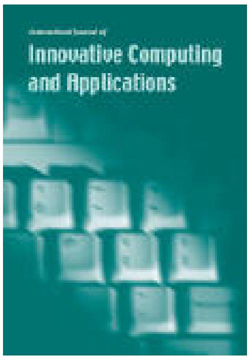


International Journal of Innovative Computing and Applications

ISSN online: 1751-6498 - ISSN print: 1751-648X
https://www.inderscience.com/ijica

Fuzzy modelling techniques for improving multi-label classification of software bugs

Rama Ranjan Panda, Naresh Kumar Nagwani

DOI: 10.1504/IJICA.2023.10056700

Article History:

Received:	24 September 2021
Last revised:	03 February 2022
Accepted:	30 May 2022
Published online:	07 June 2023

Fuzzy modelling techniques for improving multi-label classification of software bugs

Rama Ranjan Panda* and Naresh Kumar Nagwani

Department of Computer Science and Engineering,
National Institute of Technology-Raipur,
Chhattisgarh, India
Email: rrpanda.phd2018.cs@nitrr.ac.in
Email: nknagwani.cs@nitrr.ac.in
*Corresponding author

Abstract: Software bug repositories stores a wealth of information related to the problems that occurred during the software development. Today's software development is a modular approach, with multiple developers working in different locations all around the world. A software bug may belong to multiple categories and can be resolved by more than one developer. For understanding the multiple causes of software bugs and proper bug information management at large bug repositories, better classification of software bugs is needed. In the proposed work, a multi-label fuzzy system-based classification (ML-FBC) is proposed. A fuzzy system is used to compute the membership of software bugs into multiple categories. Then a fuzzy c-means clustering algorithm is used to create various clusters. Once the clusters are created, the cluster-category mapping is done for various software bugs. For a new bug, the fuzzy similarity values are computed, and the created cluster-category mappings are utilised to categorise it. Using a user-defined threshold value, a new bug is classified into multi-label categories. Experiments are carried out on available benchmark datasets to compare the performance measures F1 score, BEP score, Hloss, accuracy, training time, and testing time of various multi-label classifiers. The proposed ML-FBC outperforms existing multi-label classifiers.

Keywords: mining bug repositories; bug information management; fuzzy modelling; multi-class categorisation; multi-label classification.

Reference to this paper should be made as follows: Panda, R.R. and Nagwani, N.K. (2023) 'Fuzzy modelling techniques for improving multi-label classification of software bugs', *Int. J. Innovative Computing and Applications*, Vol. 14, No. 3, pp.141–154.

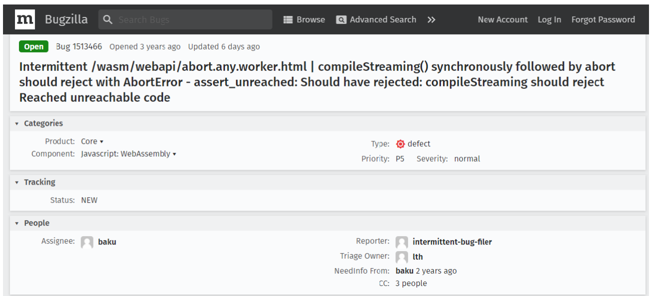
Biographical notes: Rama Ranjan Panda received his MCA from the National Institute of Science and Technology, Berhampur, Odisha, India 2011 and MTech in Computer Science and Engineering from the National Institute of Science and Technology, Berhampur, Odisha, India 2013. He is currently working toward his PhD with Computer Science and Engineering Department, National Institute of Technology, Raipur, India. His current research interests include application of fuzzy logic techniques in mining of software repositories and knowledge discovering in software repositories. He has published more than 13 papers in international journals and conference proceedings.

Naresh Kumar Nagwani has completed his graduation in Computer Science and Engineering in 2001 from the G.G. Central University, Bilaspur. He completed his Post-graduation Master of Technology in Information Technology from the ABV-Indian Institute of Information Technology, Gwalior in 2005 and completed his PhD in Computer Science and Engineering in 2013 from National Institute of Technology Raipur, India. His employment experience includes software developer and team lead at Persistent Systems Limited. Presently, he is working as an Associate Professor of Computer Science and Engineering at the NIT Raipur. He has published more than 105 research papers in various journals and conferences in the field of data mining, text analytics and software engineering.

1 Introduction

In a software development project, the software bug repository is one of the most important repositories as it stores a wealth of information related to the problems that occurred during software development. In the testing phase of the software development life cycle, the tester and quality engineer test individual modules as well as the whole software to identify any problems in different modules of the software. The flaws or defects that are identified by the tester and quality engineer are treated as software bugs (Xia et al., 2016; Yadav et al., 2019). The information about the software bugs is presented in the form of a software bug report and consists of the basic information about the software bug such as bug id, summary, product, component, type, priority, severity, assignee, etc. A sample of a software bug report from the Mozilla bug repository is illustrated in Figure 1.

Figure 1 Bug report 1513466 of Mozilla software project (see online version for colours)



The bug has a bug id of 1513466 and the summary of the bug is written in ital letters. The bug's product is core, and the component is Java script: Web assembly, the type of the bug is defect, the priority is P5, and the severity is normal, as shown in the category section of the software bug report. Similarly, the tracking section shows the status of the bug, and it is new. The people section reveals the names of the assignee as Baku, the name of the reporter, and the triager owner. The importance of different attributes of a software bug report in various research areas related to software bugs is presented by Soltani et al. (2020).

Modern-day software projects use various bug tracking systems such as JIRA, Bugzilla, Eclipse, etc. to automatically generate bug reports (Xi et al., 2019; Herbold et al., 2020). In a large software project, multiple developers are working from various locations, and there is a continuous inflow of a large number of bug reports. Analysing the multiple causes of software failures is very time-consuming and tedious work. Furthermore, one of the major problems in large software bug repositories is bug information management. A proper software bug classification algorithm is needed to address this issue. Proper software bug classification leads to better bug information management and improves the bug triaging process by finding the right fixer for fixing the bugs in a timely manner. Many of the contents of bug reports are textual in nature, and researchers have started using text mining to extract knowledge from these software bug repositories.

Over the years, various machine learning algorithms have been effectively used in the area of text mining for the classification and clustering of text documents based on the features present in the documents. Machine learning algorithms have also been applied effectively in a variety of research disciplines, including bio-medical disease classification (Singh et al., 2020; Govindarajan et al., 2020; Naseem et al., 2021), sentiment analysis (Onan, 2021), e-book classification (Thakur and Patel, 2021), and tree and utility pole classification (Das et al., 2021). It is also effectively used for the classification of software bugs and has produced significant results (Ahmed et al., 2021; Ni et al., 2020; Mohsin and Shi, 2021). A software bug has either been classified as a bug or a non-bug category in a binary software bug classification. However, the majority of software failures occur for multiple reasons related to various modules of software bugs, and these machine learning algorithms are insufficient for understanding the multiple causes of software bugs. For better classification of software bugs, the multiple causes of software bugs need to be addressed, and this can be achieved by designing a multi-label classification algorithm for software bugs.

Today's software development is a modular approach, with multiple developers working in different locations all over the world. A software bug may belong to multiple categories and be resolved by more than one developer. Clustering is one of the popular approaches widely used in data mining to determine similar items and group them into the same clusters. The clustering algorithms are broadly classified as distance-based and model-based approaches (Bei et al., 2021; Wang et al., 2022). Similarly, the clustering algorithm can be divided into two categories: non-fuzzy clustering and fuzzy clustering techniques. Non-fuzzy clustering refers to hard clustering in which a term belongs to exactly one category, whereas fuzzy clustering refers to soft clustering in which a term belongs to many categories. Fuzzy clustering techniques can be used to understand the multiple relationships between software bugs and various categories. Furthermore, most of the features of software bugs are textual in nature, a fuzzy system-based clustering algorithm can be effectively applied to these software bugs to generate a membership matrix that indicates the grade at which technical terms of software bugs belong to various categories (Panda and Nagwani, 2019, 2021). A series of recent studies has indicated that fuzzy clustering algorithms are widely adopted in the fields of image processing (Dong et al., 2022; Farahani et al., 2018; Gao et al., 2022; Ghosh et al., 2021; Kavitha and Saraswathi, 2021; Rubio et al., 2017), multi-label classification (Panda and Nagwani, 2021; Peng and Liu, 2018; Qian et al., 2021), categorical data analysis (Saha et al., 2019), and multivariate data analysis (Sanchez et al., 2017). Similarly, a combination of a fuzzy c-means clustering algorithm and a fuzzy inference system is used for the evaluation of unmanned aerial vehicles. The fuzzy clustering algorithm is utilised to create various clusters, and the fuzzy inference system is used to analyse the expert knowledge about unmanned aerial vehicles (Çolaket al., 2022). In all of the above studies, it was found that the fuzzy clustering algorithms provided statistically significant results and improved the overall performance of the system.

