2017 IEEE 28th International Symposium on Software Reliability Engineering Workshops

The Automatic Classification of Fault Trigger Based Bug Report

Xiaoting Du, Zheng Zheng*, Guanping Xiao, Beibei Yin
School of Automation Science and Electrical Engineering, Beihang University, Beijing, China
Email: {xiaoting_2015, zhengz, gpxiao, yinbeibei}@buaa.edu.cn

***Abstract*—Understanding the types of defects is of practical interest, which could help developers adopt proper measures in current and future software releases. As the amount of bug reports increasing, manual classification brings a heavy burden to developers. In this paper, we propose a word2vec based framework of multi-granularity automatic classification for bug reports based on fault triggers. Except classifying bug reports into bug/non-bug and Bohrbug/Mandelbug, the classification of Mandelbugs is the focus of this paper. Characteristic representation of common classification methods suffer from data sparsity and high dimensionality, thus we use word2vec, which can express words as low-dimensional word vectors with semantic relations in this paper. Furthermore, in order to improve the quality of classification, we analyzed the impact factors of classification. The results show that our method performs well in automatic classifying bugs into fault trigger classes.**

***Index Terms*—fault trigger; automatic classification; bug report; word2vec; Mandelbug;**

I. Introduction

Fault classification is of practical interest. Although significant efforts have been spent during software development and maintenance, failures are still a major concern. Understanding the types of defects can help developers take specific steps in current and future software releases. Grottke and Trivedi introduced a new classification method that dividing bugs into two categories: Bohrbug (BOH) and Mandelbug (MAN), according to the failure manifestation perspective [1]. Bohrbug is easy to be isolated, and whose manifestation is consistent under a failure. In contrast, Mandelbug is a special kind of faults, whose activation and/or error propagation is complex and difficult to be discovered by traditional techniques. Recent studies have focused on understanding the characteristics of Mandelbugs, including the proportion of Mandelbugs in all defects [2] , how to predict the location of Mandelbugs [3] and how to deal with failures caused by Mandelbugs [4]. Moreover, Mandelbug has two subcategories: non-aging related Mandelbug (NAM) and aging-related bug (ARB). Aging-related bug, which is first observed by Huang et al. [5], can cause an increasing failure rate and/or degraded performance [6]. Researchers in [7] proposed a more detailed bug type classification of aging-related bug and non-aging-related Mandelbug based on fault triggering conditions.

A more accurate classification of bug reports could be highly beneficial to analysis and research, since each type of defects needs to be tackled with corresponding measures, e.g., defects in Linux operating system [8]. However, with the increasing of the amount of bug reports, manual classification would bring a heavy burden to developers. Thus, it is necessary to help developers in classifying bug reports more efficiently (e.g., automatic classification). However, the study of fault trigger based bug report automatic classification is still rare. To the best of our knowledge, the only work so far is that in 2014, Xia et al. [9] proposed a fuzzy set based feature selection algorithm focusing on automatic classification of BOH/MAN categories of bugs. However, their work didn't address the automatic classification of subcategories of MAN, such as, NAM and ARB, as well as their subtypes. Therefore, it is necessary to be further studied.

In this paper, we develop a word2vec based automatic classification framework to classify bug reports into fault trigger categories from four-granularities. Since the introduction of word2vec by Google in 2013, it has been used to solve software engineering problems in several studies and has achieved good performance [10], [11]. The reasons why we choose word2vec as the based technique to develop an automatic classification approach are in the following.

First, previous study showed that linguistic information contained in bug tracking system entries is sufficient to automatically classify bug reports [12]. The bag-of-words model, which is widely used in bug report classification, can not express the semantic information of words and therefore sometimes would fail to achieve the desired accuracy. Word2vec, unlike the bag-of-words model, can bring additional semantic features by converting words and phrases into vector representations. This is helpful for classifying reports. Also, studies have shown that word2vec performs well in text categorization [13], [14]. Second, when encountering a large dataset, the dimensionality of the text represented by a vector space model (VSM) is the same as the number of all words that appear in the training dataset. Thus, it would lead to the curse of dimensionality. However, word embeddings trained by word2vec can avoid this problem by transferring words into K-dimensional vectors, where K is a constant. The parameter can be set up while training the model.

