An Automated approach for Bug Categorization using Fuzzy Logic

Indu Chawla
Jaypee Institute of Information Technology
Noida-201307
India
Indu.chawla@jiit.ac.in

Sandeep K. Singh
Jaypee Institute of Information Technology
Noida-201307
India
Sandeepk.singh@jiit.ac.in

ABSTRACT
Various automated techniques built to benefit software developers, bug triagers, stakeholders and users in open source systems, utilize information placed in issue tracking systems. The success of these techniques depends largely on the quality of information present in the issue reports. Assigning correct label to issue reports is one of the quality concerns. Previous empirical studies conducted on the issue reports show that most issues are either mislabeled or are not labeled at all. Thus, in order to enhance quality of issue reports, there is a strong need to propose an automated and accurate bug labeling approach. A label can be a bug, feature enhancement or other request.

In this paper, we propose an automated approach to label an issue either as bug or other request based on fuzzy set theory. Experiments are conducted on issue repository of three open source software systems: HTTPClient, Jackrabbit and Lucene. We have achieved an accuracy of 87%, 83.5% and 90.8% and F-Measure score of 0.83, 0.79 and 0.84 respectively. This is a considerable improvement as compared to the earlier reported work on these three datasets using topic modeling approach.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: Management

General Terms
Algorithms, Performance, Experimentation, Verification

Keywords
Bug Repository, Bug Categorization, Fuzzy Logic

1. INTRODUCTION
A variety of tasks like automatic developer assignment, priority and severity assessment, duplicate bug report identification etc [18, 21] depend upon the information present in the fields of bug reports. The studies conducted on the bug repositories utilize the issues labeled as bug or other requests.

ISEC '15, February 18 - 20, 2015, Bangalore, India

ACM 978-1-4503-3432-7/15/02…$15.00
http://dx.doi.org/10.1145/2723742.2723751

Issues reported in issue tracking systems can be a bug, a feature request, documentation request or any enhancement request. The information about the category of issues also helps developers in doing the post mortem analysis of a project for the type of issues reported [26]. The empirical studies done on the issue reports show that majority of issue reports are either mislabeled or are not labeled at all [13]. This has motivated the need of accurate classification of issue reports as bug or other request.

In Jira, issue tracking system, QA engineer is also involved in the process of bug fixing. QA is an expert and responsible member, who takes care of the bug fixes but due to large number of issue reports reported daily, there is less motivation in changing the category of the issue when the problem is fixed [13]. In a study conducted on closed issue reports of three open source software systems [4,5,6] from Jira, it has been observed that the label given to the issue reports about bug or improvement is not correct [13]. Herzig et al manually classified more than 7000 closed issue reports from five popular open source software systems to analyze the accuracy of already labeled reports. It took them 90 days to complete the job [13]. Their findings state that 33.8% of closed issue reports are misclassified. Moreover, manual analysis of issues is time consuming, tough and may be error prone. This is a motivation to build a system that automatically and correctly classify issue reports present in issue tracking systems. Moreover, this system can be used either to validate label already assigned by QA to a issue report or to assign label to newly reported bug.

The prior work to distinguish bugs from other requests is done by [1,8,10,19,21]. Thung et al manually classified the dataset and applied machine learning algorithms for bug classification [8]. Pingclasai et al applied an information retrieval model, LDA (Latent Dirichlet Allocation), followed by machine learning algorithms for classification of issues into bugs and other requests [20]. Popular topic models like LDA usually cluster words that tend to frequently co-occur together into the automatically generated topics. LDA is a topic modeling approach and not a classification approach. Its purpose is to group large dataset into abstract topics. The limitation of LDA is that there is no criterion to find the optimal number of topics for any dataset. Moreover, the topics generated are abstract and cannot be correlated with each other.

To overcome the limitation of LDA, a fuzzy logic based approach is proposed for automatic categorization of bugs. Fuzzy logic has been successfully used for the text classification task [22,27,28]. Issue reports are also textual documents as most of information present in them is in natural language. So, the

Table 1. Misclassified data from sample bug reports

	Issueid	Title	Type assigned	Manually reclassified [13]
HTTPClient	HTTPCLIENT-697	HttpClient give same message when proxy/http endpoint is down	Bug	Improvement
	HTTPCLIENT-295	HttpClient loops endlessly while trying to retrieve status line	Improvement	Bug
Lucene	LUCENE-2805	SegmentInfos shouldn't blindly increment version on commit	Bug	Improvement
	LUCENE-2397	SnapshotDeletionPolicy.snapshot() throws NPE if no commits happened	Improvement	Bug
Jackrabbit	JCR-2142	InternalValue should implement QValue.discard() for BINARY types	Bug	Improvement
	JCR-2926	DefaultProtectedPropertyImporter masks several fields from parent, causing potential derived classes to not perform correctly	Improvement	Bug

problem of bug categorization can also be seen as similar to text classification.

Bugs and other requests are termed as fuzzy sets since it is difficult to identify sharp boundaries of the words among two. Moreover, unlike previous approaches [1,8,10,19], there is no dependency of machine learning algorithms as classification is done by fuzzy approach itself. This is the first paper to the best of our knowledge that has applied fuzzy logic for bug classification. Fuzzy set theory has been applied on the bug database for the task of bug triaging by Ahmed Tamrawi et al [2,3]. The objective of this paper is to explore the possibility of using fuzzy set theory technique to classify issues either as bugs or other requests. In Fuzzy sets, elements have degree of membership. Fuzzy sets are powerful due to the fact that they allow a term to have partial membership in two or more sets. Moreover, it is computationally simpler as compared to LDA.