In the last several years, a large number of metaheuristic algorithms have been developed by using the idea of natural phenomena, and these algorithms are being used in various fields to solve complex problems (Fausto et al., 2020; Tzanetos and Dounias, 2021). An intensive investigation is being carried out by Meng et al. (2021) on ten popular metaheuristic algorithms, and they have presented a comparative analysis of these algorithms based on their convergence properties, the importance of technology, and the major challenges in different engineering fields. In order to solve optimisation problems more efficiently, machine learning techniques are also combined with metaheuristics. This integration is done to improve the quality of the solution, the rate of convergence, and the robustness of the system (Karimi-Mamaghan et al., 2021; Talbi, 2021). Similarly, metaheuristic algorithms along with deep learning techniques are used on medical data (Si et al., 2022; Bahaddad et al., 2022), image data (Ahmed and Darwish, 2021), brain-computer interface (Martínez-Cagigal et al., 2022), and biological data (Santander-Jiménez et al., 2022) to solve many complex problems. Metaheuristics can be utilised along with machine learning and deep learning techniques to create more efficient models for improving the performance of various classifiers.

Multi-label learning has been extensively studied and is being investigated in the literature. A multi-label evaluation matrix can be broadly classified as example-based or label-based. It is used for classification as well as ranking purposes (Zhang and Zhou, 2013; Qian et al., 2021). In recent years, several multi-label classification algorithms have been developed for text classification (Al-Salemi et al., 2019; Wu et al., 2020; Xia et al., 2021), pattern recognition and image processing (Zhang et al., 2020; Wang et al., 2021; Tarekegn et al., 2021), recommendation system (Zhang et al., 2020) and data computing (Mei et al., 2020). The widely used multi-label classification algorithms are multi-label K-nearest neighbour (ML-KNN) (Zhang and Zhou, 2007), ranking support vector machine (R-SVM) (Elisseeff and Weston, 2001), and multi-label radial basic function (ML-RBF) (Zhang, 2009). Similarly, various fuzzy similarity measure-based text classification (Jiang et al., 2012; Lee and Jiang, 2013; Gangavarapu et al., 2020), software bug categorisation techniques (Panda and Nagwani, 2019, 2021) are also found in the literature, and these techniques are well suited for software bug analysis.

Numerous classifying algorithms are developed based on the presence of frequent terms, named entities, and categorical terms in software bugs. The discriminative terms present in the software bugs provide much-needed information and knowledge about the software bugs (Zhou et al., 2018; Nagwani and Verma, 2014). The software bugs are categorised into various categories, such as logical, backend, graphical user interface (GUI), data types, memory, operating system (OS), security, build, analysis and enhancement, etc. by matching the categorical terms present in the software bugs (Panda and Nagwani, 2021; Nagwani and Verma, 2014). To determine whether a particular software bug belongs to multiple categories, fuzzy logic can be applied to software bug repositories, as it calculates the membership grade of each software bug towards various categories (Panda and Nagwani, 2019, 2021).

1.1 Motivation

In order to illustrate the motivation behind the proposed ML-FBC approach, let us consider the following bug reports from the Mozilla software repositories (Mozilla, 2021).

Bug Report-1577416: HTML <video> outputs error message when playing WebM file created with patched ffmpeg to encode variable resolution.

According to Nagwani and Verma (2014), the above bug report belongs to multiple categories. The terms such as error, and file belong to the logical category, the terms such as HTML, message, and resolution belong to the GUI category, variable term belongs to the data type category, and the term patch belongs to the build category. In this scenario, the above bug report belongs to four different categories: logical, GUI, data types, and build. Hence, a binary classification algorithm is insufficient to handle the multiple causes related to various categories. Furthermore, the belonging of software bugs towards various categories can be computed using fuzzy logic.

Bug Report-1472380: Assertion failure: false (MOZ_ASSERT_UNREACHABLE:unexpected CSS unit for border image area division), at src/layout/painting/ns CSSRenderingBorders.cpp:3919

Similarly, in the above bug report the terms such as assertion and fail belong to the logical category, whereas the terms like border, CSS, image, layout, and render belong to the GUI category and the term unit belongs to the analysis category. Thus, the above bug report belongs to three different categories (logical, GUI, and analysis) at a time.

The above observation highlights that a software bug may belong to more than one category simultaneously. It is very difficult to understand the multiple causes of the software bug by classifying it using binary classification. Furthermore, most of the existing multi-label classification algorithms use hard clustering, but in practice, a software bug may affect multiple modules. Hence, these hard clustering algorithms are also not sufficient to handle these kinds of software bugs. To handle these kinds of software bugs, the fuzzy c-means clustering algorithm can be applied efficiently to the software bugs that belong to more than one category simultaneously.

In this paper, to address the aforementioned problem, a fuzzy system-based multi-label classification model for software bugs is designed. Initially, the ML-FBC classifier is designed for training data. The ML-FBC classifier is used to compute the fuzzy membership values of each software bug towards multiple categories for training data. The fuzzy relevance of software bugs to different categories is computed using the term-category relationship and the term-bug relationship. Fuzzy c-means clustering is adopted to group the fuzzy relevance of training data into a collection of sub regions to from clusters. Then, the cluster-category mapping is generated to compute the membership of each cluster towards multiple categories of software bugs. When a new bug is reported, its membership with the existing categories of training data is computed by using the different terms present in the new bug. Finally, the new bug is mapped into multiple categories with the help of cluster-category mapping of training data. A category threshold value is used to classify the new bug into multiple categories.

The main contributions to this article are summarised as follows:

1. A fuzzy system-based multi-label classification model is designed to compute the fuzzy membership of software bugs into multiple categories.
2. A fuzzy clustering algorithm is adopted to group the fuzzy relevance of software bugs into various clusters.
3. A user-defined threshold value is used to classify a newly reported bug into various categories based on its fuzzy similarity values to multiple categories.

The rest of the paper is structured as follows: in Section 2, the proposed model ML-FBC is discussed. In Section 3, an illustrative example of ML-FBC is presented with real-world software bugs from the MySql bug repository. The experimental outcome of different multi-label classifiers is presented in Section 4. Some threats to the validity of ML-FBC are discussed in Section 5 and finally, the conclusion and future direction of research are presented in Section 6.

2 Proposed methodology

In this section, the proposed fuzzy modelling for multi-label classification of software bugs is presented. At first, the summary field of software bug repositories is extracted. Then preprocessing is performed to obtain the processed software bug data for further operation. The processed data is divided into training and testing data. In the training phase, the training data is used to create a multi-label classification model. Finally, the classification model is then used to classify newly reported software bugs in the testing phase. The overall working principle of the proposed fuzzy model for multi-label classification is shown in Figure 2.

2.1 *Mulit-label classification of software bugs*

In a multi-label classification or categorisation of software bugs, a bug can belong to more than one category at a time. A multi-label bug classification is consisting of a triplet (B, T, C). B represents the set of n software bugs,

$$B = \{(b^{(1)}, l^{(1)}), (b^{(2)}, l^{(2)}), ..., (b^{(n)}, l^{(n)})\} \tag{1}$$

$T = \{t_1, t_2, ..., t_m\}$ represents the set of m software bug features or terms, and $C = \{c_1, c_2, ..., c_k\}$ represents the set of k software bug categories. A software bug $b^{(i)}$, $1 \leq i \leq n$ is denoted as a vector $\langle \pi_1{}^{(i)}, \pi_2{}^{(i)}, ..., \pi_m{}^{(i)} \rangle$, where $\pi_j{}^{(i)}$ denote the positive value such as TF-IDF of term t_j that occurs in software bug $b^{(i)}$.

$$l_j{}^{(i)} = \begin{cases} 1, & \text{if } b^{(i)} \text{ belongs to category } c_j \\ 0, & \text{if } b^{(i)} \text{ does not belongs to category } c_j \end{cases} \tag{2}$$

In multi-label classification, a software bug $b^{(i)}$ belongs to more than one category c_j. For example, $l_j{}^{(i)} = \{1, 0, 1\}$ indicates $b^{(i)}$ belongs categories c_1 and c_3 simultaneously. For classifying a new software bug based on the triplet (B, T, C), a dataset of B_1, B_2, ..., B_k will be created such that

$$\begin{cases} (b^{(i)}, 1) \in B_j, & \text{if } l_j{}^{(i)} = 1 \\ (b^{(i)}, 0) \in B_j, & \text{if } l_j{}^{(i)} = 0 \end{cases} \tag{3}$$

The summary of key symbols used in this article are illustrated in Table 1.