We train a word2vec model on bug report corpus. Based on the obtained word embeddings, four-granularities classification of bug reports are performed through machine learning classifiers. Moreover, factors that affecting the classification

*Corresponding author: Z. Zheng (Email: zhengz@buaa.edu.cn).

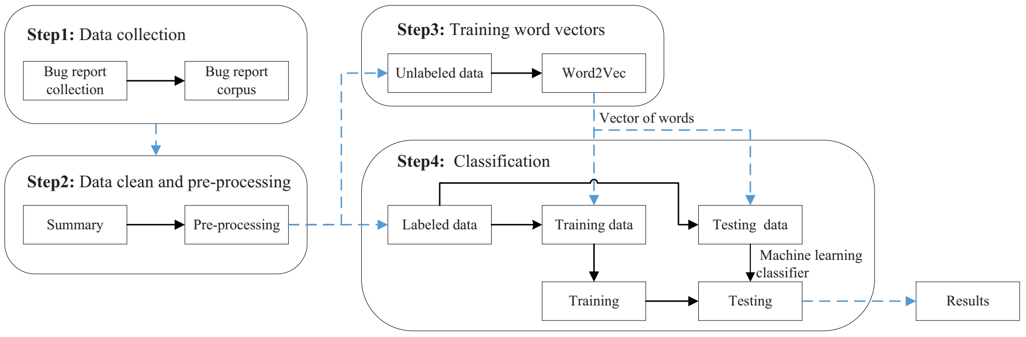

Fig. 1. Overall framework.

results are analyzed. In this study, we mainly focus on the following two research questions.

RQ1: How to perform the multi-granularity bug report automatic classification?

To answer the question, four-granularities automatic classification of bug reports, including bug/non-bug; BOH/MAN; ARB/NAM and the subtypes of NAM/ARB, is performed.

RQ2: What factors would affect the classification results?

In this part, the influences of word2vec model size and the domain of word2vec model training corpora on the classification results are explored.

The rest of the paper is organized as follows. Section II presents related work. Approach is described in Section III. In Section IV, the experiments and results analysis are presented. Section V presents the threats to validity. Finally, conclusion is given in Section VI.

II. Related Work

In this section, we first highlight the most related works about fault trigger based bug classification. Then, we present the researches and applications of word2vec in software engineering.

In 2008, Antonial et al. [12] first found that not all bug reports contain actual bugs, i.e., some problem reports concerning requests for defect fixing, enhancements, organizational issues, etc., are not considered as actual bugs. According to the fault trigger based classification method, Grottke et al. [6] examined the proportions of BOHs and MANs in JPL/NASA space mission critical software. Researchers in [7] extended a more explicit category of MAN, and concentrated on bug classification from four open source projects including Linux, MySQL, HTTPD and AXIS. Recently, the detailed distributions of subtypes of MANs in mobile operating system and Linux operating system are studied in [15] and [8], respectively.

Although several classification studies based on fault trigger conditions have been taken, all these works were performed manually, except for [9] and [16]. In [16], Frattini et al. automatically classified bug reports into workload-dependent and environment-dependent according to reproducibility characteristics. Xia et al. [9] developed a fuzzy set based feature selection algorithm for BOH/MAN automatic classification. In their work, evaluation experiments were conducted on Linux, MySQL, HTTPD, and AXIS, and the numbers of bug reports of these datasets are 267, 202, 141, and 199, respectively. The performance evaluation of their method mainly focuses on MANs classification. However, their work didn't address the subtypes classification of MAN. Thus, it is necessary to develop a fault trigger based bug report automatic classification method from more detailed granularities.

Other works related to the word embedding techniques used in our method are demonstrated in the following. There are many works for automatic classification of text [17], [18]. Most of them are implemented through the bag-of-words model, which can not express semantic information and would easily lead to dimension disaster. Recently, some novel models based on neural networks have been proposed [19], [20]. These models use deep neural networks to learn the context of the corpus to generate low-dimensional word vector representations called "word embedding" that can express the semantic information of words. In 2013, Google introduced a tool called word2vec, which is an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words [21], [22]. Word2vec has been used for text classification works [23] and produce outstanding results. Moreover, word2vec is an unsupervised learning method that allows us to learn word vectors from a large number of unlabeled bug reports from bug tracking systems.