We analyze issues from three software systems: HTTPClient[4], Jackrabbit[5] and Lucene[6]. The data shown in table 1 are mislabeled samples collected directly from the source repositories [4,5,6]. It shows the issue id, its title, type assigned to it and the manually re-classified type by Herzig et al [13]. HTTPClient report 697 is categorized as bug for it shows the same error message when proxy server or the http endpoint was down. There is currently no way to determine if the error occur due to proxy server or http endpoint. So, it is addition of a new functionality hence it should be termed as improvement and not bug. Similarly Lucene report 2805 is reported as bug and describes a problem of version increment by SegmentInfos on the assumption that there are always changes. The solution is that the two classes DirReader and IW track the changes and should notify the SIS whenever there are changes. Hence it is not the existing error but improvement of existing code. Jackrabbit 2142 complain of a bug that currently Jackrabbit-core always extracts the BLOBFileValue in order to free resources. Since InternalValue now implements QValue this could be achieved on the InternalValue directly. However, currently the base implementation is inherited instead of dealing with the BLOBFileValues. Hence it is a requirement of additional feature. We have applied fuzzy logic on the issue report data to categorize them into bugs and other requests for this experiment. Results conclude that fuzzy model alone gives better performance as compared to conventional classification models used with LDA. A comparison of performance on the same dataset with our proposed approach of bug categorization using fuzzy logic shows that our approach gives more accurate results as compared with the three classification models when used with LDA (LDA followed by AD tree algorithm, LDA followed by Naïve Bayes algorithm and LDA followed by decision tree algorithm) [20].

The rest of the paper is organized as follows. Section2 discusses related work. Section 3 shows a motivating example. Section 4 illustrates basic concepts of fuzzy logic. In section 5, we introduce our proposed approach. A description of the data used for experiment is presented in section 5. Section 6 portrays the experimental validation of proposed approach. Section 7 elaborates the evaluation criteria used to measure the performance of our proposed approach. Results are discussed in section 8.Section 9 discusses threats to the validity of our work. Concluding remarks and future work is presented in section 10.

2. RELATED WORK

Early efforts made for bug categorization mainly use data mining and information retrieval techniques. The problem of automated labeling of bugs has been previously addressed by some researchers [1,8,10,15,19,21]. Several approaches have been proposed to categorize the issues as bugs or other requests.

Giuliano et al [1] investigates whether the problem reported is a bug or an enhancement. Due to the complexity of manual classification, they have restricted the number of bug reports to be 600 for each system. The study is conducted on 1800 manually classified issues of Mozilla, Eclipse and Jboss. Title and description features are used for the task of classification.

Experiment is conducted using three machine learning techniques: alternating decision trees, Naïve Bayes classifiers and logistic regression. The conclusion from their study shows that the information present in issue reports in the form of terms present is enough to classify bugs from non bugs.

Classification of software bugs is also done by Nagwani et al. [19]. They used textual similarity of bug attributes to form clusters and then cluster labels are then matched with bug taxonomic terms to find whether the reported issue is a bug or not

bug. The algorithm is based on the frequent terms occurring in the bug reports.

Ferdian Thung et al [8] worked for the Categorization of bugs using Orthogonal Defect Classification. Experiment was conducted on 500 manually labeled bug reports of Mahout, Lucene and OpenNLP.. Their experimental dataset include textual features from the bug reports as well as its corresponding bug fixing code. Bug report title and description is used from the bug reports. Abstract syntax trees are constructed and 46 features like comments, looping structure, method invocations, number of lines added or deleted etc is taken from the code. SVM is used for classification. And an average accuracy of 77.8% is achieved.

Herzig et al [13] also conducted a study to differentiate issues as bug and feature. They have manually classified 7000 bug reports from five open source software systems: HTTPClient, Jackrabbit, Lucene from Jira issue tracker and Rhino and Tomcat5 from Bugzilla issue tracker. Their observation reveals that 33.8% issues were misclassified. They were able to finish the job in approximately 90 days which proves the importance and requirement of automated and accurate labeling. They have made certain rules to divide the issues among six categories comprising of bug, enhancement request, improvement, documentation, refactoring and others. We have made use of their data for our experiment. We have conducted experiment on 5591 bug reports from three software systems for our experiment.

Pingclasai et al [20] used topic modeling approach for bug classification. Latent Dirichlet Allocation (LDA), an information retrieval model is applied to classify issues among bug and other requests. They utilized the dataset of three software systems from the five manually classified dataset by Herzig et al. Our experiment also makes use of these three dataset HTTPClient, Jackrabbit and lucene as used by Pingclasai et al in their experiment. Results obtained from our approach are compared with their topic modeling approach on the same dataset.

Lofti Zadeh [16] proposed the theory of fuzzy set. Fuzzy techniques have been applied for natural language processing. A fuzzy similarity approach for text classification task is proposed by Widyantoro et al [7]. The experimental dataset include text documents comprising of stories from Reuters-21578, discussion from UseNet NewsGroups and textual documents in html format from onlineNews articles. Relation between a term and a category is depicted using the membership degree. Similarity between a document and a category is calculated using the fuzzy operators. Fuzzy similarity for test documents are calculated and compared using various utility operators like Drastic, MinMax, Bounded, Hamacher, Einstein and Algebraic.

