report. Constraint set has three constraints such as a_i, a_j, and a_l to create each cluster. It assumes that the constraint set is in the form of (i,j,k) and (i,j,l). The order of prioritized constraints entirely focuses on the similarity strength of the bugs due to the meta-features in BEME ontology. It decides to merge the order of prioritized constraints using the following condition,

$$d(a_i,a_j) \leq \max(d(a_i,a_j),d(a_j,a_l)) \ \forall a_i,a_j,a_l \quad (3)$$

The semi-supervised hierarchical method forms the dissimilarity matrix (DB) for bug reports. Hierarchical clustering converts DB into the ultrametric distance (D'B) formation for clustering the prioritized constraints. The optimal ultrametric distance of the bugs for clustering is as follows.

$$\arg\min_{D'B} \sum_{a_i a_j \in A} \left(DB_{ij} - D'B_{ij}\right)^2 \quad (4)$$

Equation (4) produces the exact constraints of (i,j) that has the minimum distance (similarity) to form a cluster. It provides the optimal solution based on the ultrametric distance matrix. This proposed approach does not require the minimum distance value to create the cluster. Hence, it exploits argmin function to find the clustering terms with minimum distance from the ultrametric distance matrix.

The transitive dissimilarity matrix includes the row and column of each term of the bugs and value represent the weight of the terms or words. For instance, $d(a_i,a_l)$ is the dissimilarity value between a_i and a_l from DB matrix. The proposed approach combines the constraints to form a cluster using prioritized constraints of dissimilarity matrix and ultrametric distance. Hence, it constructs a set of clusters, and each cluster has ID with similar attributes of the bug words.

3.4 Extension

Extended labeled, and unlabeled dataset also has the additional features of the initial labeled and unlabeled dataset. The outcome of stretched training data ((f1,l1)*) depends on the maximum likelihood of similarity between cluster attributes and training data under the class label. Similarly, the stretched test set (F1*) includes the additional features of each cluster. The proposed approach creates taxonomy based on the bug reports in the repository

The proposed approach exploits the weighting scheme of the TFIDF model as a classic TFIDF model. It considers the Boolean form of the presence and absence of the word in a text document for term frequency. Each bug report is in the form of a vector of words representation. Each word in the bug report collects the relevant information of the word in the cluster and training sets. Weighted similarity measurement of each bug report exploits the TFIDF model based on the term frequency (TF) and inverse document frequency (IDF) of each word. Weight of each additional feature (w_{i+a}) in the bug report is calculated as,

$$W(w_{i+a}) = TF(w_{i+a},k) * IDF(w_{i+a}) \quad (5)$$

In the training set, cluster feature based TFIDF model does not include the meta-feature accurately due to the categorization problem of similar features in the cluster. Hence, it employs taxonomy based categorization. The proposed approach considers each document of the cluster includes the additional feature (w_{i+a}). Inverse document frequency IDF(w_{i+a}) is given as,

$$IDF(w_{i+a}) = \log_2\left(\frac{|K|}{|C_a|}\right) \quad (6)$$

Where,

k – Number of documents in the cluster
|K| - total number of clusters
$|C_a|$ - number of occurrence cluster

The proposed approach uses the TFIDF model for measuring the weighted similarity of bug keywords with the cluster. Cluster-based TFIDF model adds the extra features to the test set of the bug reports. The meta-feature of the term is added to the test set if the term of the cluster has higher weight than other clusters.

3.5 Classification

Clustering based classification approach exploits a stretched training and test set of bug reports based on cluster features. Stretched training and the test set contain the initial bug report features with additional relative features. An ensemble classification using multi-label Naive

Journal of Theoretical and Applied Information Technology
20th August 2015. Vol.78. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Bayes classifier (MLNB) discussed in [21]. The stretched training set is the input to the MLNB classifier for categorizing the stretched test data sets. Multi-label classification allows more than a few classes at the same time for classifying each term. The proposed classifier categorizes each bug report in stretched test set based on the posterior probability. The primary method of multi-label Naive Bayes classifies the word in the test set classes based on the term frequency. It evaluates the presence or absence of the word in the training set. In some case, it degrades the classification accuracy of the term due to the mere consideration of the number of occurrences.

The proposed approach categorizes the term based on labeled data weight. In the training set, each word has the weight for classifying the word of the test set using TFIDF model. The stretched training set contains the filtered weight features based on the threshold value. Threshold value based weighted term in each class increases the similarity of term strength. The probability of each word depends on the number of occurrences and weight (W_t) of each term in the training set. It improves the term strength in the class based on weight.