III. Approach

In this section, we first introduce the main idea of this work. Then, we present the whole work frame. At last, we focus on the process of word embedding training and present a brief analysis of the word vectors trained via bug reports.

A. Main idea

The main goal of our research is to propose a multi-granularity automatic classification framework for bug reports. In addition to the classification of bug/non-bug and Bohrbug/Mandelbug, we further classify Mandelbug into aging-related bug and non-aging-related bug. Moreover, subtypes of aging-related bug and non-aging-related bug are also considered in the classification.

There are several classification algorithms commonly used in automatic classification of bug reports [24], [25]. In the implementation of these algorithms, documents are usually represented via bag-of-word features or by their term frequency-inverse document frequency. However, bag-of-words features suffer from data sparsity and high dimensionality. They treat words as atomic units, which couldn't capture semantic information between words. To mitigate this problem, we develop a word2vec based automatic classification method to classify bug reports into fault trigger categories. Word2vec model can express words as low-dimensional word embeddings with semantic relations, and the more similar the words, the closer their positions are in the vector space. This would be beneficial in bug report classification.

B. Overall Framework

Fig. 1 depicts the overall framework of our approach. The framework consists of four steps, each of which will be described in detail below.

- **Data collection**. In this step, two parts of data are needed to be collected. One part of them is unlabeled bug reports used for word2vec model training. We extract bug reports to a local computer from bug tracking systems by a web crawler that we designed. The other part of bug reports are labeled, which will be used for classifier training and testing.
- **Data clean and pre-processing**. For unlabeled data, summary and description parts of bug reports are extracted. While for labeled data, only summary part is needed. The extracted data is further cleaned and pretreated (i.e., word tokenization, remove stop words and punctuations, and lemmatization).
- **Training word vectors**. The vector representation of the word is generated through skip-gram model of word2vec. For two word vectors, cosine similarity between them determines their similarity. The greater the value, the closer the semantics of the two vectors are. The generated word vectors would be used as features of the bug reports.
- **Classification**. This is the last step. The input data of the classifiers is the labeled data generated from step 2. After getting the vector representation of each bug report, random cross validation is taken to evaluate the performance of our method. Training data is used to train classifiers, and testing data is used to verify the performance of the classifiers. The evaluation metrics used in this paper are accuracy and F-measure.

C. Training Word Vectors

In this part, we focus on the process of training the word embeddings, and analyze the quality of the word embeddings trained via bug reports.

There are two algorithms available in word2vec, continuous bag-of-words and continuous skip-gram [21], [22]. These models use deep neural networks to learn the context of the corpus to generate low-dimensional word vector representations, which is called "word-embedding". Continuous bag-of-words model predicts the current word based on its context, while skip-gram model predicts the surrounding words given the current word.

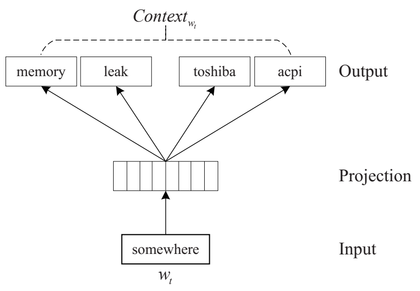

Fig. 2. An example of skip-gram model.

Our work uses the skip-gram model, which has been used to solve software engineering tasks and work well [10], [11]. Specifically, for given a current word w_t, we denote the set of the context of w_t as $Context_{w_t}$. Taking a MEM bug as an example, whose bug report's summary is "memory leak somewhere in toshiba_acpi". Fig. 2 illustrates the training procedure employed by the skip-gram model when it reaches the current word $w_t = somewhere$, which is mapped to its word vector v_{w_t}. It is used to predict the word vectors of its left and right C surrounding words (note, $C = 2$). The context words of $w_t = somewhere$ is $Context_{w_t} = \{memory, leak, toshiba, acpi\}$. The training process optimizes the word embedding v_{w_t} and the neural network model parameters to maximize the objective function f.

$$f = \frac{1}{T}\sum_{t=1}^{T}\sum_{w_c \in Context_{w_t}} logp(w_c|w_t) \tag{1}$$

In Eq. (1), w_c is the word in context of w_t. T denotes the whole length of the word sequence. The probability $p(w_c|w_t)$ is also formulated using a soft-max function:

$$p(w_c \epsilon Context_{w_t}|w_t) = \frac{exp\left\{v_{w_c}^T \cdot v_{w_t}\right\}}{\sum_{j=1}^{W} exp\left\{v_{w_j}^T \cdot v_{w_t}\right\}} \tag{2}$$

Where v_{w_c} is the word vector of word w_c, and W is the vocabulary length of all words.