Tamrawi et al [3] proposed Bugzie, an approach based on fuzzy set theory to find the appropriate bug fixer. They have chosen seven software projects including Eclipse and Jazz. They have used the correlation between bug reports and the developers with the help of membership function. It incorporates selection of fixers from a subset of some recent fixers which provides more accurate results. They have experimented with training the model by varying number of terms with respect to each developer and found that limited but significant number of terms gives more accurate and time efficient results.

Our aim is to build an automatic and more accurate bug categorization approach. In this paper, we propose to use fuzzy based approach to categorize issue reports either as bug or other request. Fuzzy approach is simple, easy to use and is able to alone classify issue reports in a bug or other request without use of LDA and the machine learning algorithms.

3. MOTIVATING EXAMPLE

Issue report data for this study is extracted for Httpclient, Jackrabbit and Lucene from the Jira bug repository. Although, the issue tracking system is generally considered for reporting errors present in the system but are also used to manage other software activities like some improvement or a new feature request. The information present in the issue reports is not of good quality and reporters generally do not have adequate knowledge about the other things which makes the issue report of low quality.

Figure 1. A misclassified sample bug report

A sample issue report for Lucene is shown in figure 1. An issue report contains several fields like issue id, title, description, type, priority, component, label etc. For issue id LUCENE-2255, type field indicates that it is a bug. The information present in the description field indicates that the bug reporter is confused if it is a bug or a request for improvement. And, the type of issue is written as bug. Manual classification also reveals that this is not a bug but an improvement request. This scenario suggests the need for accurate labeling.

4. BASIC CONCEPT OF FUZZY LOGIC

Fuzzy logic was proposed by Lofti A zadeh [16]. It is a mathematical tool which deals with uncertainty. The fuzzy theory provides a mechanism that enables appropriate human reasoning capabilities. Fuzzy sets that represent fuzzy logic provide means to model the uncertainty associated with imprecision and lack of information about a problem. The uncertainty in the issue reports

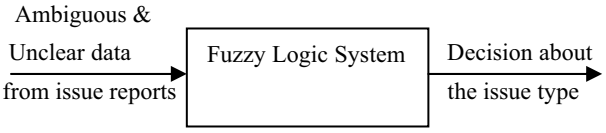


Figure 2. Input and Output of a Fuzzy system

exists due to lack of knowledge or ignorance and due to syntax of natural language itself. Uncertainty in the bug reports classification also exists due to the fact that there is no crisp set of terms or words that can be attributed to either a bug or a feature request. The input to the fuzzy system is the ambiguous and unclear data of issue reports and the output is the decision in terms of correct category as shown in figure 2. Dr Zadeh proposed the set membership idea to make suitable decisions when uncertainty occurs [16]. The theory of fuzzy logic is based upon the notion of relative graded membership. The basis of theory lies in making the membership function lie over a range of real numbers from 0.0 to 1.0.

Construction of membership function depends upon the individual perception of the data. The membership function is written as μ. A fuzzy set in the universe of discourse is defined as

$A= \{(x, \mu A(x))| x \varepsilon U\}$, where $\mu A(x) \varepsilon [0.0, 1.0]$

Where A is the fuzzy set and μA(x) is the degree of membership of x in A and indicates the degree with which x belongs to A.

An important property of fuzzy set is that it allows partial membership. The fuzzy set is characterized by [0.0, 1.0]. The values 0.0 and 1.0 describe "not belonging to "and belonging to "a conventional set respectively and values in between represent "fuzziness".

Fuzzy logic operates quite differently from probability. In fuzzy logic, the sum total of membership of a term towards categories needs not to be 1 as in the case of probability theory.

The membership in a fuzzy set need not be complete i.e. member of one fuzzy set can be member of other fuzzy set in same universe. And the sum of the membership of a term towards various categories can be greater than 1.

5. PROPOSED APPROACH

This section describes the proposed approach for categorization of issues into bugs or other requests using fuzzy logic. The membership degree of relevant terms towards each category as well as mechanism to calculate fuzzy similarity for a new report is presented. Fuzzy logic operates on the concept of membership. Membership of a term indicates the belongingness of a term towards a category. The value of membership score points to the relation of a term with a category. The membership score is denoted as μ(t), which lies in the range of [0.0, 1.0]. The membership value for each unique term from the set of training data indicates the correlation between the bug reports of one category and the terms used to describe it. The membership score for each term corresponding to each category is calculated as follows.

The unique terms from issue reports of both the categories are extracted. The term frequency i.e. number of times it occurs in each category is calculated. The membership function for a term t_i is obtained by dividing the frequency of term in reports of one category for which we are calculating the membership function with the frequency with which it occurs in reports of both the categories.

Let R= { $(b_1, C(b_1))$, $(b_2, C(b_2))$, ..…$(b_n, C(b_n))$} be the training dataset available where b_1, b_2…..b_n are bug reports and $C(b_1)$, $C(b_2)$, ……. $C(b_n)$ indicate the category of bug report b_1, b_2 ,….. b_n respectively. C_i represents C_{bug} and C_{req} the bug and other request category respectively. The unique terms as extracted from reports of both categories are represented as t_1, t_2…..t_n. The significance of a term towards a category is indicated by the fuzzy relation.

Equation (1) shows the formula used for calculating the membership function. The membership function for a term t_i for bug category is denoted as $\mu t_i(C_{bug})$ and is calculated by dividing the total number of term frequency of term t_i in reports of bug category by the total number of occurrences of term t_i in all categories. The bug and other request category is represented by C_{bug} and C_{req} respectively.

$$\mu t_i(C_{bug}) = \frac{\sum tf(t_i) \quad \{ t_i \varepsilon R \wedge C(R)= C_{bug}\}}{\sum tf(t_i) \quad \{ t_i \varepsilon R \wedge C(R)= C_{bug} \vee C(R)= C_{req} \}} \qquad (1)$$

$\mu t_i(C_i) \varepsilon [0.0,1.0]$

Similarly, the membership function for other request category is calculated by interchanging C_{bug} with C_{req}.