Figure 2 Block biagram of proposed fuzzy model (see online version for colours)

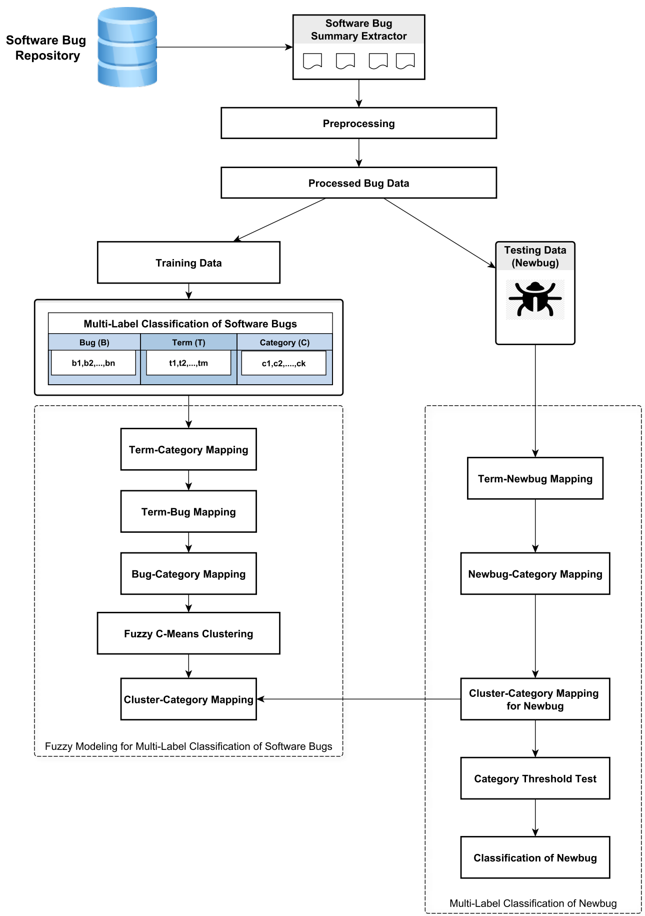


Table 1 List of symbols

Symbol	*Description*
B	Set of software bugs, $B = \{(b^{(1)}, l^{(1)}), (b^{(2)}, l^{(2)}), ..., (b^{(n)}, l^{(n)})\}$
$b^{(i)}$	The i^{th} software bug
$l^{(i)}$	Label of the i^{th} software bug $b^{(i)} \in B$ to category $c_j \in C$
T	Set of software bug features or terms, $T = \{t_1, t_2, ..., t_m\}$
C	Set of software bug categories, $C = \{c_1, c_2, ..., c_k\}$
${\pi_j}^{(i)}$	The positive TF-IDF value of term t_j that occurs in software bug $b^{(i)}$
$\mu Z_1(t_i, c_j)$	The fuzzy membership value of term $t_i \in T$ belongs to category $c_j \in C$
$\mu Z_2(t_i, b)$	The fuzzy membership value of term $t_i \in T$ belongs to software bug $b \in B$
c$\mu Z_3(b, c_j)$	The fuzzy membership value of software bug $b \in B$ belongs to category $c_j \in C$
$\otimes$	Fuzzy t-norm, i.e., $a \otimes b = a \times b$
$\oplus$	Fuzzy t-conorm, i.e., $a \oplus b = (a + b) - (a \times b)$
$F^{(i)}$	Set of fuzzy relevance vector
Y	Number of clusters for fuzzy relevance vector F
$F^{(\mu)}$	Mean of each cluster
$F^{(\sigma)}$	Standard deviation of each cluster
Y_{sim}	Cluster similarity
W	Weight matrix
C_{sim}	Category similarity
α	User defined threshold value

2.2 *Fuzzy modelling for multi-label classification of software bugs*

The proposed fuzzy model is designed to compute the fuzzy relevance of software bugs in B to different categories in C. When a new bug is reported, it is classified into multiple categories based on the computed fuzzy relevance between B and C. The details of the proposed work are described below.

Initially, the fuzzy relation between T and C is calculated, and it is represented as Z_1. The fuzzy membership of Z_1 is represented by $\mu Z_1(t_i, c_j)$ is the degree of belonging of term t_i to category c_j. The fuzzy membership value of $\mu Z_1(t_i, c_j)$ is determined as following:

$$\mu Z_1(t_i, c_j) = \frac{\sum_{v=1}^{n} \pi_i^{(v)} l_j^{(v)}}{\sum_{v=1}^{n} \pi_i^{(v)}} \frac{\sum_{v=1}^{n} S(\pi_i^{(v)}) l_j^{(v)}}{\sum_{v=1}^{n} l_j^{(v)}} \tag{4}$$

for $1 \leq i \leq m$ and $1 \leq j \leq k$, where

$$S(u) = \begin{cases} 1, & \text{if } u > 0 \\ 0, & \text{if } u = 0 \end{cases} \tag{5}$$

Let Z_2 be the fuzzy relation between term t_j to software bug b. The degree of fuzzy membership is denoted as $\mu Z_2(t_i, b)$ is the degree of belonging of term t_j in software bug b. The fuzzy membership value of $\mu Z_2(t_i, b)$ is determined as following:

$$\mu Z_2(t_i, b) = \frac{\pi_i}{\max_{1 \leq v \leq m} \pi_v} \tag{6}$$

for $1 \leq i \leq m$. The larger the value of π_i, the more the term t_i is relevant to software bug b.

The relevance between software bug b to the category c_j is denoted Z_3 and the degree fuzzy membership value $\mu Z_3(b, c_j)$is defined as follows:

$$\mu Z_3(b, c_j) = \frac{\sum_{i=1}^{m} \mu Z_1(t_i, c_j) \otimes \mu Z_2(t_i, b)}{\sum_{i=1}^{m} \mu Z_1(t_i, c_j) \oplus \mu Z_2(t_i, b)} \tag{7}$$

for $1 \leq j \leq k$. Where $\otimes$ is the fuzzy t-norm and $\oplus$ is the fuzzy t-conorm. The equation (7) provides the fuzzy mapping of each software bugs $b \in B$ to various categories $c_j \in C$ in the form of fuzzy relevance vector $F^{(i)} = \{F^{(1)}, F^{(2)}, ..., F^{(n)}\}$.

On the computed fuzzy relevance vector $F^{(i)}$, The fuzzy c-means clustering technique is applied to generate Y number of clusters and the mean $F^{(\mu)}$ and standard deviation $F^{(\sigma)}$ of each cluster are computed. The cluster similarity Y_{sim} is determined as following:

$$Y_{sim} = -\left[\left(\sum_{h=1}^{k} \frac{F_h^{(i)} - F_{h,y}^{(\mu)}}{F_{h,y}^{(\sigma)}} \right)^2 \right] \tag{8}$$

for $1 \leq y \leq Y$. In the next step, the category similarity is computed by mapping each cluster to an individual category c_j. In order to that, a linear model is created using the known value of Y_{sim} and C along with the unknown weight matrix W and it is defined as:

$$Y_{sim} W = C \tag{9}$$

Now, using the least square method on equation (9) the value of W is computed. The category similarity C_{sim} is computed by multiplying W with Y_{sim}. Once the C_{sim} is obtained for a software bug $b^{(i)}$, a user-defined threshold value α will be applied on $b^{(i)}$ to determine the corresponding multiple categories c_j.

$$c_j = \begin{cases} 1, & \text{if } Csim^j \geq \alpha \\ 0, & \text{Otherwise} \end{cases} \tag{10}$$

The overall approach is divided into two phases: the training phase and the testing phase. In the training phase, the training data is used to design the multi-label classification model. Initially, the training data is represented as triplets (B, T, C). Then the fuzzy relationship between term and category is calculated using equation (4) and the fuzzy relationship between term and software bug is calculated using equation (6). The fuzzy relationship between a software bug and category is computed using equation (7) and it will provide a fuzzy relevance vector $F^{(i)}$. This computed fuzzy relevance vector $F^{(i)}$ will provide the much needed information about the membership of each software bug towards different categories. Then clusters are generated by using the value of $F^{(i)}$, and the cluster mean and standard deviations are calculated. The cluster similarity values are computed using equation (8). Finally, the weight matrix W is computed for cluster-category mapping using the least square method.

Algorithm 1 Multi-label classification model for training phase

input : B, T and C for training data
output: Weight matrix W
begin
  Load B, T and C of training data
  //term-category mapping
  for $i \leftarrow 1$ **to** n **do**
    for $j \leftarrow 1$ **to** k **do**
      Compute $\mu Z_1(t_i, c_j)$ using equation (4)
    end
  end
  //term-bug mapping
  for $i \leftarrow 1$ **to** n **do**
    Compute $\mu Z_2(t_i, b)$ using equation (6)
  end
  //bug-category mapping
  for $i \leftarrow 1$ **to** n **do**
    for $j \leftarrow 1$ **to** k **do**
      Compute $\mu Z_3(b, c_j)$ or $F^{(i)}$ using equation (7)
      Create clusters using fuzzy c-means clustering on $F^{(i)}$
      Compute $F^{(\mu)}$ and $F^{(\sigma)}$ for each cluster
    end
  end
  Compute Y_{sim} using equation (8)
  Compute W using least square method
end

Algorithm 2 Multi-label classification for newly reported bug in testing phase

input : Newly reported bug nb, $F^{(\mu)}$, $F^{(\sigma)}$ and W from training phase
output: Multi-label classification for nb
begin
  Load nb, and W
  //term-new bug mapping
  for $i \leftarrow 1$ **to** n **do**
    Compute $\mu Z_2(t_i, nb)$ using equation (6)
  end
  //new bug-category mapping
  for $i \leftarrow 1$ **to** n **do**
    for $j \leftarrow 1$ **to** k **do**
      Compute $\mu Z_3(nb, c_j)$ using equation (7)
    end
  end
  //classification of new bug
  Compute Y_{sim} for nb using $F^{(\mu)}$, $F^{(\sigma)}$ in equation (8)
  Compute C_{sim} by multiplying W with Y_{sim}
  Apply equation (10) on C_{sim} to classify nb
end

In the testing phase, a new bug is considered for further operations. For the new bug, the fuzzy relationship between the term and the new bug is computed using equation (6). In order to calculate the fuzzy relevance vector for a new bug, the computed values from equation (6) are used in equation (7). Once the fuzzy relevance vector for a new bug is computed, the mean and standard deviation of the training phase are used to map the new bug to different clusters. After calculating the cluster similarity for the new bug, the computed weight matrix W in the training phase will be used to map the new bug to different categories. Finally, a threshold value α will be used to classify the new bug into multi-label categories using equation (10). The mechanism of the training phase is shown in Algorithm 1, and the mechanism of the testing phase is shown in Algorithm 2.