The data set contains a collection of bug reports $B=\{b_1,b_2,...b_N\}$. Each bug report consists class labels $C=\{c_1,c_2,...c_C\}$ and bag of words (terms) in the training set, $w=\{w_1,w_2,...w_T\}$. The Naive Bayes method employs the joint probability of classes and words for evaluating the class probability of the bug report. MLNB classifier trained with the stretched training set examples to assign a bug report to classes based on the probability. In this proposed approach, each bug report is considered as a document. In each bug report, word (w_t) occurs n_{wt}^{c}-times in category 'c' and $n_{wt}^{c'}$-times of other than the category 'c'. The probability of the word in the specific class label ($P(w_t \mid c)$) depends on the number of occurrences of each term (n_{wt}) and weight of each term (W_t). The probability of the word for class c and other than c-class labels is as follows.

$$P(w_t|c) = (n_{wt} * W_t)^c \Big/ \sum_{t=1}^{T}(n_{wt} * W_t)^c \quad (7)$$

$$P(w_t|c') = (n_{wt} * W_t)^{c'} \Big/ \sum_{t=1}^{T}(n_{wt} * W_t)^{c'} \quad (8)$$

The proposed approach selects the number of classes based on the filtered weight of each term. The weight of each term depending on the number of times it occurs in bug reports. If the training set contains N- the total number of bug reports and C - classes in category c, the probability of the category c and other than c is given as,

$$P(C) = C / N \quad (9)$$

$$P(c') = N - C / N \quad (10)$$

The proposed classification depends on the probability of the word in each document and the probability of the category using the total number of bug reports. In the testing set, the bug report B_i contains M number of words $W_{Bi} = \{w_i^1,w_i^2,...w_i^M\}$. According to a multi-label Naive Bayes theorem, the probability of the category c and c',

$$P(c|B_i) = \prod_{m=1}^{M} P(w_i^m|c) * P(c) \quad (11)$$

$$P(c'|B_i) = \prod_{m=1}^{M} p(w_i^m|c') * P(c') \quad (12)$$

MLNB classifier assigns the term to the corresponding category if the probability of the bug is higher than the threshold probability value among multiple classes of the stretched training set. The probability of $P(c \mid B_i) + P(c' \mid B_i) = 1$.

The proposed approach performs preprocessing the dataset of the bug reports from the bug repository. Fig.2 shows the proposed algorithm. Labeled and unlabeled dataset feature extraction is depending on the BEME ontology related terms (t_i^o) and add relative extraction of ontology terms to the dataset features. In clustering phase, each constraint has the set of three elements based on the BEME ontology. The proposed approach prioritizes the constraints for clustering using the related information. It forms the Dissimilarity Bug matrix (DB) based on the weight of each term. The proposed approach transforms the DB matrix into the ultrametric distance. Hence, the proposed method forms the cluster (C_a) using prioritized constraints. It extends the training and test set based on the cluster using the maximum likelihood of features and taxonomy category. Stretched training set ((f1,l1)*) and the stretched test set (F1*) is input for MLNB classifier. It categorizes the bug reports into stretched test set with labels ((F_i,l_i)*) based on the posterior probability of the number of occurrences and weight of the term in a document.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Input : Dataset of Bug reports

Output: Test set with predicted labels

Labeled dataset = {(f1,l1), (f2,l2),...(fn,ln)}

Unlabeled dataset ={F1,F2,....Fn}

StretchedTrainingset= {(f1,l1)*, (f2,l2)*,...(fn,ln)*}

Stretched Test set ={F1*,F2*,....Fn*}

//Meta-features generation of dataset using BEME ontology

for Parsed bug reports -> BEME ontology

Mapping ((f1,l1) and F_i) -> t_i^O

if(match((f1,l1) and F_i)=t_i^O))

Add meta-features to (f1,l1) and F_i)

endif

endfor

//Semantic-HAC algorithm

//Ontology-based constraint prioritization for clustering

for(Cons1,....consM)

ConsM={x_a^m,x_b^m,x_c^m}

if(Dist($x_a^m x_b^m$=$x_a^m x_c^m$))

Add elements into priority set (P(1), consM)

elseif (Dist($x_a^m x_b^m$<$x_a^m x_c^m$))

