Research Article

International Journal of Advanced Computer Research, Vol 11(56)
ISSN (Print): 2249-7277 ISSN (Online): 2277-7970
http://dx.doi.org/10.19101/IJACR.2021.1152031

Classifying bug reports to bugs and other requests: an approach using topic modelling and fuzzy set theory

Mohammed D. Ali[1]* and Ahmed A. Abusnaina[2]
Master Program in Software Engineering, Birzeit University, Ramallah, Palestine[1]
Faculty of Engineering and Technology, Department of Computer Science, Birzeit University, Ramallah, Palestine[2]

Received: 10-July-2021; Revised: 16-September-2021; Accepted: 20-September-2021

Abstract
Stakeholders of a software system usually deliver bug reports to a software bug tracking system to report problems they encounter during the use of that system. After that, those incoming requests are assigned to the proper technicians to be analysed and fixed. However, issue reporters frequently misclassify bug requests as non-bug and vice versa. This problem is called bug report misclassification. In fact, it is very costly in terms of time and effort as developers have to manually reclassify those requests to then be able to take the appropriate action. Accordingly, the automatic classification of incoming new reports would be of a valuable demanded feature in bug tracking systems. Careful analysis for literature related to this problem was carried out. It has been found that a hybrid approach combining between topic modelling as a feature extraction technique and the fuzzy logic as a classification technique is promising but rarely utilized approach. In this work, a combined approach of topic modelling and fuzzy logic was introduced to classify bug reports to bugs and non-bugs. The proposed approach was validated using three open-source projects. Finally, the conducted experiments have shown that proposed classifier achieves significant and competitive predictive accuracy.

Keywords
Software maintenance, Bug reports, Fuzzy sets, Topic modelling, Classification, Bug tracking systems.

1.Introduction
1.1Overview
Bug Tracking Systems like Bugzilla, Gnats and Jira [1, 2] are a type of systems that are used to support the maintenance activities during the lifecycle of software systems [3]. They are a communication mean between users and developers to report and fix issues [4]. In particular, these systems commonly manage the following two types of reported requests: corrective requests (bugs), and perfective requests (non-bugs) such as new feature request, improvement request, documentation, and so on [3–7]. Software bugs are unavoidable. Therefore, more and more bug reports are continually submitted to bug tracking systems. Specifically, statistical information in [8] states that more than 30 maintenance requests are reported daily. Hence, bug tracking systems are considered a rich source of historical information that could be crucial to support several software engineering tasks such as bug priority assignment, bug severity identification, developer recommendation for bug resolution [9], bug localization [10], defect prediction, duplicate bug report identification [7], and bug categorization [4].

The perfect support for these software engineering tasks reduces the costs of maintenance activities [6]; decreases developer time and effort; and generally, achieves better overall reliability [2, 11]. Thus, to perfectly enrich such these tasks, bug tracking systems need to be enhanced with an automatic classification feature to correctly and accurately classify reported issues to bugs and other requests.

1.2Research Motivation
Misclassification of the reported bug reports is a common problem that threatens the quality of software engineering tasks and activities that are carried out during software system life cycle. Simply, it happens because bug tracking systems lack the feature of automatic classification of bug reports. Instead, the

*Author for correspondence

process itself is accomplished manually. Consequently, its quality completely depends on human understanding [3, 9], and the number of issue reports submitted during a specific time window. On the other hand, manual classification of issue reports is a time-consuming and error-prone process [5, 6, 8]. According to [12], researchers manually examined more than 7000 issue reports during 90 days and they found that 30.8% of inspected reports were really misclassified. As well as, bug reports may be originally submitted to bug tracking systems with missing labels which also leads to the same problem [5]. As a result, high maintenance costs are incurred [13], uncertainty is increased, and project planning is negatively affected [4]. As well, the costs of manual categorization of bug reports increase dramatically according to the size and complexity of the software system being supported [8].

Accordingly, this work will address the following research problem: *Given a new bug report and historical bug reports, predict the type of the new bug report with respect to either Bugs or Other Requests*.

1.3Research questions

In order to resolve the research problem mentioned above, the following two research questions are going to be answered by this study:

- RQ1: In terms of performance metrics, how effective is the proposed approach to classify bug reports as compared with baseline approaches?
- RQ2: How much performance could be achieved by tuning the number of topics hyperparameter that model a bug report?

1.4Main contributions

In this paper, a new approach is proposed to classify Jira bug reports into bugs and non-bugs, and to validate the type of already submitted report. Exclusively, this approach was built using a hybrid method that combined topic modelling as a natural language processing technique with fuzzy logic as a classification mechanism. Considering the conducted experiments, it is concluded that the highest F-measure ranges between 78-84% which is considered a significant improvement to the baseline works. Moreover, this paper concludes that topic modelling using a number of topics equals 20 topics is the optimal hyperparameter setting that could be used to model Jira bug reports.

2.Literature review

This section sheds lights on the most related works. Then, it analyses related works by focusing on two factors; the technique used to extract useful features from textual data of bug reports, and the main classifier technique used to classify bug reports to either bugs or other requests.

In Pingclasai et al. [3]. proposed a two-phase approach to automatically classify bug reports to either bugs or other requests. First, topic modelling was used to convert textual data into numeric membership vectors. Then, ADTree, Naive Bayes classifier, and Logistic Regression were utilized to classify reported bug reports. As a result, F-measure metric score was ranging between 0.66-0.76, 0.65-0.77, and 0.71 0.82 for HTTPClient, Jackrabbit, and Lucene datasets respectively.