3 Illustrative example

In order to demonstrate the fuzzy approach-based multi-label classification of software bugs, an illustrative example is presented using nine real world software bugs from the MySql bug repository (Singh et al., 2020). The id and summary of the selected MySql bugs are shown in Table 2.

Table 2 Nine real software bugs available in MySql bug repository

Software bug (B)	*Id*	*Summary*
$b^{(1)}$	10039	Memory engine is reported as HEAP
$b^{(2)}$	10194	Scheduled backup causes memory access violation
$b^{(3)}$	10704	Memory information scheme table inaccessible if resident of memory is full
$b^{(4)}$	12184	Access violation
$b^{(5)}$	12578	Linked 5.0.11-views fail with Access 97 SR2
$b^{(6)}$	12906	Scheduled backup gails with exception EAccessViolation message
$b^{(7)}$	13307	Access violation Error 1.1.14
$b^{(8)}$	13412	Access violation in module libmysqlx.dll. Read of address 0000000
$b^{(9)}$	1358	Access control

Eleven terms [access, columns, data, DB, heap, memory, privileges, query, stored, table, user] related to three categories, namely, backend (l_1), memory (l_2) and security (l_3) are considered for the illustrative example. Based on the bag-of-words representation of software bugs using the 11 terms and three categories, is presented in Table 3. In Table 3, selected nine records with 11 features of software bugs π: $\{\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6, \pi_7, \pi_8, \pi_9, \pi_{10}, \pi_{11}\}$, three categories l: $\{l_1, l_2, l_3\}$ training data is taken as an example.

At first, the fuzzy relation between T and C is calculated using equation (4) and the value of $\mu Z_1(t_i, c_j)$ is

$$\mu Z_1(t_i, c_j) = \begin{bmatrix} 0.57 & 0.05 & 1.00 \\ 0.14 & 0.00 & 0.14 \\ 0.14 & 0.50 & 0.00 \\ 0.14 & 0.00 & 0.14 \\ 0.14 & 0.00 & 0.00 \\ 0.20 & 0.71 & 0.04 \\ 0.07 & 0.00 & 0.29 \\ 0.29 & 0.00 & 0.29 \\ 0.14 & 0.00 & 0.14 \\ 0.29 & 0.33 & 0.05 \\ 0.14 & 0.00 & 0.14 \end{bmatrix}$$

The $\mu Z_1(t_i, c_j)$ is the term category mapping and the individual values provide the membership of each bug term towards different categories, i.e., for term t_1, its membership to category c_1 is 0.57, c_2 is 0.05, and c_3 is 1.00.

In the next step, the fuzzy relationship between T and B is calculated using equation (6) and the computed values are represented by $\mu Z_2(t_i, b)$.

$$\mu Z_2(t_i, b) = \begin{bmatrix} 0.00 & 0.00 & 0.00 & 0.00 & 0.50 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.67 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.25 & 0.00 & 0.00 & 0.75 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.33 & 0.00 & 0.00 & 0.00 \\ 1.00 & 0.17 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.33 & 0.00 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.33 & 0.00 & 0.00 & 0.00 & 0.00 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.33 & 0.00 & 0.00 & 0.00 & 0.00 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.33 & 0.00 & 0.33 & 0.00 & 0.00 \\ 1.00 & 0.00 & 0.00 & 0.33 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.67 \end{bmatrix}$$

The $\mu Z_2(t_i, b)$ provides the membership values of individual bug terms towards different software bugs.

The fuzzy relevance between B and C is computed using equation (7) and the final fuzzy relevance vector $F^{(i)}$ is

$$F^{(i)} = \begin{bmatrix} 0.08 & 0.30 & 0.01 \\ 0.22 & 0.19 & 0.36 \\ 0.12 & 0.38 & 0.02 \\ 0.23 & 0.02 & 0.44 \\ 0.22 & 0.05 & 0.39 \\ 0.20 & 0.02 & 0.44 \\ 0.23 & 0.02 & 0.44 \\ 0.19 & 0.02 & 0.41 \\ 0.20 & 0.01 & 0.37 \end{bmatrix}$$

The $F^{(i)}$ provides the bug category mapping values, i.e., the membership values for bug b_1 to category c_1 is 0.08, to category c_2 is 0.30, and to category c_3 is 0.01.

Using the value of $F^{(i)}$ two clusters are formed for the illustrative example, and the cluster mean and standard deviation are computed. The cluster similarity, Y_{sim} is computed using equation (8), and the calculated value of Y_{sim} is

$$Y_{sim} = \begin{bmatrix} -1.63 & -222.02 \\ -4,725.13 & -6.91 \\ -1.63 & -176.38 \\ -7,407.19 & -2.10 \\ -5,543.62 & -1.02 \\ -7,401.31 & -2.25 \\ -7,407.19 & -2.10 \\ -6,439.34 & -1.90 \\ -5,081.79 & -2.10 \end{bmatrix}$$

Finally, for the training data, the value of W is computed using equation (9) with the help of the least square method. The computed value of W is

$$W = \begin{bmatrix} -0.00011 & -0.01059 & -0.01040 \\ -0.00491 & -0.11233 & -0.01209 \end{bmatrix}$$

Now let us consider three unseen software bugs (testing data)

$$b^{(1)}_{new} =< 3, 0, 0, 1, 0, 0, 0, 0, 1, 0, 3 >,$$
$$b^{(2)}_{new} =< 3, 0, 0, 0, 1, 2, 0, 0, 0, 0, 0 >,$$
$$b^{(3)}_{new} =< 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 > .$$

The fuzzy relationship between the term and the new bug mapping is calculated using equation (6) and the computed values are represented by $\mu Z_2(t_i, nb)$.

$$\mu Z_2(t_i, nb) = \begin{bmatrix} 1.00 & 0.00 & 0.00 & 0.33 & 0.00 & 0.00 & 0.00 & 0.00 & 0.33 & 0.00 & 1.00 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.33 & 0.67 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \end{bmatrix}$$

In the next step, the new bugs are mapped into different categories and the computed value of $\mu Z_3(nb, c_j)$ is

$$\mu Z_3(nb, c_j) = \begin{bmatrix} 0.20 & 0.01 & 0.34 \\ 0.21 & 0.17 & 0.32 \\ 0.21 & 0.15 & 0.20 \end{bmatrix}$$

Now the new bugs are mapped to the existing clusters of the training dataset and the cluster similarity value for the new bugs is calculated by using equation (8) with the cluster mean and standard deviation of the training data. The computed value of Y_{sim} is

$$Y_{sim} = \begin{bmatrix} -4,213.45 & -5.44 \\ -3,765.55 & -9.19 \\ -1,435.30 & -33.39 \end{bmatrix}$$

Finally, the category similarity values C_{sim} are calculated by multiplying W of training data with Y_{sim} of testing data. The computed value of C_{sim} is

$$C_{sim} = \begin{bmatrix} 0.490 & 0.078 & 0.650 \\ 0.079 & 0.459 & 0.581 \\ 0.322 & 0.097 & 0.222 \end{bmatrix}$$

Table 3 The bag-of-words representation of MySql bugs with 11 terms and three categories

Software bug (B)	*Positive TF-IDF values for term (T)*											*Category (C)*		
	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}	π_{11}	l_1	l_2	l_3
$b^{(1)}$	0	0	0	0	1	2	0	0	0	0	0	1	0	0
$b^{(2)}$	3	0	0	0	0	2	0	0	0	0	0	0	1	1
$b^{(3)}$	0	0	1	0	0	3	0	0	0	4	0	1	1	0
$b^{(4)}$	6	0	0	0	0	0	0	2	0	0	0	1	0	1
$b^{(5)}$	6	1	0	0	0	0	0	0	0	2	0	1	0	1
$b^{(6)}$	3	0	0	0	0	0	1	0	0	0	0	0	0	1
$b^{(7)}$	3	0	0	0	0	0	0	1	0	0	0	1	0	1
$b^{(8)}$	3	0	0	0	0	0	1	0	1	0	0	1	0	1
$b^{(9)}$	6	0	0	2	0	0	0	0	0	0	4	1	0	1

Now let us consider the value of α = 0.2, the different categories for new bugs using equation (10) are

$$C_j = \begin{bmatrix} 1\,0\,1 \\ 0\,1\,1 \\ 1\,0\,1 \end{bmatrix}$$

Generally the similar software bugs are belonging to the similar categories, in order to show this let us consider two similar bugs from the illustrative examples $b^{(2)}$ = <3, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0> and $b^{(2)}_{new}$ = <3, 0, 0, 0, 1, 2, 0, 0, 0, 0, 0> for this two bugs cosine similarity is calculated as shown in equation (11)

$$\begin{aligned}\cos(b^{(2)}, b^{(2)}_{new}) &= \frac{3 \times 3 + 0 \times 1 + 2 \times 2}{\sqrt{(3^2 + 2^2)} \times \sqrt{(3^2 + 1^2 + 2^2)}} \\ &= \frac{13}{\sqrt{13} \times \sqrt{14}} = 0.963 \end{aligned} \tag{11}$$

Both of the bugs are 96.3% similar to each other. In the example training dataset, the category of $b^{(2)}$ is <0, 1, 1> and the category identified by the fuzzy-based approach for the bug $b^{(2)}_{new}$ is <0.079, 0.459, 0.581>, i.e., <0, 1, 1> which also indicates that the category of new bug is dominating the second and third category. Hence, it also shows that similar bugs belong to the same categories.