The output of the training phase is two fuzzy sets in this case. Each fuzzy set denotes the membership function of a term towards the category. Let {(t_1, $\mu t_1(C_{bug})$, (t_2, $\mu t_2(C_{bug})$,……(t_n, $\mu t_n(C_{bug})$} and {(t_1, $\mu t_1(C_{req})$, (t_2, $\mu t_2(C_{req})$,……(t_n, $\mu t_n(C_{req})$}be the two fuzzy set. First one indicates the term t_1, t_2.....t_n and their membership with the bug category. Similarly, second one shows the same terms with their membership score for other requests category.

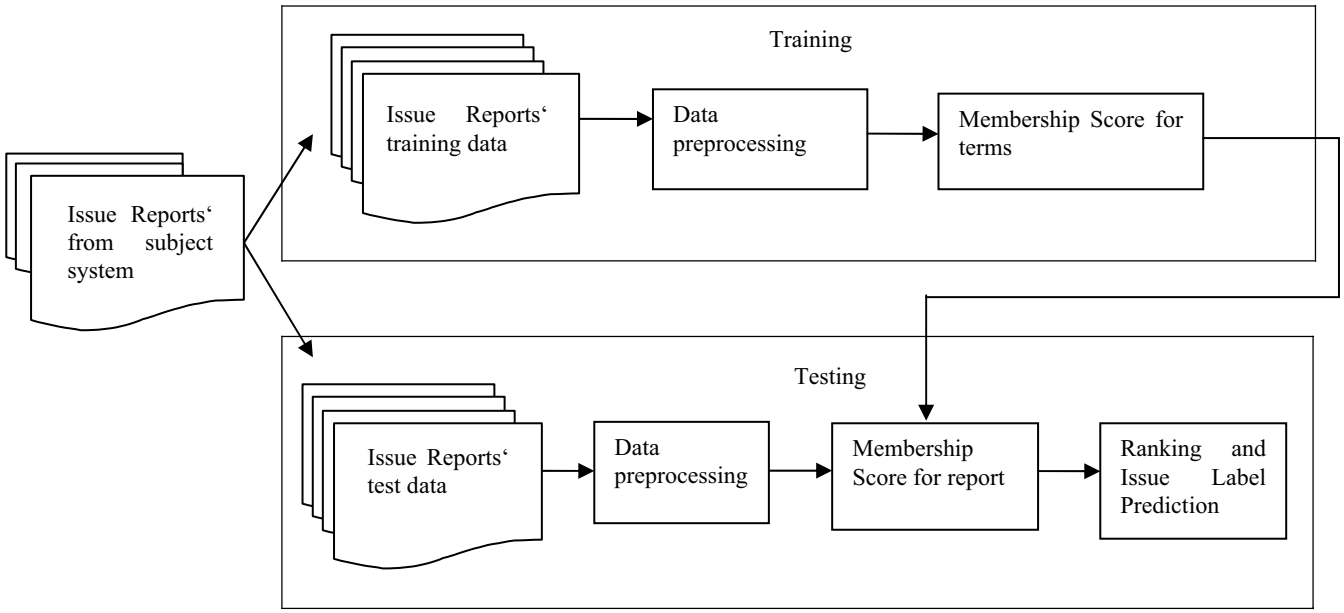
If the membership score of any term is zero for bug data set, it indicates "non membership" and means that the term does not belong to bug category. A membership score of 1 for a term signify "full membership" and implies that the term belongs to the given category only. The higher value of membership score for one category indicates the more contribution of a term towards that category.

Fuzzy Similarity of test issue reports towards each category is calculated by combining the membership score of terms co-occurring in a test report. According to [9], the membership score of a bug report with respect to the union is calculated as in equation 2. Membership score for bug and other request category for each test issue report is calculated using equation 2.

$$\mu_B(C_i) = 1 - \prod_{ti \varepsilon B} (1 - \mu t_i(C_i)) \qquad (2)$$

The sum total of membership score for any issue report is not always 1 as it is expected in case of probability. This is because the membership of a report is independent in one category as compared to its membership in any other category. If a word has a membership score of 0.7 or 70% in one category, then according to probability theory, it can have only 30% probability to occur in other category. But it can still have a higher membership like 0.6 or 60% or more depending upon its occurrence in other category.

Figure3. Block Diagram of Experimental Setup



6. EXPERIMENTAL VALIDATION OF PROPOSED APPROACH

Experimental work consists of two phases: training and testing and is shown in figure 3. Implementation is done in java [11]. We have divided selected bug report dataset in 80:20 ratios for training and testing. Bug or other request categories are expressed by the terms used in the collection of training dataset of bug reports or other requests respectively. Other requests may be any feature enhancement, documentation request or some improvements.

A term can be a member in both categories. A set of unique terms is extracted from bug reports as well as other requests. During training, we apply data preprocessing steps and then membership score for both categories with respect to each unique term is calculated using Equation 1 in section 5.