For training the skip-gram model, we use a large dataset containing 134277 bug reports from Linux, MySQL, HTTPD and AXIS. All the words in the vocabulary of the corpus can be represented as a K-dimensional vectors where K is a variable parameter and its value is set as 350.

To measure the quality of word vectors, we present three example words and their similar words (i.e., *error*, *bug* and *memory*) in Table I. Note, the similarity between two words is measured by cosine distance. For instance, the greater the

TABLE I
EXAMPLES OF SIMILAR WORDS AND THEIR SIMILARITIES.

error		**bug**		**memory**	
Words	**& Similarity**	**Words**	**& Similarity**	**Words**	**& Similarity**
stating	0.663	failed	0.728	consumption	0.657
saying	0.612	filing	0.689	addresssanitizer	0.643
errorcode	0.601	issue	0.665	leaked	0.626
paradox	0.581	investigating	0.646	permgen	0.619
eror	0.577	apology	0.643	excessive	0.616
ioerror	0.571	bugfix	0.636	grow	0.615

value, the closer the semantics of the two vectors are. It can be observed that most of similar words are synonyms, such as *errorcode* is like *error*, *issue* is similar to *bug*. There are some other cases that output words are common matched, such as *leaked* with *memory*, memory error detector (e.g., *address-sanitizer*) with *memory*. This shows that the word vectors can present semantic information between words preferably.

To represent a sentence, we average the vectors of all the words in it. Each sentence can be represented as a word embedding vector, which would be further used as an input of machine learning classifiers.

IV. EXPERIMENTS AND RESULT ANALYSIS

In this section, we first present experiment setups, and then investigate the two questions proposed in Section I.

A. Experiment Setups

Five datasets are used to evaluate our method: Linux data1, Linux data2, MySQL, HTTPD, and AXIS, as shown in Table II (*#reports* means the number of reports). Among them, Linux data1 is a manual classified dataset from [8]. The other four labeled datasets are provided by [7].

We first perform the experiments on Linux data1 and classify bug reports from four-granularities. In addition, to compare with the classification results of BOH/MAN obtained by Xia et al. [9], we implement the experiment using the same datasets form Ref. [7].

For RQ2, we download the word2vec model trained by Google named "GoogleNews-vectors-negative300.bin.gz", which contains 3 million words from Google News. Besides, we train another word2vec model using bug report corpus. Its dimension of vectors is 300, which is same as Google's model. Details of the two models are displayed in Table III. Note that, *#words for training* means the number of words used for training the model, while *#words in model* means the number of words in model dictionary. Based on these two models, experiments are carried on all the five datasets in Table II.

TABLE II
DETAILS OF DATASETS.

Project	**#reports**	**Time frame**
Linux data1	5741	Nov 2002 - Nov 2016
Linux data2	346	Jul 2003 - May 2011
MySQL	244	Aug 2006 - Feb 2011
HTTPD	157	Mar 2002 - Oct 2007
AXIS	216	Jul 2001 - Nov 2005

TABLE III
DETAILS OF WORD2VEC MODELS.

Model	**#words for training**	**#words in model**
Google news	100 billion	3 million
Bug reports	16 million	60 thousand

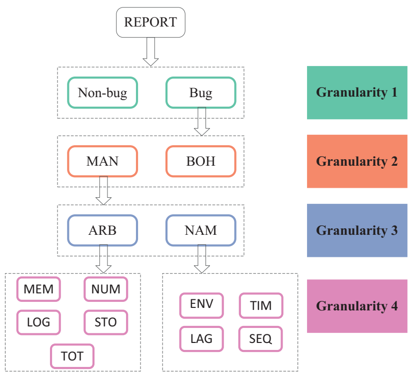

Fig. 3. Multi-granularity automatic classification.