Add elements into priority set (P(0), consM)

endif

endfor

Create matrix for bug report attributes of DB

Transitive dissimilarity strength of S

Init: DB = S

Select similarity based constraints

for k <- 0 to n **do**

for i <- 0 to n **do**

for j <- 0 to n **do**

for all cons = {a_i,a_j,a_k}**do**

min cons = min(min cons,d(a_i,a_k))

endfor

m_{ij} = min{m_{ij} , max(m_{il},m_{jl}),min cons}

endfor

endfor

endfor

Return S

Create cluster based on constraints order

C_a = {(f_i+F_i,l_i)}

//Generation of stretched training and testing set

Taxonomy based stretched features formation

Create category depends on bug report repository

Consider maximum likelihood features of C_a for training set

(f_i,l_i)* = ((f_i,l_i),C_a)

F_i* = (F_i,C_a)

//Multi-label Naive Bayes classifier

Training set contains threshold based filtered weight for each term

Calculate W_t for each word in training set

(f_i,l_i)* -> MLNB

Categorize F_i* under labels of (f_i,l_i)* using

Figure 2: Ontology-assisted Semi-supervised Bug report classification algorithm

4. Experimental Evaluation

This experiment exploits Eclipse and Mozilla dataset to evaluate Ontology-assisted Semi-supervised Clustering Based Classification (OS-CBC) of bug reports. The proposed approach compared with baseline algorithms of constraint-based semi-supervised hierarchical clustering (UltraTran) [6], CBC [13] and bug report classification (Data grafting) [5].

4.1 Experimental Setup

The proposed approach evaluates 1000 unique samples of bug reports from each dataset of Eclipse and Mozilla. It employs Bugzilla bug tracker for both Eclipse and Mozilla repository. The proposed framework forms 1000 Bugs into 600 labeled data and 400 unlabeled data. It creates the set of bug reports into 3 clusters. Classify samples of 1000

Journal of Theoretical and Applied Information Technology
20th August 2015. Vol.78. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

bug reports into three classes. Each class includes 200 bug reports of the labeled data.

The experimental evaluation implements the proposed framework using Java platform. The proposed framework exploits two datasets such as Eclipse and Mozilla dataset for classifying the bug reports. Stanford parser performs the preprocessing for both Eclipse and Mozilla bug reports. The parsed data in the bug reports are inputs for constructing bug ontology (BEME). The proposed implementation relies on the Jena API for bug reports and BEME ontology. Each dataset contains the labeled and unlabeled data (.txt files) of the bug reports. The proposed approach clusters the labeled and unlabeled bug reports using BEME ontology. Semantic-HAC algorithm prioritizes the constraints based on the BEME ontology meta-features.

4.2 Evaluation Metrics

- **True Positive (TP):** It is the ratio between the number of predicted bug reports to be similar and actual similar bug reports.
- **True Negative (TN):** It is the ratio between the number of predicted bug reports to be dissimilar and actual dissimilar bug reports.
- **False Positive (FP):** It is the ratio between the number of predicted bug reports to be similar and actual dissimilar bug reports.
- **False Negative (FN):** It is the ratio between the number of predicted bug reports to be dissimilar and actual similar bug reports.

4.3 Evaluation Results

The proposed framework evaluates the results in terms of precision, recall, and F-measure.

4.3.1 Precision

Precision = TP / (TP + FP)

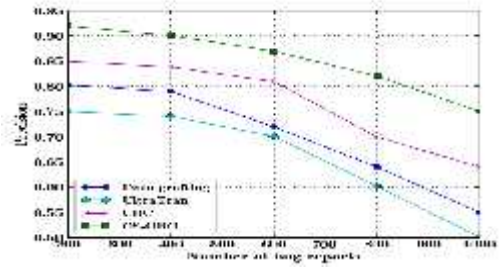

Figure 3: Precision of OS-CBC

In this experiment, the proposed approach exploits 1000 samples for classifying the bug reports. Fig.3 shows the impact of OS-CBC precision value for Eclipse dataset. The proposed precision value of 92 % has the better result than UltraTran due to the constraint prioritization based on the BEME ontology. Hence, the proposed approach improves the performance of precision than existing algorithms due to Semantic-HAC algorithm and MLNB classifier.