During testing, we use the output of training phase i.e. membership scores for each term, to calculate the membership score of a test report towards the categories. Membership score for a test bug report represents the membership of its terms towards each category. Text present in the title field from issue reports is used in testing. Same preprocessing steps are applied as was done in training phase. The membership score for the test issue report for each category is calculated as the union of the membership scores of all the unique terms co-occurring in the test report using equation 2 in section 5. With the membership values of a test report computed for two categories, we have a numeric value for each one of them. A comparison among the membership score obtained for the bug and other request category is done and the category having highest membership score for the test report is assigned as the category label to the test report.

6.1 Dataset Used

We have conducted experiment on three software systems: HTTPClient, Jackrabbit and Lucene. Herzig et al manually classified bug reports of these three software systems from Apache Jira bug repository [4,5,6]. The manually classified dataset in the form of issue id, manually classified type, actual type from issue report is available [23] and downloaded by us. We have also downloaded issue reports for three subject software system from Jira bug repository. The manually classified issues are matched with the downloaded issues on issue id. Issue id, title and manually classified type of the bug reports are stored in Mysql database for further experimentation.

Pingclasai et al [20] also conducted experiment of bug classification on these three software systems with the categories assigned by Herzig et al. We have chosen the same dataset for our experiment as used by Pingclasai et al and compared our results with their approach.

A brief description of the three systems is given below.

HTTPClient [4]: The Apache HttpClient library simplifies handling HTTP requests. It is a Client-side implementation of the HTTP protocol, useful in constructing tests of HTTP-based servers and applications. It also provides reusable components for client-side authentication, HTTP state management, and HTTP connection management [4].

Jackrabbit [5]: Apache Jackrabbit is an open source content repository for the Java platform. A content repository is a hierarchical content store with support for structured and unstructured content, full text search, versioning, transactions, observation and more [5].

Lucene [6]: Apache Lucene is a high-performance, full-featured, open source text search engine library written entirely in Java. It is technology suitable for nearly any application that requires full-text search, especially cross-platform [6].

Table 2 shows the total number of issues, number of bug reports and number of other requests chosen for three software systems HTTPClient, Jackrabbit and Lucene from Apache Jira. We have divided the bug reports and other requests data available in the

ratio of 80:20 for training and testing. 80% of data is selected randomly for training and rest 20% data is used for testing. Training dataset and testing dataset for issue reports labeled as bug as well as other request is disjoint i.e. there is no record similar in the two datasets. Table 3 shows the total number of issues labeled as bug, number of bug reports used for training and number of bug reports used for testing for HTTPClient, Jackrabbit and Lucene. Table 4 depicts the training and test count corresponding to three subject software systems for other requests.

Table 2. Overview of dataset chosen

Maintainer	**Apache**		
Tracker Type	**Jira**		
Project	**HTTPCLIENT**	**JACKRABBIT**	**LUCENE**
No of Reports	745	2402	2443
No of Bugs	305	938	697
No of Other Requests	440	1464	1746

Table 3. Number of issues used for training and testing bug reports

	Bugs	**Training bugs**	**Test bugs**
HTTPCLIENT	305	244	61
JACKRABBIT	938	750	188
LUCENE	697	557	140

Table 4. Number of issues used for training and testing other requests

	Requests	**Training Requests**	**Test Requests**
HTTPCLIENT	440	352	89
JACKRABBIT	1464	1171	293
LUCENE	1746	1396	350

6.2 Data preprocessing

Data preprocessing is applied to remove noise from the data and following steps have been applied for the task.

6.2.1 Data Preparation

Bug reports are extracted as XML documents. For each bug report, we extracted its unique id, its title and the type assigned to it. Bug reports of three software systems: HTTPClient, Jackrabbit and Lucene are extracted separately and stored in Mysql database [17].

6.2.2 Sampling

Random sampling technique is used on the bug reports to reduce any bias. 80% of the total reports for each software system is selected for training and stored in separate tables. The test dataset consists of remaining 20% reports from each software system and is also stored in three tables.

6.2.3 Feature selection

A bug report contains various features like title, description, version, operating system etc. We have observed that the concept is well versed in the title of the bug report. So, we have chosen title of the bug report for further experimentation. Moreover, description is generally too long and may also contain code patches sometimes. As a result, the main concept or idea is superseded by so many words and accuracy of a model may be compromised as it creates confusion for a model to deal with so many words. The smaller set of training data from the title field would also enhance the speed of categorization.

6.2.4 Tokenization and Case folding

The title of the bug reports is chosen for further experimentation. The document is separated by the white spaces and all the words have been extracted. Special symbols, numerals and punctuation marks which do not contribute much in classification of bug and other requests are removed from the extracted words. Case folding is done to convert all capital letters to small letters as it brings more clarity and removes ambiguity.

6.2.5 Stop word removal

There are many words used for completion of sentences and does not contribute much to the topic such as ‘a’, ‘the’, ‘this’ etc. These words have been removed to emphasis more on context related words.

7. Evaluation Criteria

Precision, recall, accuracy and F-measure are the measures used to assess the effectiveness of our approach. A confusion matrix is calculated to measure precision and recall.

Confusion matrix is also called an error matrix which represents the information about the exact label and the predicted label for software bugs. A confusion matrix consists of four parts: True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). A true positive value indicates the number of labeled bug reports that were also relevant. A false positive value specifies the number of bug reports labeled with no relevant class. A false negative value means the number of relevant bug reports that were not retrieved. A True Negative value implies the number of bug reports that were not relevant and also not retrieved by the algorithm.