4 Experimental results

According to Hooimeijer and Weimer (2007) and Antoniol et al. (2008) the important surface features for bug classification and mining (knowledge acquisition) are title (summary) and description only, whereas the comment part can be ignored. This is because there is more relevance between the title and summary of a bug, whereas comments consist of more random information for software bugs.

In order to compare the significance of the proposed fuzzy-based classification technique with the other techniques, different performance measures are used, such as micro averaged precision (MicroP), micro averaged recall (MicroR), micro averaged F1 (F1), micro averaged break-even point (BEP), hamming loss (Hloss) and accuracy. The number of categories is denoted by k and the number of software bugs is denoted by n. The different performance measures are defined as follows:

$$MicroP = \frac{\sum_{l=1}^{k} TP_l}{\sum_{l=1}^{k} TP_l + FP_l} \tag{12}$$

$$MicroR = \frac{\sum_{l=1}^{k} TP_l}{\sum_{l=1}^{k} TP_l + FN_l} \tag{13}$$

$$F_1 = \frac{2 \times MicroP \times MicroR}{MicroP + MicroR} \tag{14}$$

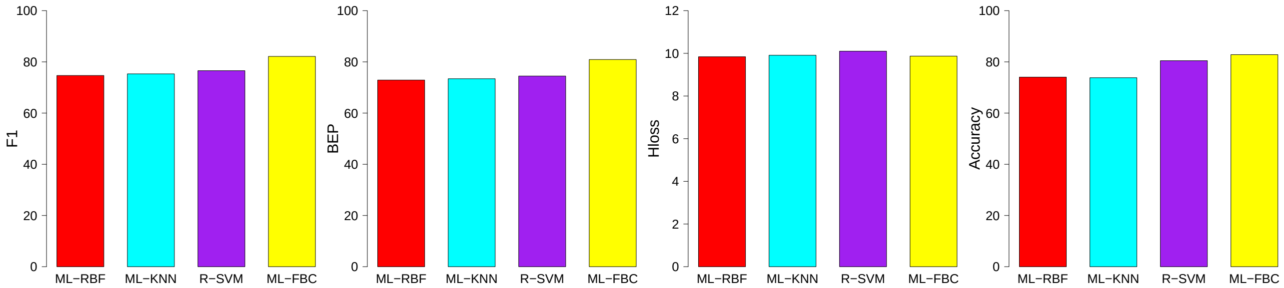
$$BEP = \frac{MicroP + MicroR}{2} \tag{15}$$

$$Hloss = \frac{\sum_{l=1}^{k} FP_l + FN_l}{k \times n} \tag{16}$$

$$Accuracy = \frac{\sum_{l=1}^{k} \frac{TP_l + TN_l}{TP_l + FP_l + TN_l + FN_l}}{k} \tag{17}$$

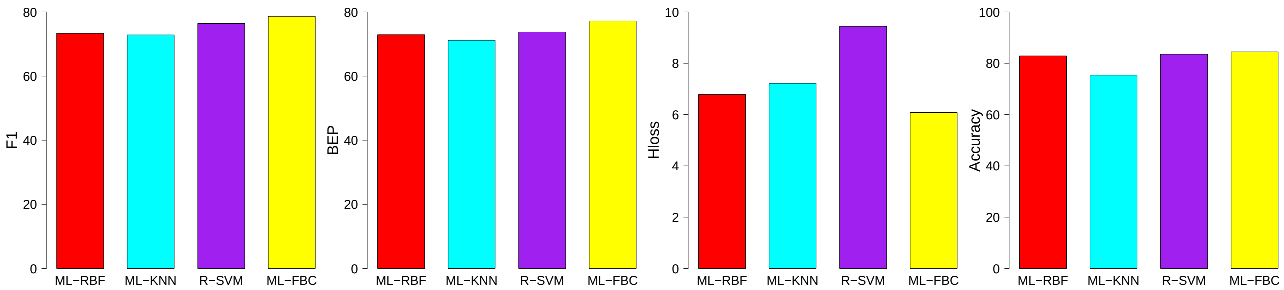
where TP_l represents the true positive rate with respect to category k, is the number of software bugs that have a positive value and are correctly classified as positive by the system. Similarly, TN_l represents the true negative rate with respect to category k, is the number of software bugs that have negative values, and the system is also classified as negative. FP_l is the false positive rate with respect to category k is the number of software bugs that have a negative value and the system is classified as positive. Similarly, FN_l is the false negative rate with respect to category k is the number of software bugs that have positive value and the system classified as negative. In the classification results, the performance of a system is considered better when the values of $F1$, BEP are larger and the values of $Hloss$ are smaller.

Figure 3 Comparison of various classifier on Eclipse dataset with different parameters, (a) F1 (b) BEP (c) Hloss (d) accuracy (see online version for colours)



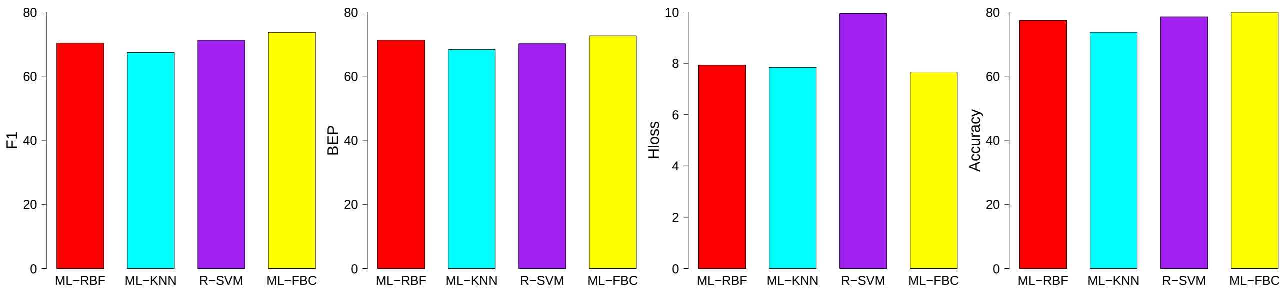
(a) (b) (c) (d)

Figure 4 Comparison of various classifier on Mozilla dataset with different parameters, (a) F1 (b) BEP (c) Hloss (d) accuracy (see online version for colours)



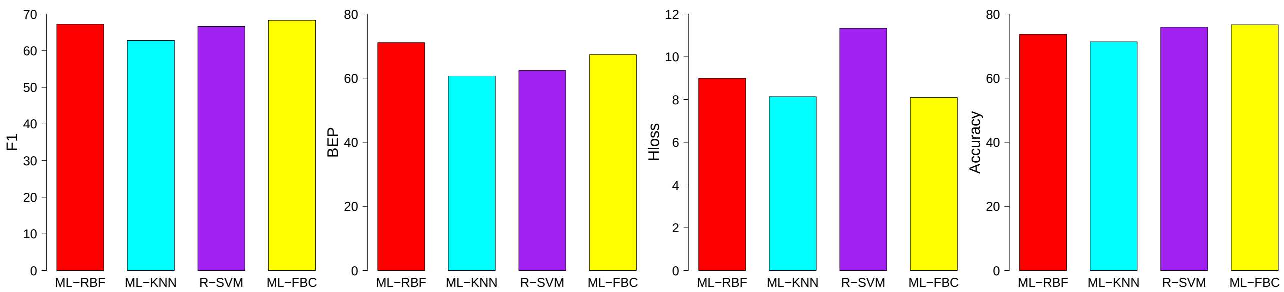
(a) (b) (c) (d)

Figure 5 Comparison of various classifier on MySql dataset with different parameters, (a) F1 (b) BEP (c) Hloss (d) accuracy (see online version for colours)



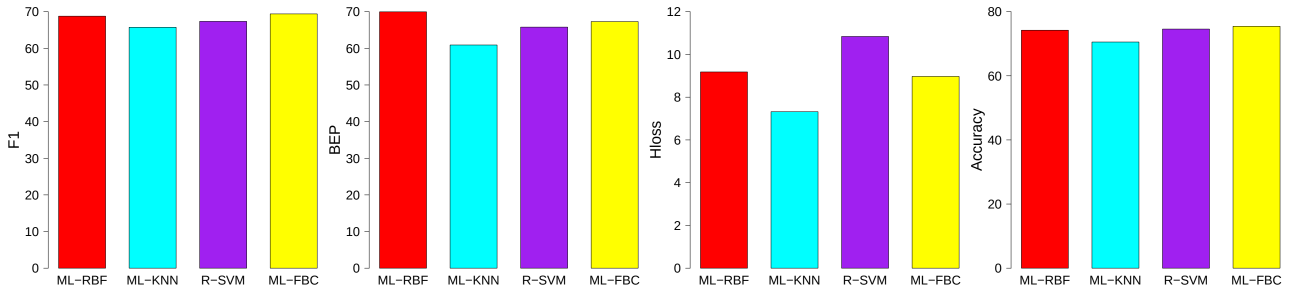
(a) (b) (c) (d)

Figure 6 Comparison of various classifier on Android dataset with different parameters, (a) F1 (b) BEP (c) Hloss (d) accuracy (see online version for colours)



(a) (b) (c) (d)

Figure 7 Comparison of various classifier on JBoss-Seam dataset with different parameters, (a) F1 (b) BEP (c) Hloss (d) accuracy (see online version for colours)



In order to compare the various classifiers, two statistical significance tests [the Friedman (1937, 1940) statistical test and the post-hoc Nemenyi (1963) test recommended by Demšar et al. (2006) are carried out on different datasets]. The Friedman statistical test is used to determine whether the classifiers are the same or different based on the null hypothesis (H_0: all classifiers are the same) and alternative hypothesis (H_1: all classifiers are different).