In Chawla and Singh [5] proposed an automatic classification of bug reports using a fuzzy logic-based approach. In particular, the proposed approach used the concept of membership in fuzzy logic. First, they extracted the terms form bug reports for the both categories. Then, the membership value for a term in a specific category was calculated by dividing term frequency in that category with the term frequency of both categories. After that, a bug report was classified to the category that gets the higher overall membership value from summing the membership values of all terms contained in that bug report. The results showed that the fuzzy logic-based approach achieved better results than the machine learning based approach.

In Zhou et al. [6] proposed a three-phase approach for automatic bug report classification. The first phase used Multinomial Naive Bayes Classifier that analysed the textual fields of bug reports to extract three features defining three levels of possibilities: high, middle, and low. The second phase used Data grafting technique to link between structured and unstructured features. The third phase applied Bayesian Net Classifier as a machine learning technique to finally achieve competitive results.

In Terdchanakul et al. [9], conducted a study aiming at comparing performance of N-gram IDF based classification models with topic-based models. First, they applied N-gram IDF to create features vector, then it was pre-processed using correlation-based methods for feature selection purposes. After that, Logistic Regression and Random Forest models were trained. The results showed that proposed approach had a higher performance than the topic-based models. In Qin and Sun [10] built a model for bug classification using Long Short-Term Memory (LSTM). The followed approach included three general steps; data

preparation aimed to eliminate the noise from the textual fields of the bug document; representing words as vectors; and applying RNN. Cleary, results showed that LSTM-based technique outperformed both topic-based and N-gram IDF-based techniques as shown in terms of f-measure performance metric.

In Hammad et al. [13] proposed a new automatic and unsupervised approach to identify the feature or topic that handled by a given bug report based on finding and analysing the similar previous bug reports using a technique called agglomerative hierarchical clustering (AHC) which produces a set of clusters of bug reports and also labels each cluster with tags the represent the topic existing in that cluster.

All the reviewed papers have made several textual data preparation steps that aimed to reduce the effects of noise and increase the performance of the main classifier models used. For example, tokenization and stemming were employed in [2, 3, 6, 7, 8, 10, 13, 14]. However, some papers [11, 15] intentionally did not apply stop words removals because this action may influence the meaning of the reported bug report.

With regard to feature extraction techniques, the majority of the reviewed papers have used term-based methods to convert textual data to numeric useful features [1, 5, 7, 10, 13]. However, the literature has stated that topic modelling techniques are the most appropriate techniques to extract features from textual content in form of topic distributions for each bug report because topic modelling is very similar to the manual methodology followed by human to classify textual data. Basically, this method is based on linguistic classification rules [3, 14]. Despite of this fact, it has been found that only one paper [3] has used it. The reasons for limited usage of topic modelling in this context are the complex configuration to apply topic modelling algorithms and poor scientific documentation on how to use them [16, 17]. Besides, there is no silver bullet to determine the optimal number of topics that do well for a given data set [3, 5].

On the other hand, regarding the main classification technique used to classify bug reports, it is found that supervised machine learning was the dominant field for this problem since it was used by many works [1, 3, 4, 6, 7, 9, 11, 15]. Whereas, two of the surveyed papers have used unsupervised machine learning techniques: clustering [13], and association rule mining [18]. Only three papers [5, 14, 19] have used fuzzy logic-based approaches to support the current research problem.

More importantly, although machine learning did well in supporting the current research problem, fuzzy logic-based classification techniques still have a strong motivation to be used. This is because bug reports naturally have some vagueness in its content that came out from the lack of information and the syntax errors as well [5]. Moreover, according to literature, it is unfair to hardly classifying a bug report into just two categories. Conversely, it is likely that bug report may have a certain degree of membership to both categories [19].

In conclusion, To the best of our knowledge, this study proposes a newly approach to tackle the misclassification problem of reported bug reports. This approach will be based on topic modelling and the fuzzy logic. Moreover, to overcome the limitation of using topic modelling algorithms identified by literature, the proposed approach will utilize Stanford LDA Topic Modelling Toolbox [20].

3.Background

This section presents some background information with regard to the main ideas employed through proposed approach such as the process of bug reporting, topic modelling, and fuzzy logic.

3.1Bug reports

Software bug reports are artifacts that manage the faults of software systems. Because they contain useful information about the submitted bugs, developers depend on the description of these defects in resolving the corresponding bugs [17]. Many papers have described the bug reporting process [17, 18]. Initially, the newly incoming bug report is given the state “Unconfirmed”. When a developer verifies severity of the bug report, its status is updated to “New”. Then, the bug report is assigned to an appropriate developer or fixer. After that, when the developer resolves the bug, “Resolved” state is given. Otherwise, “New” state is given a gain to start a new cycle. Finally, Once the bug is resolved successfully, the cycle is finished and the state is changed to “Closed”. *Figure 1* shows a sample issue report taken from Lucene project. Surprising, the assigned type for this report is bug. However, the manual classification process for this report proved that this report is a request for new feature. In fact, it is misclassified.

Figure 1 A sample misclassified bug report from Lucene project

3.2Topic modelling

Classically, topic modelling was originally used as a method of clustering and changing large unstructured documents into structured ones [16]. In particular, Topic Modelling is a statistical model that can extract the "topics" from a corpus of textual documents. Each discovered topic is actually consisting of a certain number of terms that existed or occurred in the textual document. As well as, each document may be related to one or more topics in form of membership values. In the terminology of topic modelling, it could be said, for example, document (d) contains the topic (t) with probability 30% [16, 17]. In the same way, if two bug reports belong to the same topic or topics; they absolutely have some similarity. Therefore, topics are considered a kind of features that could be used as inputs for several data mining techniques [16, 17] to train models that are capable to offer certain solutions.