B. Multi-granularity Classification of Bug Reports

In this part, we take four-granularities classification on the 5741 bug reports from Linux kernel Bugzilla, as depicted in Fig. 3. The four-granularities include: (1) bug/non-bug; (2) BOH/MAN; (3) ARB/NAM; (4) subtypes of ARB/NAM. Note that, the subclasses of ARB are LOG, MEM, NUM, STO and TOT, while the subclasses of NAM include TIM, ENV, LAG and SEQ [7]. In the following, seven classifiers are used for comparing, including Stochastic Gradient Descent classifier (SGD), Logistic Regression Classifier (LR), Gaussian Naive Bayes (GNB), Gradient Boosting Classifier (GBC), Decision Tree Classifier (DTC), Random Forest Classifier (RFC) and Linear Discriminant Analysis classifier (LDA).

1) Granularity 1: bug and non-bug: In this part, reports concerning requests for new features or for enhancements, documentation issues (e.g., missing, outdated documentation, or harmless warning output), compile-time issues (e.g., make errors or linking errors), operator errors and duplicates, would be regarded as non-bugs. Otherwise, they are bugs. For Linux data1, among 5741 bug reports, 4378 reports were identified as actual bugs, the proportion is 76.26% [8]. It means that 23.74% are not real bugs. The results of bug/non-bug classification are shown in Table IV. We get 71.8–83.4% accuracy rate, and F-measures of bug and non-bug are 80.9–90% and 46–61%, respectively.

2) Granularity 2: BOH and MAN: The real bugs can be further divided into BOHs and MANs, which is the second granularity considered in this paper. Among 4378 real bugs in Linux data1, 2444 are BOHs, accounting for 55.82%, while

TABLE IV
RESULTS OF BUG/NON-BUG CLASSIFICATION.

Classifier	Accuracy	F-measure		
		bug	non-bug	Avg.
SGD	0.773	0.848	0.540	0.774
LR	**0.832**	**0.893**	**0.610**	**0.842**
GNB	0.740	0.815	0.557	0.729
GBC	0.829	0.894	0.565	0.848
DTC	0.718	0.809	0.460	0.715
RFC	0.791	0.866	0.528	0.802
LDA	**0.834**	**0.900**	**0.598**	**0.848**

TABLE V
RESULTS OF BOH/MAN CLASSIFICATION.

Classifier	Accuracy	F-measure		
		BOH	MAN	Avg.
SGD	0.621	0.688	0.517	0.622
LR	**0.682**	**0.750**	**0.560**	**0.688**
GNB	0.663	0.716	0.586	0.661
GBC	**0.691**	**0.765**	**0.547**	**0.703**
DTC	0.590	0.658	0.489	0.589
RFC	0.629	0.694	0.529	0.629
LDA	0.677	0.747	0.553	0.684

TABLE VI
COMPARISON WITH USES [9].

Evaluation	Techniques	HTTPD	Linux	MySQL	AXIS	Avg.
MAN F-measure	USES	0.375	0.524	0.615	0.298	0.453
	word2vec+GNB	0.466	0.580	0.615	0.449	0.528
	Impro.	**24.267%**	**10.687%**	**0.000%**	**50.671%**	**16.556%**
BOH F-measure	USES	0.872	0.587	0.758	0.906	0.781
	word2vec+GNB	0.910	0.517	0.765	0.947	0.785
	Impro.	**0.436%**	-11.925%	**0.923%**	**4.525%**	**0.512%**
Accuracy	USES	0.787	0.558	0.703	0.834	0.721
	word2vec+GNB	0.847	0.553	0.709	0.904	0.753
	Impro.	**7.624%**	-0.896%	**0.853%**	**8.393%**	**4.438%**

1591 are MANs, accounting for 36.34% [8]. We classify 4035 labeled bug reports (i.e., BOHs + MANs) into BOHs and MANs automatically, and the results are exhibited in Table V. We get 59–69.1% accuracy rate, 65.8–76.5% and 48.9–58.6% F-measure of Bohrbug and Mandelbug, respectively.