According to Friedman (1937, 1940) the Friedman statistic χ_F^2 is computed as follows:

$$\chi_F^2 = \frac{12 \times N}{K(K+1)} \left[\sum_{j=1}^{k} R_j^2 - \frac{K(K+1)^2}{4} \right] \tag{18}$$

where N is the number of rows in the dataset, K is the number of classifiers, and R_j is average rank value of the classifier.

The post-hoc Nemenyi test is done to find the significance difference between individual classifiers. The critical difference cd of post-hoc (Nemenyi, 1963) is computed as follows:

$$cd = q_\alpha \sqrt{\frac{K(K+1)}{6N}} \tag{19}$$

where q_α is the studentised ranged statistic.

Five different datasets, namely the Eclipse (2021) dataset, the Mozilla (2021) dataset, the MySql (2021) dataset, the Android (2021) dataset and the JBoss-Seam (2021) dataset were experimented with R-programming (R-Software, 2021). To perform all these techniques, a PC with a 2.00 GHz Intel Core i3-6006 CPU and 4 GB of RAM is used. For the classification of software bugs, the different categories are considered, and they are broadly classified as ‘bug’ and ‘non-bug’ categories. The bug category is further subdivided into two subcategories: logical and backend. The logical category is again divided into two: the GUI category and the non-GUI category. The non-GUI category consists of software bugs that are related to memory, data types, operating systems, and security, etc. Similarly, the non-bug category is further sub divided into enhancement and build and analysis category (Nagwani and Verma, 2014).

Table 4 Performance of various classifier on different datasets

Datasets	*Parameter*	*Different classifier*			
		ML-RBF	*ML-KNN*	*R-SVM*	*ML-FBC*
Eclipse	F1 (%)	74.68	75.32	76.56	*82.13*
	BEP (%)	72.90	73.41	74.47	*80.91*
	Hloss (%)	*9.84*	9.91	10.10	9.87
	Accuracy (%)	74.04	73.81	80.46	*82.82*
	Training time (sec)	272.88	283.19	134.32	*102.14*
	Testing time (sec)	12.50	71.12	24.35	*3.33*
Mozilla	F1 (%)	73.33	72.85	76.40	*78.63*
	BEP (%)	72.91	71.17	73.75	*77.18*
	Hloss (%)	6.78	7.22	9.44	*6.08*
	Accuracy (%)	82.88	75.39	83.53	*84.44*
	Training time (sec)	282.42	332.41	168.97	*142.43*
	Testing time (sec)	5.50	63.52	4.87	*2.20*
MySql	F1 (%)	70.34	67.38	71.20	*73.65*
	BEP (%)	71.26	68.28	70.12	*72.57*
	Hloss (%)	7.93	7.84	9.94	*7.66*
	Accuracy (%)	77.38	73.68	78.49	*79.94*
	Training time (sec)	235.68	342.67	132.48	*97.61*
	Testing time (sec)	4.50	52.51	24.42	*2.10*
Android	F1 (%)	67.22	62.77	66.59	*68.28*
	BEP (%)	*71.06*	60.67	62.35	67.32
	Hloss (%)	8.99	8.13	11.33	*8.09*
	Accuracy (%)	73.67	71.34	75.90	*76.64*
	Training time (sec)	209.99	254.87	110.05	*95.08*
	Testing time (sec)	2.72	54.42	20.46	*1.52*
JBoss-Seam	F1 (%)	68.78	65.74	67.35	*69.39*
	BEP (%)	*69.96*	60.91	65.79	67.30
	Hloss (%)	9.18	*7.32*	10.84	8.97
	Accuracy (%)	74.22	70.54	74.58	*75.45*
	Training time (sec)	198.71	273.64	132.47	*110.36*
	Testing time (sec)	3.10	63.58	12.51	*1.82*

The hyperparameters for the various classifiers are selected based on the best results obtained for each classifier. For ML-RBF, the scaling factor value is set to 1.0, and the fraction parameter value is set to 0.01. In the case of ML-KNN, a model with k values of 3, 6, 9, 12, 15, 18, and 21 is used for conducting the experiments. For ML-KNN, the best result is obtained for k = 12 and the hyperparameter k is set to 12. In rank-SVM, the linear kernel provides a better result as compared to polynomial kernels with degree 8 for all the datasets. Finally, two-parameter values for the proposed technique ML-FBC are fixed based on the best findings. The user-defined threshold value α is set to 0.5, and the number of clusters in fuzzy c-means clustering is set to 10. A ten-fold cross-validation is performed for each classifier. The average results of each fold are calculated and taken as the final result.

Table 5 Friedman test statistic and average rank of different classifier on various datasets

Datasets	*ML-RBF*	*ML-KNN*	*R-SVM*	*ML-FBC*	*Friedman test statistic* χ^2_F
Eclipse	2.83	3.50	2.50	1.17	10.36
Mozilla	2.83	3.83	2.33	1.00	14.78
Mysql	2.67	3.67	2.67	1.00	13.42
Android	2.33	3.67	2.83	1.17	11.79
JBoss-Seam	2.33	3.50	2.83	1.33	8.84

The results of the proposed work (ML-FBC) are compared with the results of ML-RBF, ML-KNN, and R-SVM. The performance of different classifiers with various parameters on different datasets is shown in Table 4. The best results for each dataset are highlighted in ital letters. For the Eclipse dataset, the proposed ML-FBC obtained 82.13% F1 score, 80.91% BEP score, and 82.82% accuracy with a training time of 102.14 seconds and a testing time of 3.33 seconds. Whereas the ML-RBF has the lowest Hloss value of 9.84% as compared to other classifiers. In terms of testing and training time, the proposed ML-FBC classifier runs much faster than the other classifiers. A comparison graph of various classifiers on the Eclipse dataset with different parameters is plotted and is shown in Figure 3. For the Mozilla bug dataset, the best results are obtained by using the ML-FBC classifier and the values are 78.63% as F1 score, 77.18% as BEP score, 6.08% as Hloss and accuracy of 84.44%. The training and testing times for the Mozilla dataset using the ML-FBC classifier were 142.43 seconds and 2.20 seconds, which is significantly faster than the other classifiers. A comparison graph between various classifiers with different parameters is plotted and shown in Figure 4.

When the classifiers are tested on MySql datasets, the best F1 score, BEP score, Hloss, and Accuracy scores are 73.65%, 72.57%, 7.66% and 79.94% respectively, and it is obtained for the ML-FBC classifier. Whereas the lowest F1 score, BEP score, and accuracy are 67.38%, 68.28% and 73.68% respectively for the ML-KNN classifier. Similarly, the lowest Hloss is 9.94% and it is obtained for the R-SVM classifier. The ML-FBC classifier took less training and testing time as compared to other classifiers, and the measured training time is 97.61 seconds and the testing time is 2.10 seconds, respectively. A comparison graph among various classifiers is plotted and shown in Figure 5. For the android dataset, the ML-FBC outperforms other classifiers. The best F1 score, Hloss, and accuracy scores are 68.28%, 8.09% and 76.64% respectively, and they are achieved using the ML-FBC classifier. Whereas the best BEP score 71.08% is obtained by the ML-RBF classifier. The training and testing time for the ML-FBC classifier is 95.08 seconds and 1.52 seconds respectively, and the ML-FBC classifier runs faster than that of other classifiers. The comparison of various classifiers for the Android dataset is plotted and shown in Figure 6. Finally, the classifiers are tested using JBoss-Seam datasets. The highest F1 score is 69.39% and it is achieved using the ML-FBC classifier, whereas the ML-RBF classifier provides the highest BEP score of 69.96%. The best Hloss score is 7.32% and it is obtained by using the ML-KNN classifier. The highest accuracy is 75.45% and it is achieved by using the ML-FBC classifier. The training and testing time for ML-FBC is faster than that of other classifiers. The training time is 110.36 seconds and the testing time is 1.82 seconds when the ML-FBC classifier is used. A comparison graph among various classifiers is plotted and shown in Figure 7.