As it was mentioned earlier, this work utilized Latent Dirichlet Allocation (LDA) to find the topics and their terms from a group of historical bug reports. So that, it can be possible to know the bug reports that are similar to the newly submitted one.

Moreover, application of LDA algorithm requires to set and configure four hyperparameters. N refers to the number of topics; R symbolizes the number of iterations; α and β are association factors. Higher value of α means that a bug report will associate with many topics in a higher probability. As well, higher value of β means that a topic will associate with many terms in a higher probability [17].

3.3Fuzzy set theory

Classical or crisp set is a group of distinct values that have specific and precise membership value to that set. For example, given a set A. An element x, in the universe X, has either 0% or 100% of membership in the set A [21]. Fuzzy logic is a type of multi-valued logic. It deals with approximate reasoning rather than exact one. It resembles the way followed by humans to make decisions based on if-then rules.

Importantly, it is used to model the Vagueness associated with a specific problem. Moreover, fuzzy logic deals with fuzzy sets which were proposed by Lotfi Zadeh in 1965 [21].

Fuzzy sets are considered as extension of crisp or classical sets. In this context, elements of a fuzzy set have a multi-valued membership relation toward that set, on the contrary of the elements of classical sets which have only a binary membership value, 0, or 1. The membership function μÃ(x) of an element x in the fuzzy set Ã produces degree of membership for that element in the given fuzzy set. It is given in the range [0,1], where 1 indicates that the element is completely in the set, 0 indicates that the element is completely not in the set, whereas values between 0 and 1 indicate that the element has a partial membership in the set.

Another essential point, fuzzy logic does not function similar to probability theory. In fuzzy logic, the sum of memberships of an element towards a set is not essentially equals one. Oppositely, in probability theory it needs to equal one [5].

4.Methods

4.1Data collection

Proposed approach was validated using datasets from three open-source projects: Jackrabbit [22], Lucene [23], and HttpClient [24]. These datasets have been

used by the majority of the surveyed papers and especially in baseline papers. Each dataset contains two fields bug id and summary.

In addition, each dataset has corresponding corrective dataset that contains the actual data type. Actually, these corrective datasets were manually classified by Herzig et al. [12] and they were downloaded from [25].

The following lines introduce a brief information of these three projects:

- Apache HTTPClient: it implements HTTP protocol from client side. It streamlines HTTP requests and used in testing HTTP-based applications. It is also helps in client-side authentication, state management [5, 24].
- Apache Jackrabbit: it is a content repository for java used to store structured and unstructured data, text search, version management, and others [5, 22].
- Apache Lucene: it is an open-source Java library for full text search that could be used by any application [5, 23].

Table 1 shows dataset sizes and modelling split size used in the conducted experiments.

Table 1 Overview of datasets and split size

Item	HTTPClient	JACKRabbit	Lucene
#Reports	745	2402	2443
#Bugs	305	938	697
#Non bugs	440	1464	1746
#Training	596	1922	1954
#Tesing	149	480	489

4.2Overview of the proposed approach

This section details the steps used in the proposed approach. In general, it consists of two main phases: topic modelling phase which is used to convert textual data of bug reports to numeric features in a form of topics, the classification phase which is based on fuzzy logic concepts.

4.3Topic modelling phase

In this phase, topics are automatically extracted from a corpus of bug reports. A topic is a group of semantically related terms that co-occur in a report. Hence, it would be possible to discover latent semantic relationships and provide effective analysis on report contents. In addition, inputs of this phase are the textual data of bug reports. Whereas., the output is topic membership vectors, one vector for each bug report.

This phase can be divided into the following steps:

1. Data parsing: in this step, the collected bug reports were parsed. Particularly, fields containing textual information were determined besides to the linkage column with corrective data sets.
2. Applying topic modelling: given a textual bug report, Latent Dirichlet Allocation (LDA) extracts a predefined number of topics that represent the overall idea this report is talking about. Fundamentally, implementation of LDA algorithm requires identifying the number of topics (k) to be extracted from the textual contents. To be bias-free, this approach experimentally tuned the value of (k) on every 10 successive number of topics from 50 topics. This is an important experimental hyperparameter to be tuned because the higher value of (k), the more general topics are trained. Oppositely, the lower value of (k), the more specific topics are probably to be extracted [3, 17]. Moreover, this approach has applied Stanford topic modelling toolbox [20] that implemented LDA algorithm. Before that, many common preprocessing steps were performed such as data cleaning operations, tokenization, stemming, stop words removing, and data filtering.
3. As a result, each topic membership vector consists of report id and (k) number of topics with 0 or 1 value for each topic indicating the existence of that topic in the given report. Specifically, this mechanism was used by Pingclasai et el. [3] to convert probability values produced by LDA into 0's, and 1's.

4.4Fuzzy logic phase

In fact, this phase operates on two data sets. The first one is the dataset resulted from the topic modelling phase in a form of topic membership vector for each bug report. The second data set is the corrective dataset that contains the actual and manual classified type for each bug report that will later be compared with the predicted type to evaluate the performance of the classification model.

Importantly, this phase is replicated from [5] that employed the terms occurred in bug reports. Instead, the proposed approach has employed the topics occurred in bug reports. Key aspects of this phase are listed below:

1. The idea of fuzzy logic for bug reports classification:

Given a newly submitted bug report to a bug tracking system. This report will be automatically given the category (bug, non-bug) that includes the larger

number of historical reports that are similar to this new report. Hence, fuzzy similarity is proposed.

2. Bug report representation in fuzzy logic:
The concept of membership and fuzzy sets was utilized. Bug and Non-bug categories can be represented by two fuzzy sets. Membership of a topic expresses the belongingness of that topic in the corresponding fuzzy set. Membership value is symbolized using μ(t). It is contained within the closed interval [0-1].