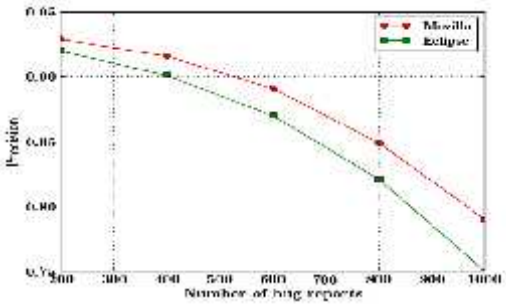

Figure 4: Comparison of Precision in Mozilla and Eclipse

The proposed experiment evaluates the datasets of Mozilla and Eclipse. The performance evaluation in Fig.4 describes the precision value while increasing the number of bug reports in both Mozilla and Eclipse dataset. BEME ontology improves the performance of relevant features of the bug reports. Semantic-HAC algorithm significantly increases the similarity strength due to ontology-based clustering and extension of labeled and unlabeled datasets. Hence, it improves the classification accuracy of the bug reports.

4.3.2 Recall

Recall = TP / (TP + FN)

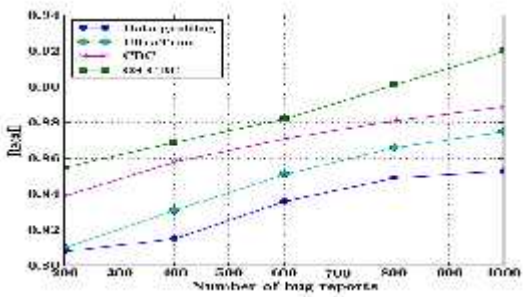

Figure 5: Recall of OS-CBC

Fig.5 illustrates the Recall of OS-CBC value while increasing the number of bug reports from Eclipse dataset. The proposed approach improves the recall value than data grafting techniques due to clustering based classification. It improves the classification accuracy based on the cluster attributes and extended dataset. The recall value decreases while increasing the number of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

documents. In this case, the recall value of the proposed approach is better value than CBC due to ontology-assisted clustering based classification. The OS-CBC framework effectively classifies the bug reports using MLNB classifier.

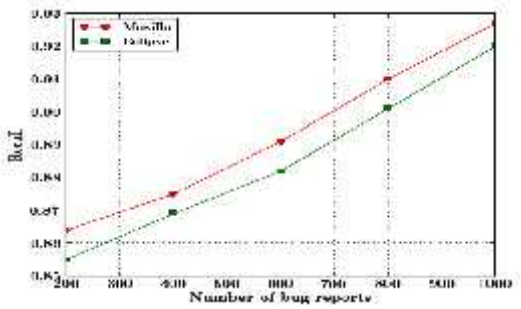

Figure 6: Comparison of Recall in Mozilla and Eclipse

Fig.6 shows the recall of two data sets (Mozilla and Eclipse) of 1000 bug reports classification. The proposed experiment achieves the high performance due to semantic constraint-based semi-supervised clustering. MLNB classifier categorizes the bug reports based on the features in each class with high similarity due to ontology-based extraction and extended features of the bug reports.

4.3.3 F-measure

F-measure = 2 * ((Precision * Recall) / (Precision +Recall))

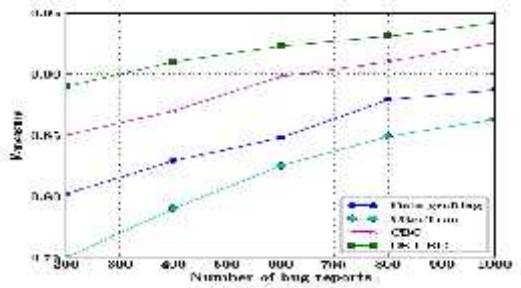

Figure 7: F-measure of OS-CBC

F-measure or F-score is a measure to test the accuracy of the bug report classification. It depends on the precision and recall value of the bug report clustering based classification. The proposed approach has the best F-measure value due to the better precision and recall value. It is the harmonic mean of the precision and recall value in each label due to multi-label classification. Fig.7 shows the F-measure value than existing approaches due to ontology-assisted bug report classification. The proposed OS-CBC achieves 89% of the F-measure value while evaluating 1000 samples of bug reports. Fig.7 plots the experimental results for Eclipse dataset.

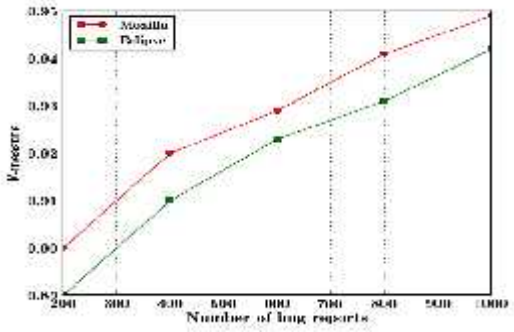

Figure 8: Comparison of F-measure in Mozilla and Eclipse

Fig 8 illustrates the F-measure performance while using 1000 bug reports from Mozilla and Eclipse dataset.