In a multi-label classification algorithm, the accuracy of any model is low as it deals with multiple categories, and the accuracy is calculated based on the number of categories present in the model. If the number of categories increases, the accuracy of the model decreases. As a result, accuracy is not considered the most appropriate measure for evaluating the performance of a multi-label classification model. The performance of a multi-label classification algorithm is better if the value of $F1$, BEP score is larger, and $Hloss$ score is smaller (Lee and Jiang, 2013; Pereira et al., 2018; Zhang and Zhou, 2013). The same thing can be observed from the experimental results of the proposed ML-FBC, that the highest accuracy of 84.44% is obtained for the ML-FBC classifier, and it is obtained for the Mozilla dataset. For all the datasets, the accuracy of the ML-FBC classifier is significantly higher than the other existing multi-label classifiers. Furthermore, the performance of the ML-FBC classifier in terms $F1$, BEP, and $Hloss$ scores is far better than the other existing multi-label classifiers for the majority of datasets. Hence, the proposed ML-FBC model outperforms all other existing multi-label classification models in terms of performance, training, and testing time.

In order to calculate the average rank of the classifiers, the data of ten-fold cross-validation presented in Table 4 is considered. For each dataset, the value of N is 6, and the value of K is 4. For the parameters F1, BEP, and accuracy, the classifier with the highest value is allocated rank 1, and the classifier with the lowest value is assigned rank 4. Similarly, for Hloss, training time, and testing time, the classifier having the lowest value is assigned rank 1, and the classifier having the highest value is assigned rank 4. The average rank for each classifier is computed, and the Friedman statistic value is computed using equation (18). The result of the Friedman statistical test and the average rank of different classifiers on different datasets is shown in Table 5. Among all the classifiers, the proposed ML-FBC classifier has the best average ranking, whereas the ML-KNN classifier has the lowest average ranking.

The critical chi-square value for the Friedman test for α = 0.05 and degree of freedom = 3 (K – 1 = 3) is 7.815. For all the datasets, the value of the Friedman statistic is greater than the critical chi-square value. Hence, the null hypothesis (H_0: all classifiers are the same) is rejected. The post-hoc Nemenyi test is conducted to find the significance difference between individual classifiers. Furthermore, the cd value for the Nemenyi test is computed using equation (19) with the value of q_α = 2.569, K = 4, and N = 6. The computed cd value is 1.9148. Now, based on the Nemenyi, if the difference between two classifiers is greater than that of cd then they are different. Here, for all the datasets, ML-KNN and ML-FBC have a higher difference than cd. Hence, these two classifiers are significantly different from each other, and the ML-FBC is far better than the ML-KNN.

5 Threats to validity

This section discusses the threats to the validity of the proposed work ML-FBC. There can be several factors that can have a profound impact on the results of the proposed work. The proposed work is entirely automated and randomly generated. The selection of random samples for different datasets may vary from person to person and can result in some experimental bias. The categories that are generated are based on the developers' bug handling conventions and previous bug fixing processes. The selection of categorical discriminative terms is entirely up to the triager and the developers involved in the bug triaging process. It varies from developer to developer, as each developer utilises their own set of vocabulary to present the software bug information in a bug report. These are a few of the exceptional cases that might have an impact on the outcomes of the proposed work.

6 Conclusions and future work

In this paper, a fuzzy system-based approach is presented for improving the multi-label classification of software bugs. A multi-label classification approach for software bugs are developed using the discriminative terms appears in the software bugs. The efficiency of the proposed ML-FBC classifier is investigated using five different datasets: the Eclipse dataset, Mozilla dataset, MySql dataset, Android dataset and JBoss-Seam dataset. The different performance measures, F1 score, BEP score, Hloss, accuracy, training time, and testing time of different classifiers are compared. The experiments show that the ML-FBC classifier outperformed other classifiers on various performance measures. As a result, the fuzzy model is extremely useful for multi-label classification of software bugs and provides a better bug information management in large bug repositories. Furthermore, as fuzzy system is used, the training and testing time of ML-FBC is much faster as compared to other classifiers.

In the future, the relationship between developers and various categories can be investigated using the membership, non-membership, and hesitancy relations between software bugs and multiple categories. These relations will illuminate the uncharted area towards the use of advanced fuzzy systems such as intuitionistic fuzzy sets, interval fuzzy sets, pythagorean fuzzy sets, spherical fuzzy sets, etc. for modelling and developing better classification models for software bugs. These advanced fuzzy techniques will provide a better understanding of the developer category relationship and improve bug information management in large bug repositories.

References

Ahmed, A.A. and Darwish, S.M. (2021) 'A meta-heuristic automatic CNN architecture design approach based on ensemble learning', *IEEE Access*, Vol. 9, pp.16975–16987.

Ahmed, H.A., Bawany, N.Z. and Shamsi, J.A. (2021) 'CaPBug – a framework for automatic bug categorization and prioritization using NLP and machine learning algorithms', *IEEE Access*, Vol. 9, pp.50496–50512.

Al-Salemi, B., Ayob, M., Kendall, G. and Noah, S.A.M. (2019) 'Multi-label arabic text categorization: a benchmark and baseline comparison of multi-label learning algorithms', *Information Processing & Management*, Vol. 56, No. 1, pp.212–227.

Android (2021) *Android Bug Dataset* [online] https://code.google.com/p/android/issues (accessed 15 May 2021).

Antoniol, G., Ayari, K., Di Penta, M., Khomh, F. and Guéhéneuc, Y-G. (2008) '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*, pp.304–318.

Bahaddad, A.A., Ragab, M., Ashary, E.B. and Khalil, E.M. (2022) 'Metaheuristics with deep learning-enabled Parkinson's disease diagnosis and classification model', *Journal of Healthcare Engineering*.

Bei, H., Mao, Y., Wang, W. and Zhang, X. (2021) 'Fuzzy clustering method based on improved weighted distance', *Mathematical Problems in Engineering*.

Çolak, M., Kaya, İ., Karaşan, A. and Erdoğan, M. (2022) 'Two-phase multi-expert knowledge approach by using fuzzy clustering and rule-based system for technology evaluation of unmanned aerial vehicles', *Neural Computing and Applications*, pp.1–17.

Das, S., Datta, S., Zubaidi, H.A. and Obaid, I.A. (2021) 'Applying interpretable machine learning to classify tree and utility pole related crash injury types', *IATSS Research*.

Demšar, J. (2006) 'Statistical comparisons of classifiers over multiple data sets', *The Journal of Machine Learning Research*, Vol. 7, pp.1–30.

Dong, X., Huang, B. and Zhou, Y. (2022) 'Research on fast face retrieval optimization algorithm based on fuzzy clustering', *Scientific Programming*.

Eclipse (2021) *Eclipse Bug Dataset* [online] https://bugs.eclipse.org/bugs/ (accessed 15 June 2021)

Elisseeff, A. and Weston, J. (2001) 'A kernel method for multi-labelled classification', *Advances in Neural Information Processing Systems*, Vol. 14, pp.681–687.

Farahani, F.V., Ahmadi, A. and Zarandi, M.H.F. (2018) 'Hybrid intelligent approach for diagnosis of the lung nodule from CT images using spatial kernelized fuzzy c-means and ensemble learning', *Mathematics and Computers in Simulation*, Vol. 149, pp.48–68.

Fausto, F., Reyna-Orta, A., Cuevas, E., Andrade, Á.G. and Perez-Cisneros, M. (2020) 'From ants to whales: metaheuristics for all tastes', *Artificial Intelligence Review*, Vol. 53, No. 1, pp.753–810.

Friedman, M. (1937) 'The use of ranks to avoid the assumption of normality implicit in the analysis of variance', *Journal of the American Statistical Association*, Vol. 32, No. 200, pp.675–701.

Friedman, M. (1940) 'A comparison of alternative tests of significance for the problem of M rankings', *The Annals of Mathematical Statistics*, Vol. 11, No. 1, pp.86–92.

Gangavarapu, T., Jayasimha, A., Krishnan, G.S. and Kamath, S. (2020) 'Predicting ICD-9 code groups with fuzzy similarity based supervised multi-label classification of unstructured clinical nursing notes', *Knowledge-Based Systems*, Vol. 190, p.105321.

Gao, Y., Wang, Z., Xie, J. and Pan, J. (2022) 'A new robust fuzzy c-means clustering method based on adaptive elastic distance', *Knowledge-Based Systems*, Vol. 237, p.107769.

Ghosh, S., Samanta, G. and De la Sen, M. (2021) 'Multi-model approach and fuzzy clustering for mammogram tumor to improve accuracy', *Computation*, Vol. 9, No. 5, p.59.

Govindarajan, P., Soundarapandian, R.K., Gandomi, A.H., Patan, R., Jayaraman, P. and Manikandan, R. (2020) 'Classification of stroke disease using machine learning algorithms', *Neural Computing and Applications*, Vol. 32, No. 3, pp.817–828.

Herbold, S., Trautsch, A. and Trautsch, F. (2020) 'On the feasibility of automated prediction of bug and non-bug issues', *Empirical Software Engineering*, Vol. 25, No. 6, pp.5333–5369.

Hooimeijer, P. and Weimer, W. (2007) 'Modeling bug report quality', in *Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering*, pp.34–43.

JBoss-Seam (2021) *JBoss-Seam Dataset* [online] https://issues.jboss.org/browse/JBSEAM (accessed 12 July 2021).