In [9], Xia et al. have performed this granularity automatic classification in four software systems. To verify the effectiveness of our method, an evaluation is made using the same datasets [7], as shown in Table II. Due to the difficulty in classifying MANs manually, the performance of MAN classification results should be paid more attention. Therefore, in this comparison experiment, GNB classifier is used, which got the best MAN F-measure evaluation in Table V. Table VI presents the results comparing with the method in [9]. It can be observed that, our method improves the F-measures of MAN by 24.267%, 10.687% and 50.671% for HTTPD, Linux and AXIS, respectively. The improvement of average MAN F-measure of the proposed method is 16.556%.

3) Granularity 3: NAM and ARB: MAN could be further categorized as ARB and NAM, depending on whether the bug is aging-related or not. This is the third granularity. More detailed categories would help developers implement effective test policies or fault mitigation methods when dealing with these bugs. According to Linux data1 [8], there are 205 ARBs, which accounts for 12.88% of 1591 MANs. The results of automatic classification are shown in table VII. Note that, with respect to all classifiers, the accuracy rates range from 75.3% to 88.2%, while the range of F-measures of ARB and NAM are 19.2% to 39% and 84.6% to 93.5%, respectively.

TABLE VII
RESULTS OF ARB/NAM CLASSIFICATION.

Classifier	Accuracy	F-measure		
		ARB	NAM	Avg.
SGD	0.856	0.358	0.919	0.865
LR	0.856	0.390	0.918	0.861
GNB	0.753	0.371	0.846	0.721
GBC	**0.882**	**0.303**	**0.935**	**0.909**
DTC	0.797	0.260	0.882	0.792
RFC	**0.874**	**0.192**	**0.932**	**0.912**
LDA	0.841	0.369	0.909	0.843

4) Granularity 4: Subtypes of NAM/ARB: This is the forth granularity. Researchers in [7] defined more detailed subtypes for ARB (i.e., MEM, STO, LOG, NUM and TOT) and NAM (i.e., ENV, LAG, TIM and SEQ). The results of these two aspects are presented in the following.

Automatic classification of ARB subtypes: In Linux data1 [8], the largest subtype of ARB is MEM, whose number of bugs is 141 account for 68.78% of 205 ARBs. The number of STO, NUM, LOG and TOT are 16 (7.8%), 12 (5.85%), 11 (5.37%) and 3 (1.46%), respectively. The remaining 22 bugs couldn't be classified, due to the lack sufficient information. Since the number of TOT is only 3, this subtype is ignored in this experiment. The automatic classification results are presented in Table VIII. It can be observed that the range of accuracy rates are from 65.0% to 78.1%, while the average F-measures are from 65.5% to 74.4%.

TABLE VIII
Results of ARB/NAM Subtypes Classification.

ARB							NAM					
Classifier	Accuracy	F-measure					Classifier	Accuracy	F-measure			
		LOG	MEM	NUM	STO	Avg.			ENV	LAG	TIM	Avg.
SGD	0.652	0.066	0.795	0.215	0.227	0.661	SGD	0.548	0.558	0.436	0.591	0.548
LR	0.675	0.072	0.809	0.258	0.241	0.678	LR	0.549	0.560	0.436	0.593	0.550
GNB	**0.781**	**0.111**	**0.876**	**0.268**	**0.165**	**0.744**	**GNB**	**0.576**	**0.619**	**0.432**	**0.584**	**0.586**
GBC	0.740	0.026	0.855	0.155	0.137	0.696	**GBC**	**0.587**	**0.610**	**0.724**	**0.626**	**0.592**
DTC	0.650	0.056	0.800	0.135	0.159	0.655	DTC	0.478	0.497	0.358	0.520	0.480
RFC	**0.776**	**0.015**	**0.876**	**0.135**	**0.127**	**0.720**	RFC	0.548	0.584	0.390	0.575	0.550
LDA	0.661	0.071	0.801	0.199	0.230	0.666	LDA	0.478	0.492	0.377	0.522	0.486

Automatic classification of NAM subtypes: According to Linux data1 [8], the number of ENV, LAG, TIM and SEQ are 506 (36.51%), 265 (19.12%) 516 (37.23%), and 10 (0.72%), respectively. In this part, we only consider subtypes ENV, TIM and LAG, since these subtypes are the major classes of NAM according to previous studies [7], [8], [15], and they have sufficient samples for the classification. The automatic classification results are exhibited in Table VIII. The results show that the accuracy rates range from 54.8% to 58.7%, while the average F-measures are from 48.0% to 59.2%.