3. Definition:
The topic frequency is how many times does the topic occur in the category (Bug, Non-bug) of each bug report.

4. On training data, calculate the membership score for topics:
Given a topic $topic_i$, and c_{bug}, c_{non} categories. the membership score $\mu topic_i(c_{bug})$ for a topic ($topic_i$) toward a category (c_{bug}) can be calculated by dividing topic frequency in the reports of category (c_{bug}) on topic frequency in the reports of both categories (c_{bug}, c_{non}). And vice versa with regarding to the membership score $\mu topic_i(c_{non})$ for a topic ($topic_i$) toward a category (c_{non}).

Memberships in each category have been calculated using the equations (1) and (2).

$$\mu Topic_i(c_{bug}) = \frac{\sum TF_{bug}(topic_i)}{\sum TF_{bug}(topic_i) + \sum TF_{non}(topic_i)} \quad (1)$$

$$\mu Topic_i(c_{non}) = \frac{\sum TF_{non}(topic_i)}{\sum TF_{bug}(topic_i) + \sum TF_{non}(topic_i)} \quad (2)$$

where TF donates the topic frequency in the given category.

5. The output of training phase are membership scores for the given topic into the both fuzzy sets (bug and non-bug categories). Hence, zero membership score means that the topic does not belong to the given category. Also, one membership score means that the topic belongs to the given category only. whereas, a membership score in the open interval (0,1) means that the topic partially belongs to the given category.

6. On test data, the fuzzy similarity is obtained by calculating the membership score for the given bug report:
Given a bug report rep_i, and c_{bug}, c_{non} categories. the membership scores $\mu rep_i(c_{bug})$ for a report (rep_i) toward a category (c_{bug}) can be calculated by multiplying the membership scores of topics occurring in the report according to (3) and (4) formulas. And vice versa with regarding to the membership score $\mu rep_i(c_{non})$ for a report (rep_i) toward a category (c_{non}).

Here, equations (3) and (4) have been used:

$$\mu Rep(bug_i) = 1 - \Pi\,(1 - \mu Topic_i(bug_i)) \quad (3)$$

$$\mu Rep(non_i) = 1 - \Pi\,(1 - \mu Topic_i(non_i)) \quad (4)$$

7. Ranking and Classification:
In this step, the new bug report is given a membership score for both categories: bug and non-bug. In particular, it is assigned to the category that obtains the highest membership score calculated according to (3) and (4).

8. Importantly, fuzzy logic operates differently from the probability theory. Thus, the membership score of a bug report towards a category needs not to equal 100%. For example, if a topic has 60% membership score in one category, then probability theory expects 40% score in the other category. Whereas, fuzzy logic allows to have any score because the membership score is obtained independently for each category [5].
9. Finally, as the actual bug report type is given from the corrective datasets and the tested bug report is predicted by the proposed approach. Then, many suitable evaluation metrics have been utilized to assess how performant is the proposed approach in ranking and classifying bug reports. *Figure 2* illustrates those two phases.

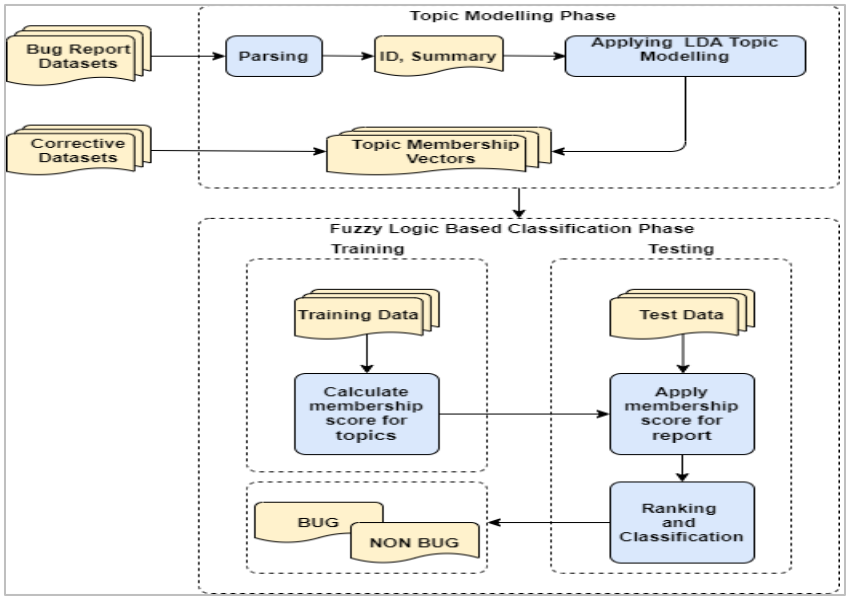

Figure 2 The main steps of the proposed approach

4.5 Motivating example

Table 2 shows a motivating example illustrating the fuzzy logic implementation for bug report ranking and classification according to the equations (1), (2), (3), and (4).

Assume that a training dataset contains five topics, and only the three topics asterisked occur in the test report.

For each report in the five bug reports, the topic frequency in the both fuzzy sets (bugs and not bugs) were calculated using the topic-based datasets. Then, the membership value for each report in the bug fuzzy set is computed using equation (1) and it is shown in the second column in the bellow table, i.e.,
3/ (3+7) = 0.30, 6/ (6+5) = 0.55, 4/ (4+4) = 0.50, 5/ (5+9) = 0.36, and 4/ (4+1) = 0.80.