5. Conclusion

This paper presents Ontology-assisted Semi-supervised Clustering Based Classification (OS-CBC). The proposed approach clusters the bug reports using BEME ontology-based labeled and unlabeled dataset and Semantic-HAC algorithm. Taxonomy of the category and maximum likelihood of the features extend the labeled and unlabeled data based on the cluster features. The extension phase produces the stretched training and test set for classifying the bug reports. Furthermore, the proposed approach categorizes the stretched test set of the bug reports using MLNB classifier and considers the threshold weight based filtered term in the stretched training set. Experimental evaluation reveals the classification accuracy of the labeled and unlabeled bug reports.

REFRENCES:

[1] Aggarwal, Charu C., and ChengXiang Zhai, "A survey of text clustering algorithms", Springer transaction on Mining Text Data, 2012, pp.77-128,

[2] K. Coursey, R. Mihalcea, and W. Moen, "Automatic Keyword Extraction for Learning Objects Repositories" In Proceeding Conference of the American Society for Information Science and Technology, Vol.45, No.1, 2008, pp.1-10

[3] Chandrasekaran, Balakrishnan, John R. Josephson, and V. Richard Benjamins, "What are ontologies, and why do we need them?",

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IEEE Intelligent systems, 1999 Vol.14, No.1, pp.20-26

[4] Gašević, Dragan, Nima Kaviani, and Milan Milanović, "Ontologies and software engineering", Springer Berlin Heidelberg, In Handbook on Ontologies,2009, pp.593-615

[5] Zhou, Yu, Yanxiang Tong, Ruihang Gu, and Harald Gall, "Combining Text Mining and Data Mining for Bug Report Classification", IEEE International Conference on Software Maintenance and Evolution (ICSME), 2014, pp.311-320,

[6] Zheng, Li, and Tao Li, "Semi-supervised hierarchical clustering", IEEE 11th International Conference on Data Mining (ICDM),2011, pp.982-991

[7] Tamrawi, Ahmed, Tung Thanh Nguyen, Jafar Al-Kofahi, and Tien N. Nguyen, "Fuzzy set-based automatic bug triaging: NIER track", 33rd IEEE International Conference on Software Engineering (ICSE), 2011, pp.884-887

[8] Lamkanfi, Ahmed, Serge Demeyer, Emanuel Giger, and Bart Goethals, "Predicting the severity of a reported bug", 7th IEEE Working Conference on Mining Software Repositories (MSR),2010, pp.1-10

[9] Thung, Ferdian, David Lo, and Lingxiao Jiang, "Automatic defect categorization", IEEE 19th Working Conference on Reverse Engineering (WCRE),2012, pp.205-214

[10] Huang, LiGuo, Vincent Ng, Isaac Persing, Ruili Geng, Xu Bai, and Jeff Tian, "AutoODC: Automated generation of orthogonal defect classifications", 26th IEEE/ACM International Conference on Automated Software Engineering (ASE),2011, pp.412-415

[11] Xuan, Jifeng, He Jiang, Zhilei Ren, Jun Yan, and Zhongxuan Luo, "Automatic Bug Triage using Semi-Supervised Text Classification", In SEKE,2010, pp.209-214

[12] Lodhi, Huma, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins, "Text classification using string kernels", The Journal of Machine Learning Research, 2002, Vol.2, pp.419-444

[13] Zeng, Hua-Jun, Xuan-Hui Wang, Zheng Chen, Hongjun Lu, and Wei-Ying Ma, "CBC: Clustering based text classification requiring minimal labeled data", Third IEEE International Conference on Data Mining ICDM, 2003, pp.443-450

[14] Zhao, Haifeng, and Zijie Qi, "Hierarchical agglomerative clustering with ordering constraints", IEEE Third International Conference on Knowledge Discovery and Data Mining,2010, pp.195-199

[15] Tian, Yuan, David Lo, and Chengnian Sun, "Drone: Predicting priority of reported bugs by multi-factor analysis", 29th IEEE International Conference on Software Maintenance (ICSM), 2013, pp.200-209

[16] Rus, Vasile, Xiaofei Nan, Sajjan G. Shiva, and Yixin Chen, "Clustering of Defect Reports Using Graph Partitioning Algorithms", In SEKE,2009, pp.442-445.