7.1 Precision

Precision is calculated as the ratio of the number of TP (True positives) to sum of Number of True positives and False positives (TP+FP) [12]. It is a measure of correctness and is expressed as the ratio of the number of correctly labeled software bugs retrieved to the total number of correctly and incorrectly labeled

records retrieved. A precision score of 1.0 is perfect and means that every software bug is accurately labeled but does not describe if all software bug reports are retrieved.

$$\text{Precision} = \frac{\text{Number of TP}}{\text{Number of TP} + \text{Number of FP}}$$

7.2 Recall

Recall is computed as the ratio of the number of True Positives to the sum of number of True positives and False Negatives [12]. It is a fraction of relevantly labeled software bugs that are retrieved and expressed as the ratio of the number of correctly labeled software bugs to the total number of correctly labeled and the actual software bug reports that could not be retrieved. A perfect Recall score of 1.0 represent that all bug reports of particular label has been retrieved but does not state how many others are also incorrectly labeled.

$$\text{Recall} = \frac{\text{Number of TP}}{\text{Number of TP} + \text{Number of FN}}$$

7.3 Accuracy

The accuracy is the proportion of the total number of bug reports that were correct. It indicates how close a measured value is to the actual value [24]. An accuracy of 100% means that the measured values are exactly the same as the given values. It is determined by using the equation:

$$\text{Accuracy} = \frac{\text{Number of TP} + \text{Number of TN}}{(\text{Number of TP} + \text{Number of TN} + \text{Number of FP} + \text{Number of FN})}$$

7.4 F-measure

F-measure is a harmonic mean of precision and recall [24]. The value for F-measure lies between 0 and 1. A larger value shows a higher classification quality.

$$\text{F-measure} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

8. RESULTS AND DISCUSSIONS

The proposed approach is tested for number of bug reports for each software system. 20% of the total extracted issue reports is used for testing. Details about the number of issues used for testing are depicted in table 3 and 4. In total, testing was done on 1121 issue reports. The experimental results demonstrate that our approach is more effective as compared to the topic modeling approach.

The confusion matrix generated for our proposed approach for HTTPClient, Jackrabbit and Lucene are shown in table 5, 6 and 7 respectively. The confusion matrix helps in visualizing the performance of the algorithm. In our experiment, it depicts the number of bugs and other requests correctly and incorrectly classified.

The effectiveness of proposed approach is measured in terms of four most popular measures: precision, recall, accuracy and precision. Table 8 shows the precision, recall, F-measure and accuracy for the three subject software systems. The fuzzy logic method shows the precision of 0.9 for HTTPClient, 0.79 for Jackrabbit and 0.83 for Lucene. We obtain the recall value of 0.77, 0.78, 0.85 for HTTPClient, jackrabbit and Lucene respectively. It shows an ac accuracy of 87% for HTTPClient, 83.5% for jackrabbit and 90.8% for Lucene. F-measure which is a Harmonic mean of precision and recall is calculated and a score of 0.829 is achieved for HTTPClient, 0.784 for Jackrabbit and 0.839 for Lucene is achieved.

Table 5. Confusion matrix from fuzzy approach for HTTPClient

HTTPCLIENT	Bugs	Requests
Bugs:61	47	14
Requests: 89	5	84

Table 6. Confusion matrix from fuzzy approach for Jackrabbit

JACKRABBIT	Bugs	Requests
Bugs:188	147	41
Requests: 293	38	255

Table 7. Confusion matrix from fuzzy approach for Lucene

LUCENE	Bugs	Requests
Bugs:140	119	21
Requests: 350	24	326

Table 8. Precision, Recall, Accuracy and F-measure for three software systems

	Httpclient	Jackrabbit	Lucene
Precision	0.9	0.79	0.83
Recall	0.77	0.78	0.85
Accuracy	0.87	0.835	0.908
F -measure	0.829	0.784	0.839

Table 9 shows the comparison of the F-measure as calculated from the fuzzy based approach proposed by us with the F measure value obtained from three classification algorithms- AD tree, Naïve Bayes and Logistic Regression used after topic modeling approach on the three software systems.

Since, we have experimented on the same dataset as was used by Pingclasai et al [20]. We have directly compared the results of proposed fuzzy approach with the results obtained from topic modeling approach used by them. They have classified their dataset with the three machine learning algorithms: AD Tree,

Naïve Bayes and Logistic Regression. Figure 4, 5 and 6 shows the comparison of F-measure score for HTTPClient, Jackrabbit and Lucene respectively.

Table 9. Comparison of Fuzzy approach with LDA

	HTTPClient	Jackrabbit	Lucene
Proposed fuzzy based approach	0.829	0.784	0.839
AD Tree	0.72	0.715	0.805
Naïve Bayes	0.742	0.738	0.782
Logistic Regression	0.725	0.762	0.803