Similarly, the membership value for each report in the not-bug fuzzy set is computed using equation (2) and it is shown in the third column in the bellow table, i.e., 7/ (7+3) = 0.70, 5/ (5+6) = 0.45, 4/ (4+4) = 0.50, 9/ (9+5) = 0.64, and 1/ (1+4) = 0.20.

According to equation (3) and assuming that the newly coming bug report has the three topics asterisked in the bellow table; the membership value for the new coming bug report in the bug fuzzy set is calculated as follows:
1 - [(1 - 0.30) * (1 - 0.50) * (1 - 0.36)] = 0.78.

Similarly, according to (3) and assuming that the new coming bug report has the three topics asterisked in the bellow table; the membership value for the new coming bug report in the non-bug fuzzy set is calculated as follows:

1 - [(1 - 0.70) * (1 - 0.50) * (1 - 0.64)] = 0.95.
Consequently, the new bug report is classified to (not-bug) as it has the higher membership value in the not bug fuzzy set.

Another important point, the membership value of a report toward any of bug or not-bug fuzzy sets is insensitive to noises. For example, noise resulted from those topics that are irrelevant to the reported issue. To illustrate that, assume that a bug report consists of two topics; the first is relevant with 0.6 membership value, the second is irrelevant and rarely exists with 0.1 membership value. Then the membership value of the report containing both topics toward the bug fuzzy set equals 1- [(1-0.6) * (1-0.1)] = 0.64. whereas the membership value of the report containing only the relevant topic toward the bug fuzzy set equals 1- [(1-0.6)] = 0.60, which it is not much smaller than 0.64.

Table 2 Motivating example

Topics	TF_{bug}	TF_{non}	μ_{bug}	μ_{non}
T1*	3	7	0.30	0.70
T2	6	5	0.55	0.45
T3*	4	4	0.50	0.50
T4*	5	9	0.36	0.64
T5	4	1	0.80	0.20

5.Experimental design

This section presents the implementation of the main two phases in this approach besides to the data preprocessing operations that have been applied.

5.1Data preparation

This paper has applied Stanford topic modelling toolbox [20] that was written by Stanford natural language processing research group. This toolbox implements LDA algorithm through several types of scripting steps that have been written in Scala programming language. In particular, it enables importing and preparing spreadsheets data; training several LDA models such as LDA and LDA labeled; configuring several parameters; and producing usable and accurate results.

Data preparation is a pipeline process that extracts the textual content and converts it into a usable form by LDA algorithm, besides it guarantees having a model with improved performance. In this work, data preparation pipeline included the following steps:

- Identifying the columns including the bug-id, and the textual content to be analyzed.
- Remove punctuation symbols.
- Change texts into lower case to decrease the number of unique words.
- Remove words that are shorter that three characters in order to having a meaningful word.
- Removing words that appear in less than four bug reports because these words are considered rare and do not contribute much in measuring the similarity.
- The words appearing more than 30 times were removed because they are also unimportant for measuring similarity.
- Bug reports that have missing textual contents were discarded.
- Importantly, as seen in some literature, the implemented approach did not apply stemming technique because it may remove the negative phrases that they could have a special meaning in bug reporting process.

5.2Topic modelling

There are many parameters to configure LDA topic modelling algorithm. Mainly, there are four configurable parameters to be set. First, number of topics. Second, number of iterations required to train the model. Third, the association factor α; higher value of α means that a bug report is likely to be associated by many topics. Fourth, the association factor β; higher value of β means that a topic is likely to be associated by many terms. *Table 3* shows values for LDA hyperparameters.

Table 3 LDA parameters

Parameter	Value	Notes
maxTopics	10-50	Tuned experimentally
maxIterations	1500	[17]
topicSmoothing(α)	0.01	[17]
termSmoothing(β)	0.01	[17]

Furthermore, there are two types of learning techniques to train topic models. The first one is called collapsed variational Bayes approximation (CVB0LDA). It uses all the available CPU cores in the machine, so that it can reach the convergence state faster. The second type is called collapsed Gibbs sampler (GibbsLDA) which requires less memory for training. Therefore, this research used Gibbs sampler for training and inference [20].

Finally, the main output of the topic modelling phase is topic distribution probabilities (p) for each report. Then, these probabilities were converted into 0 or 1 indicating whether a topic appears in a report or not. In particular, a one value was given when p>0; otherwise, zero was given.

5.3Classification

As a result of the previous phase, 15 experiments were conducted in this phase; each experiment was implemented against one dataset out of the three projects with each value for number of topics (10, 20, 30, 40, and 50).

Data of each experiment is divided to training and testing. Then, a special programming module was built implementing the formulas (1), (2), (3), and (4) introduced in the previous section to develop a fuzzy similarity-based classifier. Importantly, 10 runs for each experiment were executed and the average of performance values was recorded for each run.

More important, from machine learning perspective, imbalanced data is a challenging problem [26]. It occurs when the distribution of bug reports across classes is imbalanced. However, the resulted topic-based data sets were statistically checked by finding number of instances in both positive and negative classes. Consequently, it was confirmed that they were balanced and they could be used to train classification models. Importantly, according to the logic used in the proposed approach which is built based on relative weighting for both positive and negative classes; there will be no influence to the imbalanced data (if existed). As well as, topic modelling is considered as a kind of feature selection technique.

5.4Validation and evaluation methods

• Resampling technique

From machine learning perspective, there are many evaluation methods that could estimate how well the classifier model does on unseen data. Specifically, these methods are train and test splits, k-fold cross validation, leave one out cross validation, and repeated random test-train splits [26]. In this paper, random test-train splits method was used. For each dataset out of 15 datasets, it was divided into training and testing parts using size ratio of 80:20. The training data was used to train the fuzzy logic algorithm. The test data was used make predictions and evaluate results against the actual results. This process was repeated 10 times and the average performance was recorded.