[17] Jing, Liping, Lixin Zhou, Michael K. Ng, and J. Zhexue Huang, "Ontology-based distance measure for text clustering", In Proceedings of the Text Mining Workshop, SIAM International Conference on Data Mining, 2006, Vol.23

[18] Zhao, Ying, George Karypis, and Usama Fayyad, "Hierarchical clustering algorithms for document datasets", Data mining and knowledge discovery,2006, Vol.10, No.2, pp.141-168.

[19] Panov, Pance, Saso Dzeroski, and Larisa N. Soldatova, "OntoDM: An ontology of data mining", IEEE International Conference on Data Mining Workshops ICDMW'08, 2008, pp.752-760.

[20] Aaberge, Terje, and Rajendra Akerkar, "Ontology and Ontology Construction: Background and Practices", International Journal of Computer Science and Applications (IJCSA), 2012, Vol.9, No.2, pp.32-41.

[21] Alessandro, Antonucci, Giorgio Corani, Denis Mauá, and Sandra Gabaglio, "An ensemble of Bayesian networks for multilabel classification", ACM Proceedings of the Twenty-Third international joint conference on Artificial Intelligence, pp.1220-1225, AAAI Press

Journal of Theoretical and Applied Information Technology
20th August 2015. Vol.78. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

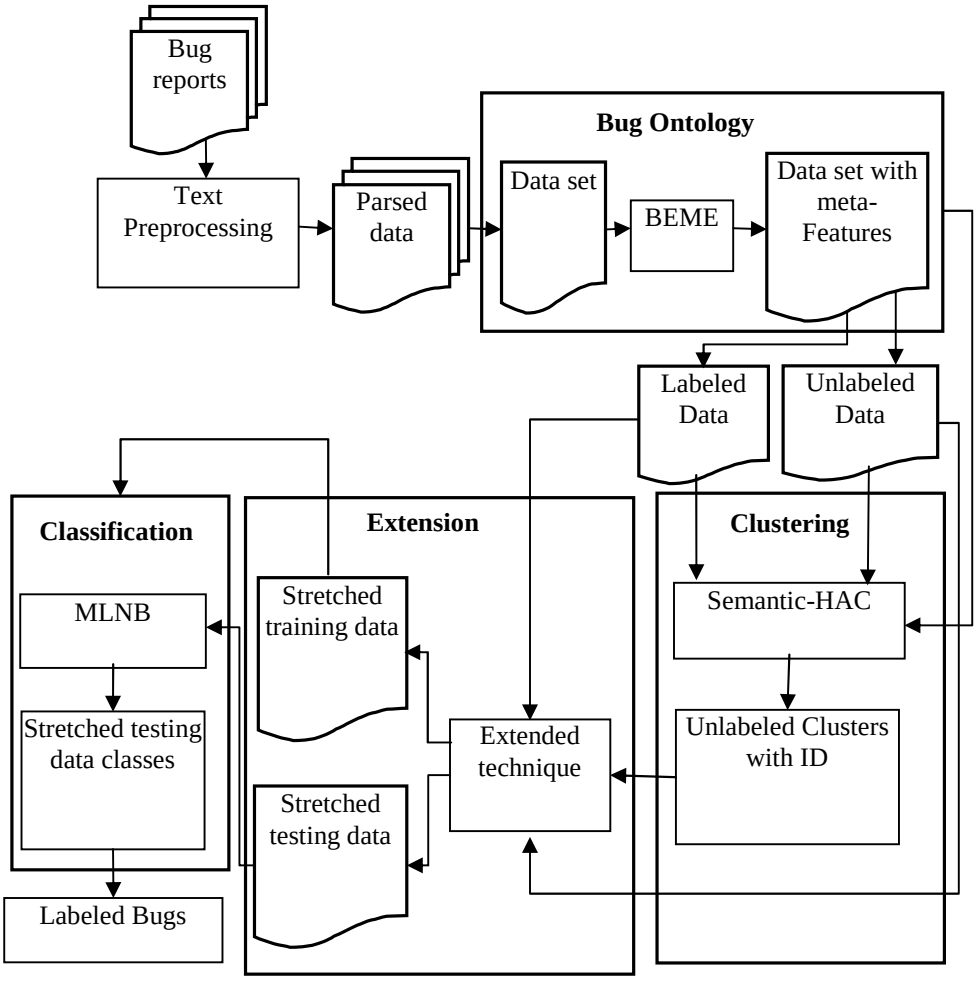

Figure 1: Ontology-Assisted Semi-Supervised Bug Report Classification