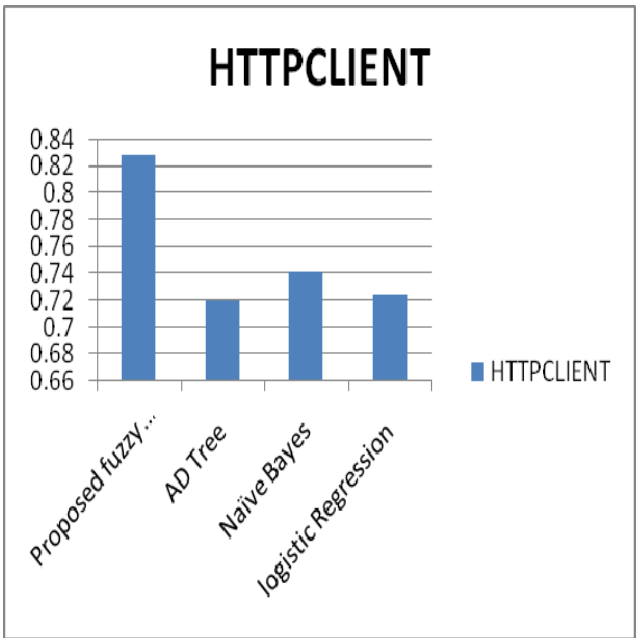


Figure 4. Comparison of F-measure score for HTTPClient

Figure 4 illustrates the comparison of the F-measure score of Fuzzy approach with the LDA approach for the three machine learning algorithms in HTTPClient. As it is clear from the figure, that the fuzzy approach gives better results as compared to rest of three. There is a highest difference in the F measure value in case of HTTPClient dataset. Fuzzy approach gives 9% more accurate results than the machine learning approaches.

Figure 5 exemplify the results for Jackrabbit and a comparison of Fuzzy approach with the rest three for the F-measure score. Value of 0.784 is obtained from fuzzy approach and 0.715 from AD tree, 0. 738 for naïve bayes and 0.762 for Logistic regression. Fuzzy approach still supersedes among all.

Figure 6 shows the F-measure comparison for Lucene. The F-measure value for fuzzy approach is 0.839 while LDA can generate a score of 0.805 from AD tree, 0. 782 for naïve bayes and 0.803 for Logistic regression. It is clear from the figure that fuzzy approach gives better results in this case too.

The result shown depicts the usefulness of our experiment as they prove the applicability of fuzzy approach in context of bug classification.

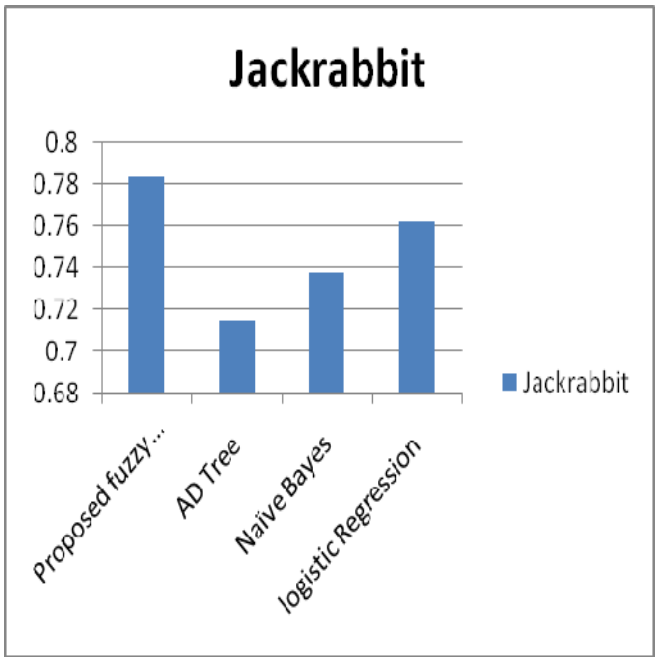


Figure 5. Comparison of F-measure score for Jackrabbit

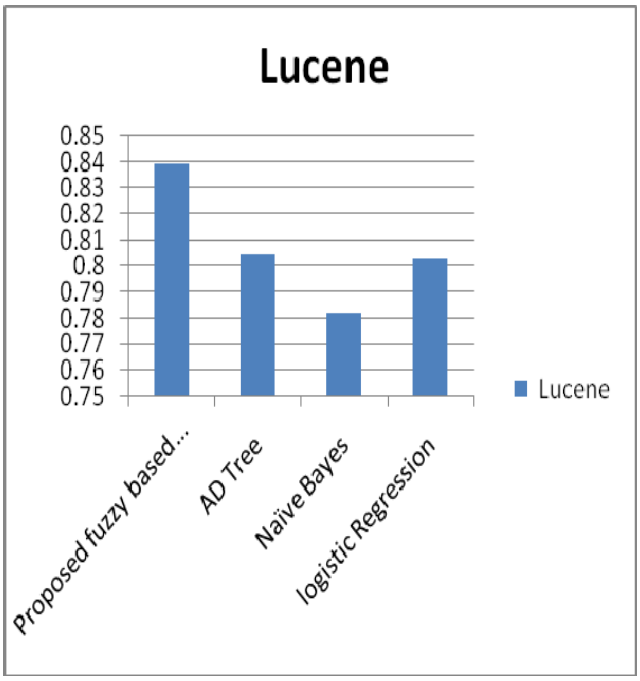


Figure 6. Comparison of F-measure score for Lucene

9. THREATS TO VALIDITY

These are some possible threats to the validity of our approach. Since we used the textual information present in the bug reports for bug categorization, and the accuracy of the results also depend upon the bug report quality. The system may not generate accurate results in the absence of appropriate training data. For example, if for a test bug report, no word comes in the training data; it won't be able to classify it into any of the given categories.

In case of accurate and varied set of bug reports, there are more chances of getting good results.

Since random sampling technique is used for selection of training and validation data. In case of totally different samples chosen, there may be variation in the results obtained.

10. CONCLUSION AND FUTURE WORK

This paper has proposed an approach for automatic categorization of bug reports using fuzzy logic. The labeling of bug reports is done in three phases. First, text from the bug reports is pre processed. Second, the Fuzzy technique is applied and third the labeling is done using scores obtained after fuzzification. The results for bug labeling obtained from fuzzy based approach are compared with the results from LDA based (machine learning) approach. The results signify that fuzzy based approach is effective. In particular, we have shown that fuzzy logic is one promising direction in the task of bug categorization. We have introduced the feasibility of Fuzzy approach for bug classification. Other machine learning algorithms like SVM can be explored for further improvement in accuracy.