• Confusion matrix

It is a tool that provides useful information about the behavior of the models in terms of the following [27]: Ture Positives (TP): the number of bug reports that are actually bugs. True Negative (TN): the number of non-bug reports that are actually non-bugs. False Negative (FN): the number of non-bug reports that are actually bugs. False Positive (FP): the number of bug reports that are actually non bugs.

• Performance metrics

This work used three metrics in order to measure the performance of the proposed classifier. They are precision, recall, and F-measure. They measure the performance by focusing on the positive class (bugs in this case) [27, 28].

- Precision computes the ratio of the total number of correctly classified reports as bugs divided by the total number of all reports that are actually bugs. Precision = TP / (TP + FP).
- Recall finds the ratio of the total number of correctly classified reports as bugs divided by the total number of reports that are correctly classified into bug and non-bug. Recall = TP / (TP + TN).
- F-measure is used to summarize both the Precision and the Recall in one metric. The higher F-measure is, the higher both Precision and Recall are at the same time. F-measure = (2 * Precision * Recall) / (Precision + Recall).

6.Results

This section presents the research results in four main points: Newly submitted Jira bug reports can be automatically classified into either bug or non-bug and indicating to which extent they belong to both classes. This research manages to classify bug reports to bugs and other requests in two ways. The first way is crisp classification which means that a bug report is classified completely as a bug or completely as a non-bug. The second way is soft classification which indicates to which degree a bug report belongs to the both categories bug and non-bug.

To the best of our knowledge, this is the first approach among other fuzzy logic approaches that raises this feature. This new feature offers many advantages with regard to the software maintenance activities in terms of efforts and costs, as compared with the classical first methods. *Table 4* shows two sample test reports that were classified using both hard and soft methods by the proposed approach.

Table 4 Two sample classified bug reports using proposed approach

Membership	**LUCENE REP-1003**	**LUCENE REP-2526**
Membership in Bug fuzzy set	0.69	0.81
Membership in Not-bug fuzzy set	0.80	0.69
Classified	Not bug	Bug

2. Lucene project shows the best performance out of the other two projects.

Table 5 shows the evaluation results of the three projects HTTPclient, Jackrabbit, and Lucene using the proposed approach in terms of precision, recall, and F-measure metrics. In particular, the F-measure metric varies between 0.63 – 0.81 for HTTPclient, 0.70 – 0.78 for Jackrabbit, and 0.78 – 0.84 for Lucene. Moreover, the results illustrate that the improved performance is recorded for Lucene project.

3. Regarding Jira projects, number of topics equals 20 is the optimal tuned value that showed the best predictive performance.

Table 5 shows number of topics that was used in model training and fitting. It is seen that the best performance score is gained when using 20 topics to train the model. However, increasing or decreasing number of topics parameter does not imply any trend or pattern between number of topics and the overall performance. Finally, this work also proves that LDA topic modelling algorithm was correctly configured.

Table 5 Performance of topic-based fuzzy logic classifier (P: Precision, R: Recall, F: F-measure)

#Topics	Dataset	P	R	F
10	HTTPClient	0.59	100	0.74
	Jackrabbit	0.57	0.94	0.71
	Lucene	0.65	0.98	0.78
20	HTTPClient	0.77	0.85	0.81
	Jackrabbit	0.70	0.88	0.78
	Lucene	0.73	0.98	0.84
30	HTTPClient	0.68	0.83	0.75
	Jackrabbit	0.70	0.86	0.77
	Lucene	0.70	0.94	0.80
40	HTTPClient	0.75	0.77	0.76
	Jackrabbit	0.62	0.80	0.70
	Lucene	0.69	0.95	0.80
50	HTTPClient	0.51	0.80	0.63
	Jackrabbit	0.61	0.88	0.72
	Lucene	0.68	0.99	0.81

Figure 3 shows a chart of performance curve for each of the three research subjects' models. In x-axis appears the number of topics used to train each model. In y-axis appears the F-measure score that is achieved by the proposed model. Actually, it indicates that each software project may have its own optimal value for the number of topics that could achieve the maximum possible performance in classification, i.e., 20 topics in this case.

Finally, according to the shown evaluation results, it is concluded that hybrid classification models that combine topic-based data and fuzzy logic are able to classify bug reports to two categories with significant score of performance. In particular, the software projects that can be benefited from this approach are projects managed by JIRA bug tracking system.

4. The proposed approach outperforms the baseline approaches in terms of performance metrics. Generally, fuzzy logic approaches outperform machine learning approaches.

The proposed approach used topic-based data to train a classification model by leveraging the fuzzy set theory. The first baseline approach used word-based data to train a classification model using the fuzzy set theory [5]. The second baseline used topic-based data to train three models using several machine learning techniques [3]. The last baseline used word-based data to train classification models using the same machine learning algorithms utilized by the second approach [11]. More importantly, all of the models are compared by using the same datasets. *Table 6* shows a comparison summary among the four approaches in terms of F-measure performance metric. The asterisk (*) indicates the models that have been outperformed by the proposed approach in this paper.