Jiang, J-Y., Tsai, S-C. and Lee, S-J. (2012) 'FSKNN: multi-label text categorization based on fuzzy similarity and k nearest neighbors', *Expert Systems with Applications*, Vol. 39, No. 3, pp.2813–2821.

Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A.M. and Talbi, E-G. (2021) 'Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: a state-of-the-art', *European Journal of Operational Research*.

Kavitha, P.K. and Saraswathi, P.V. (2021) 'Content based satellite image retrieval system using fuzzy clustering', *Journal of Ambient Intelligence and Humanized Computing*, pp.1–12.

Lee, S-J. and Jiang, J-Y. (2013) 'Multilabel text categorization based on fuzzy relevance clustering', *IEEE Transactions on Fuzzy Systems*, Vol. 22, No. 6, pp.1457–1471.

Martínez-Cagigal, V., Santamaría-Vázquez, E. and Hornero, R. (2022) 'Brain-computer interface channel selection optimization using meta-heuristics and evolutionary algorithms', *Applied Soft Computing*, Vol. 115, p.108176.

Mei, M., Zhong, Y., He, F. and Xu, C. (2020) 'An innovative multi-label learning based algorithm for city data computing', *Geo Informatica*, Vol. 24, No. 1, pp.221–245.

Meng, Z., Li, G., Wang, X., Sait, S.M. and Yıldız, A.R. (2021) 'A comparative study of metaheuristic algorithms for reliability-based design optimization problems', *Archives of Computational Methods in Engineering*, Vol. 28, pp.1853–1869.

Mohsin, H. and Shi, C. (2021) 'SPBC: a self-paced learning model for bug classification from historical repositories of open-source software', *Expert Systems with Applications*, Vol. 167, p.113808.

Mozilla (2021) *Mozilla Bug Dataset* [online] https://bugzilla.mozilla.org/describecomponents.cgi (accessed 25 June 2021).

MySql (2021) *MySql Bug Dataset* [online] https://bugs.mysql.com (accessed 10 July 2021).

Nagwani, N.K. and Verma, S. (2014) 'A comparative study of bug classification algorithms', *International Journal of Software Engineering and Knowledge Engineering*, Vol. 24, No. 1, pp.111–138.

Naseem, U., Khushi, M., Khan, S.K., Shaukat, K. and Moni, M.A. (2021) 'A comparative analysis of active learning for biomedical text mining', *Applied System Innovation*, Vol. 4, No. 1, p.23.

Nemenyi, P.B. (1963) *Distribution-Free Multiple Comparisons*, Princeton University.

Ni, Z., Li, B., Sun, X., Chen, T., Tang, B. and Shi, X. (2020) 'Analyzing bug fix for automatic bug cause classification', *Journal of Systems and Software*, Vol. 163, p.110538.

Onan, A. (2021) 'Sentiment analysis on massive open online course evaluations: a text mining and deep learning approach', *Computer Applications in Engineering Education*, Vol. 29, No. 3, pp.572–589.

Panda, R.R. and Nagwani, N.K. (2019) 'Software bug categorization technique based on fuzzy similarity', in *2019 IEEE 9th International Conference on Advanced Computing (IACC)*, IEEE, pp.1–6.

Panda, R.R. and Nagwani, N.K. (2021) 'Multi-label software bug categorisation based on fuzzy similarity', *International Journal of Computational Science and Engineering*, Vol. 24, No. 3, pp.244–258.

Peng, L. and Liu, Y. (2018) 'Feature selection and overlapping clustering-based multilabel classification model', *Mathematical Problems in Engineering*.

Pereira, R.B., Plastino, A., Zadrozny, B. and Merschmann, L.H.C. (2018) 'Correlation analysis of performance measures for multi-label classification', *Information Processing & Management*, Vol. 54, No. 3, pp.359–369.

Qian, W., Huang, J., Wang, Y. and Xie, Y. (2021) 'Label distribution feature selection for multi-label classification with rough set', *International Journal of Approximate Reasoning*, Vol. 128, pp.32–55.

Qian, W., Xiong, C. and Wang, Y. (2021) 'A ranking-based feature selection for multi-label classification with fuzzy relative discernibility', *Applied Soft Computing*, Vol. 102, p.106995.

R-Software (2021) [online] https://www.r-project.org (accessed 10 June 2021).

Rubio, E., Castillo, O., Valdez, F., Melin, P., Gonzalez, C.I. and Martinez, G. (2017) 'An extension of the fuzzy possibilistic clustering algorithm using type-2 fuzzy logic techniques', *Advances in Fuzzy Systems*.

Saha, I., Sarkar, J.P. and Maulik, U. (2019) 'Integrated rough fuzzy clustering for categorical data analysis', *Fuzzy Sets and Systems*, Vol. 361, pp.1–32.

Sanchez, M.A., Castillo, O., Castro, J.R. and Melin, P. (2014) 'Fuzzy granular gravitational clustering algorithm for multivariate data', *Information Sciences*, Vol. 279, pp.498–511.

Santander-Jiménez, S., Vega-Rodríguez, M.A. and Sousa, L. (2022) 'Exploiting multi-level parallel metaheuristics and heterogeneous computing to boost phylogenetics', *Future Generation Computer Systems*, Vol. 127, pp.208–224.

Si, T., Bagchi, J. and Miranda, P.B.C. (2022) 'Artificial neural network training using metaheuristics for medical data classification: an experimental study', *Expert Systems with Applications*, p.116423.

Singh, L.K., Garg, H., Pooja and Khanna, M. (2020) 'Performance analysis of machine learning techniques for glaucoma detection based on textural and intensity features', *International Journal of Innovative Computing and Applications*, Vol. 11, No. 4, pp.216–230.

Soltani, M., Hermans, F. and Bäck, T. (2020) 'The significance of bug report elements', *Empirical Software Engineering*, Vol. 25, No. 6, pp.5255–5294.

Talbi, E-G. (2021) 'Machine learning into metaheuristics: a survey and taxonomy', *ACM Computing Surveys (CSUR)*, Vol. 54, No. 6, pp.1–32.

Tarekegn, A., Giacobini, M. and Michalak, K. (2021) 'A review of methods for imbalanced multi-label classification', *Pattern Recognition*, p.107965.

Thakur, V. and Patel, A.C. (2021) 'An improved dictionary based genre classification based on title and abstract of e-book using machine learning algorithms', in *Proceedings of Second International Conference on Computing, Communications, and Cyber-Security*, Springer, pp.323–337.

Tzanetos, A. and Dounias, G. (2021) 'Nature inspired optimization algorithms or simply variations of metaheuristics?', *Artificial Intelligence Review*, Vol. 54, No. 3, pp.1841–1862.

Wang, R., Kwong, S., Wang, X. and Jia, Y. (2021) 'Active k-labelsets ensemble for multi-label classification', *Pattern Recognition*, Vol. 109, p.107583.

Wang, S., Xiao, S., Zhu, W. and Guo, Y. (2022) 'Multi-view fuzzy clustering of deep random walk and sparse low-rank embedding', *Information Sciences*, Vol. 586, pp.224–238.

Wu, G., Zheng, R., Tian, Y. and Liu, D. (2020) 'Joint ranking svm and binary relevance with robust low-rank learning for multi-label classification', *Neural Networks*, Vol. 122, pp.24–39.

Xi, S-Q., Yao, Y., Xiao, X-S., Xu, F. and Lv, J. (2019) 'Bug triaging based on tossing sequence modeling', *Journal of Computer Science and Technology*, Vol. 34, No. 5, pp.942–956.

Xia, X., Lo, D., Ding, Y., Al-Kofahi, J.M., Nguyen, T.N. and Wang, X. (2016) 'Improving automated bug triaging with specialized topic model', *IEEE Transactions on Software Engineering*, Vol. 43, No. 3, pp.272–297.

Xia, Y., Chen, K. and Yang, Y. (2021) 'Multi-label classification with weighted classifier selection and stacked ensemble', *Information Sciences*, Vol. 557, pp.421–442.

Yadav, A., Singh, S.K. and Suri, J.S. (2019) 'Ranking of software developers based on expertise score for bug triaging', *Information and Software Technology*, Vol. 112, pp.1–17.

Zhang, D., Zhao, S., Duan, Z., Chen, J., Zhang, Y. and Tang, J. (2020) 'A multi-label classification method using a hierarchical and transparent representation for paper-reviewer recommendation', *ACM Transactions on Information Systems (TOIS)*, Vol. 38, No. 1, pp.1–20.

Zhang, M-L. (2009) 'ML-RBF: RBF neural networks for multi-label learning', *Neural Processing Letters*, Vol. 29, No. 2, pp.61–74.

Zhang, M-L. and Zhou, Z-H. (2007) 'ML-KNN: a lazy learning approach to multi-label learning', *Pattern Recognition*, Vol. 40, No. 7, pp.2038–2048.

Zhang, M-L. and Zhou, Z-H. (2013) 'A review on multi-label learning algorithms', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 26, No. 8, pp.1819–1837.

Zhang, Y., Wang, Y., Liu, X-Y., Mi, S. and Zhang, M-L. (2020) 'Large-scale multi-label classification using unknown streaming images', *Pattern Recognition*, Vol. 99, p.107100.

Zhou, C., Li, B., Sun, X. and Guo, H. (2018) 'Recognizing software bug-specific named entity in software bug repository', in *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)*, IEEE, pp.108–10811.