Although the results achieved by fuzzy logic are promising and more accurate than the previous approach still they are not perfect. The motive of fuzzy logic is to provide best results which may not be optimal. A qualitative Study on the misclassified bug reports from our proposed approach as compared to the manually classification by Herzig et al [13] will elaborate the reasons or shortcomings of our approach.

In future research, several directions are worth exploring. We need to find if the training data used from one system can be used and worth for categorizing the data for other system. Second, we have applied supervised learning technique; the applicability of unsupervised learning techniques for bug report categorization can be explored. This is important as in many cases we are not sure if the labeled data used for training is accurate or not.

The future task can be to experiment with other fields and attributes present in the bug report like description, comments etc for their contribution in the task of labeling. The same experiment can be conducted on more bug reports from other large and different software systems.

REFERENCES

[1] A., Giuliano, K. Ayari, M. D. Penta, F. Khomh, and Y.Gaël Guéhéneuc. Is it a bug or an enhancement?: a text-based approach to classify change requests."In Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds, p. 23. ACM, 2008.

[2] A. Tamrawi, T. T. Nguyen, J. M. Al-Kofahi, and T. N. Nguyen. Fuzzy set and cache-based approach for bug triaging. In Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering, pp. 365-375. ACM, 2011.

[3] A. Tamrawi, T. Nguyen, J. Al-Kofahi, and T.N. Nguyen. Fuzzy Set-based Automatic Bug Triaging. In International Conference on Software Engineering. ACM, 2011.

[4] Apache. http://hc.apache.org/

[5] Apache. http://jackrabbit.apache.org/

[6] Apache. http://lucene.apache.org/

[7] D. H. Widyantoro, J. Yen. "A fuzzy similarity approach in text classification task." In IEEE International conference on fuzzy systems, vol. 2, pp. 653-658. 2000.

[8] F. Thung, D. Lo, and L. Jiang. Automatic Defect Categorization. 19th Working Conference on Reverse Engineering, pp. 205-214. IEEE, 2012.

[9] G.J. Klir and Bo Yuan. Fuzzy Sets and Fuzzy Logic:Theory and Applications. Prentice Hall, 1995.

[10] Javed, Muhammad Younus, and Hufsa Mohsin. An Automated Approach for Software Bug Classification. Sixth International Conference on Complex, Intelligent and Software Intensive Systems, pp. 414-419. IEEE, 2012.

[11] Java: http://www.java.com/

[12] J. Han, M. Kamber, and J. Pei. Data Mining: Concepts and Techniques. The Morgan Kaufmann Series in Data Management Systems, 2006.

[13] K. Herzig, S. Just, and A. Zeller, "It's not a bug, it's a feature: how misclassification impacts bug prediction," in Proc. ICSE '13, 2013, pp. 392–401.

[14] K.K. Chaturvedi and V. B. Singh. "Determining bug severity using machine learning techniques." CSI Sixth International Conference on Software Engineering, pp. 1-6. IEEE, 2012.

[15] Linares-Vásquez, Mario, Collin McMillan, Denys Poshyvanyk, and Mark Grechanik. On using machine learning to automatically classify software applications into domain categories. Empirical Software Engineering 1-37, 2012

[16] L. A. Zadeh, "Fuzzy sets", Information and Control, Vol. 8, 1965, pp. 338-353.

[17] Mysql: www.mysql.com

[18] N. Jalbert and W. Weimer. Automated duplicate detection for bug tracking systems. DSN 2008. IEEE International Conference on Dependable Systems and Networks With FTCS and DCC, pp. 52-61. IEEE, 2008

[19] 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

[20] N. Pingclasai, H. Hata, K.I. Matsumoto. Classifying Bug Reports to Bugs and Other Requests Using Topic Modeling. In Software Engineering Conference (APSEC, 20th Asia-Pacific (pp. 13-18). IEEE, 2013

[21] P. Anbalagan, M. Vouk. On predicting the time taken to correct bug reports in open source projects. IEEE

International Conference on Software Maintenance, pp. 523-526. IEEE, 2009.

[22] R. Khoury, F. Karray, Y. Sun, M. Kamel, & O. Basir. Semantic understanding of general linguistic items by means of fuzzy set theory. IEEE Transactions on Fuzzy Systems, 757-771, 2007.

[23] http://www.softevo.org/bugclassify

[24] S.N. Sivanandam and S.N.Deepa: Principles of Soft Computing.Wiley, 2008.

[25] S. C.Tsai, J. Y. Jiang, S.J. Lee. A Mixture Approach for Multi-Label Document Classification. International Conference on Technologies and Applications of Artificial Intelligence (pp. 387-391). IEEE, 2010

[26] Tan, L., Liu, C., Li, Z., Wang, X., Zhou, Y., & Zhai, C. Bug characteristics in open source software. Empirical Software Engineering, 1-41, 2013.

[27] W. Liu, N. Song, A fuzzy approach to classification of text documents. Journal of Computer Science and Technology, 18(5), 640-647, 2003

[28] Y. Sun, R. Khoury, F. Karray, O. Basir. Semantic context classification by means of fuzzy set theory. Proceedings of 2005 IEEE International Conference on Natural Language Processing and Knowledge Engineering, (pp. 250-255). IEEE, 2005