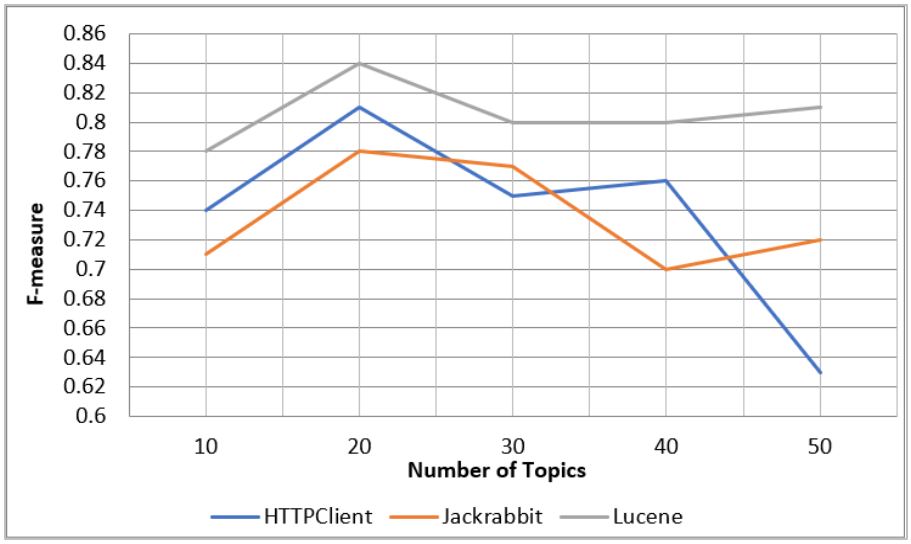

Figure 3 Model performance with number of topics

Table 6 Comparison between proposed approach and other three ones in terms of F-measure

Model		HTTP	JACK	LUCENE
Topic-based logic	Fuzzy	0.81	0.78	0.84
Word-based logic [5]	Fuzzy	0.82	0.78	0.83*
Topic-based learning [3]	Machine	0.73*	0.74*	0.80*
Word-based learning [15]	Machine	0.71*	0.70*	0.67*

Significantly, the two fuzzy logic-based approaches, including the proposed one, have outperformed machine learning based approaches in terms of F-measure performance metric in all of the three datasets.

regarding the two fuzzy logic-based approaches, including the proposed one. It is seen that topic-based proposed approach has outperformed word-based one in Lucene project, and it has a similar performance score in Jackrabbit project, and it has achieved a slightly lower performance in HTTPclient project.

7.Discussion

The topic-based models, either fuzzy logic or machine learning, have outperformed other models that mainly used feature selection techniques. This can be due the nature of the process itself. Actually, topic models extract topics from textual contents. Each topic contains several terms which appear in the text. Importantly, the terms appearing in a topic are semantically related [16]. However, feature selection techniques loss this advantage. Accordingly, this clearly explains why measuring the similarly of textual contents using topic modelling is more efficient than using feature selection. In addition, it also explains why there is no trending regarding increasing or decreasing the number of topics.

Furthermore, word-based fuzzy logic model has outperformed topic-based fuzzy logic (proposed approach) in HTTPclient project, and it shows slightly similar performance in Jackrabbit and Lucene projects. In fact, LDA algorithm configurations could be the reasons. Specially, increasing the value of the association factor β will make the topic more generic. Thus, decreasing value of β will produces one-term topics which is similar to word-based approaches. Finally, another justification would explain this result; it is the quality of HTTPclient dataset and to which extent it is rich in useful terms.

The proposed approach is able to indicate how much a report is faulty or non-faulty. This can be determined by comparing the membership value of the report in the both categories bug and not-bug. Importantly, summation of both membership values need not to equal 100%. This is because our research problem has a vagueness nature and not uncertainty problem. This makes fuzzy logic the appropriate technique to model such this problem. Whereas, probability theory is the suitable technique to model uncertainty problems which essentially requires 100% for all events.

Limitations

The proposed approach was validated using three datasets. Each dataset was accompanied by a corrective data set that was manually classified. Hence, any errors in this manual process may produce misleading and unrealistic results. Moreover, the utilized datasets were open source written in java and they were managed by Jira bug tracking system. Accordingly, any change in the used datasets, programming language, or bug tracking systems will affect the generalizability of the proposed approach.

8.Conclusion and future work

This research employed the textual information available in the reported bug reports in bug tracking systems to conduct an automatic approach for classifying bug reports into either bug or other request. As a result, this proposed approach eliminates the need of manual classification process and its accompanied drawbacks.

First, Stanford topic modelling toolbox implementing LDA algorithm was adopted to structure the content of bug reports in form of topics with their probabilities of appearing in the report. These resulted topics were assumed to distinguish bug reports from other type of reports. Second, a model was developed using the fuzzy logic to measure the bug reports similarity as a technique for classification. Several experiments were conducted using three open-source systems with careful configuration. Evaluation of the obtained results leads to two main conclusions. It is concluded that modelling of bug reports in the form of topics is able to classify bug reports. Also, the optimal number of extracted topics that achieved the most improved performance is 20 topics.

With this number of topics, and in terms of performance metrics, the classification model produces F-measure score varies within 0.78 to 0.84 across the three projects. Finally, this approach outperformed two baseline approaches that used machine learning. Moreover, this approach achieves a

relatively similar performance as compared with another baseline approach that used fuzzy logic based on word-based data.

In future work, recommendations are going toward exploring new directions as follows:

- Using cross-project datasets to experiment whether a data of one project can be used to classify data of another project.
- Employing new datasets for new projects and related to bug tracking systems other than JIRA, besides experimenting more richer datasets in textual contents.
- Experimenting several types of LDA algorithms for topic modelling with careful setting for their parameters.
- In terms of machine learning, the proposed approach is called supervised modelling as the actual data labels were given. But these labels are often not available. So that, unsupervised techniques with fuzzy logic are also considered targets for future research.

Acknowledgment

None.

Conflicts of interest

The authors have no conflicts of interest to declare.

References

[1] Pandey N, Sanyal DK, Hudait A, Sen A. Automated classification of software issue reports using machine learning techniques: an empirical study. Innovations in Systems and Software Engineering. 2017; 13(4):279-97.

[2] Xia X, Lo D, Ding Y, Al-Kofahi JM, Nguyen TN, Wang X. Improving automated bug triaging with specialized topic model. IEEE Transactions on Software Engineering. 2016; 43(3):272-97.

[3] Pingclasai N, Hata H, Matsumoto KI. Classifying bug reports to bugs and other requests using topic modeling. In Asia-pacific software engineering conference 2013 Dec 2 (pp. 13-8). IEEE.

[4] Otoom AF, Al-jdaeh S, Hammad M. Automated classification of software bug reports. In proceedings of the 9th international conference on information communication and management 2019 (pp. 17-21).

[5] Chawla I, Singh SK. An automated approach for bug categorization using fuzzy logic. In proceedings of the 8th india software engineering conference 2015 (pp. 90-9).

[6] Zhou Y, Tong Y, Gu R, Gall H. Combining text mining and data mining for bug report classification. Journal of Software: Evolution and Process. 2016; 28(3):150-76.

[7] Chawla I, Singh SK. Automated labeling of issue reports using semi supervised approach. Journal of Computational Methods in Sciences and Engineering. 2018; 18(1):177-91.

[8] Jangra M, Singh SK. Evaluating topic modeling as pre-processing for component prediction in bug reports. In advanced computing and communication technologies 2016 (pp. 465-73). Springer, Singapore.

[9] Terdchanakul P, Hata H, Phannachitta P, Matsumoto K. Bug or not? bug report classification using n-gram idf. In international conference on software maintenance and evolution 2017 (pp. 534-8). IEEE.

[10] Qin H, Sun X. Classifying bug reports into bugs and non-bugs using lstm. In proceedings of the tenth Asia-Pacific symposium on internetware 2018 (pp. 1-4).

[11] Kukkar A, Mohana R. A supervised bug report classification with incorporate and textual field knowledge. Procedia Computer Science. 2018; 132:352-61.

[12] Herzig K, Just S, Zeller A. It's not a bug, it's a feature: how misclassification impacts bug prediction. In 2013 international conference on software engineering 2013(pp. 392-401). IEEE.

[13] Hammad M, Alzyoudi R, Otoom AF. Automatic clustering of bug reports. International Journal of Advanced Computer Research. 2018; 8(39):313-23.

[14] Behl D, Handa S, Arora A. A bug mining tool to identify and analyze security bugs using naive bayes and tf-idf. In international conference on reliability optimization and information technology 2014 (pp. 294-9). IEEE.

[15] Antoniol G, Ayari K, Di Penta M, Khomh F, Guéhéneuc YG. Is it a bug or an enhancement? a text-based approach to classify change requests. In proceedings of the conference of the center for advanced studies on collaborative research: meeting of minds 2008 (pp. 304-18).

[16] Chen TH, Thomas SW, Hassan AE. A survey on the use of topic models when mining software repositories. Empirical Software Engineering. 2016; 21(5):1843-919.

[17] Zhang T, Chen J, Yang G, Lee B, Luo X. Towards more accurate severity prediction and fixer recommendation of software bugs. Journal of Systems and Software. 2016; 17:166-84.

[18] Zolkeply MS, Shao J. Classifying software issue reports through association mining. In proceedings of the 34th ACM/SIGAPP symposium on applied computing 2019 (pp. 1860-3).

[19] Panda RR, Nagwani NK. Software bug categorization technique based on fuzzy similarity. In 9th international conference on advanced computing 2019 (pp. 1-6). IEEE.

[20] http://nlp.stanford.edu/software/tmt/tmt- 0.4/. Accessed: 10-September-2021.

[21] Zadeh LA. Fuzzy sets. In fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh 1996 (pp. 394-432).

[22] http://jackrabbit.apache.org/. Accessed: 10-September-2021.

[23] http://lucene.apache.org/. Accessed: 10-September-2021.

[24] http://hc.apache.org/. Accessed: 10-September-2021.
[25] https://www.st.cs.uni-saarland.de//softevo/. Accessed: 10-September-2021.
[26] Brownlee J. Machine learning mastery with python. Machine Learning Mastery Pty Ltd. 2016; 527:100-20.
[27] Han J, Pei J, Kamber M. Data mining: concepts and techniques. Elsevier; 2011.
[28] Brownlee J. Imbalanced classification with Python: better metrics, balance skewed classes, cost-sensitive learning. Machine Learning Mastery; 2020.

Mohammed Dahesh Ali is the head of system analysis unit in the Information & Communication Technology Center at Al-Quds Open university, Palestine. He received his Master Degree in Software Engineering from Birzeit university, Palestine, in 2021. He received her B.S in Computer Information Systems, from Al-Quds Open University - Palestine in 2006. He is interested in software engineering research with focus on Software Evolution and Maintenance, Mining Software Repositories, and Machine Learning.
Email: modsh.ali@gmail.com

Ahmed A. Abusnaina received his B.Eng. degree in Computer Systems Engineering from Palestine Polytechnic University (PPU) in 2007, and MSc and PhD degrees in Computer Science from Universiti Sains Malaysia (USM). His academic qualifications are in engineering and sciences. However, his work experience is blended in nature with strong ties to education, engineering, neuroscience, as well as the industry. His recent research focuses on utilizing the Computational Intelligence and Neural Networks for wide applications in medicine, biology, physics, engineering and many other research fields. In addition, Brain-inspired Computing by developing more realistic and accurate neurons to the brain neurons is one of the main open research areas.
Email: aabusnaina@birzeit.edu