C. Impact Factors of Classification Results

In the following, we investigate the impact of two factors on the classification results, i.e., word2vec model size and word2vec model training corpus' domain.

1) The Effect of Model Size on Classification Results: For training word2vec model, we used a total number of 134277 bug reports from HTTPD, Linux, MySQL and AXIS. Since the most important thing in the process of model training is to learn the semantic information of the sentence in the reports, we use both summaries and descriptions of bug reports as model training corpus. Therefore, the number of items in training corpus is 268554. In order to analyze the impact of word2vec model size on classification results, we built several different size of the word2vec models using randomly selected items from the training corpus (i.e., the number of training corpus items ranging from 20000 to all items). The experiments are conducted in the datasets of Table II with GNB classifier, and the results are depicted in Fig. 4. It can be seen that with the increasing of model size, both accuracy and F-measure are improved. Therefore, we can conclude that the classification results can be further improved by increasing the corpus size.

2) The Effect of Model Training Corpus' domain on Classification Results: In this part, we evaluate the impact of training corpora from different domains on classification results. Experiments are performed on all the five datasets (i.e., datasets in Table II) using both the Google news model and the model we have trained with bug reports. Note that, GNB classifier is used in this part. As displayed in Fig. 5, although the corpus used in our model contains less than sixty thousand words, which is significant smaller than the Google news model (exhibited in Table III), the classification results (i.e., average accuracy and F-measure across all the five datasets) obtained by our model are better than that gotten by Google news. This indicates that the classification results would be affected by the domain of training corpus of the word2vec model. A more relevant training corpus can bring better classification performance.

V. Threats to Validity

In this part, we identify the following threats that might appear in this paper.

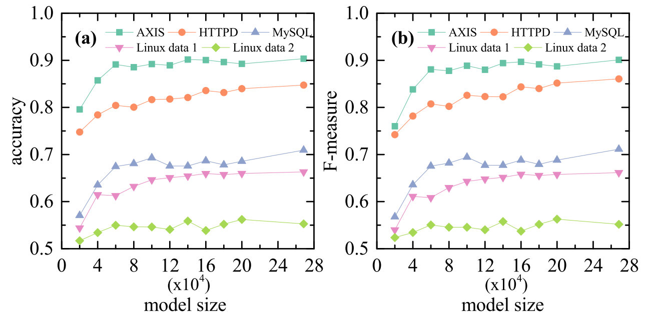

Fig. 4. The effect of model size on classification results. (a) The influence of model size on accuracy. (b) The influence of model size on F-measure.

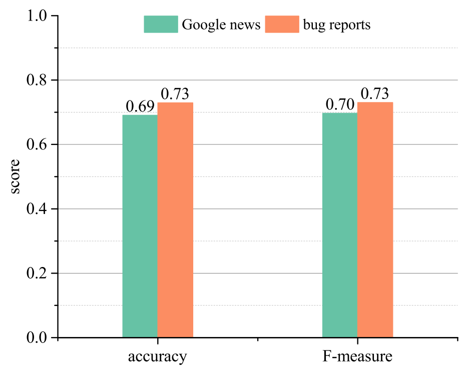

Fig. 5. The effect of model type on classification results. (a) The influence of model training corpus' domain on accuracy. (b) The influence of model training corpus' domain on F-measure.

Internal threats: the experiments are performed on datasets provided by Refs. [7] and [8]. Although We have conducted our experiments carefully and removed the duplicated bugs from the datasets, there may still have errors that we have not noticed. Besides, since the reports are written by users and developers, the accuracy and completeness of the description could also influence the automatic classification results.

External threats: we use hundreds of thousands of bug reports for word vectors training, but the word vectors can be further improved. In the future, we plan to mitigate this threat further by implement a larger corpora to make the results more accurate.

VI. Conclusion

In this paper, we proposed a word2vec based automatic classification method for categorizing bug reports into fault trigger classes from four-granularities, including bug/non-bug, BOH/MAN, ARB/NAM and subtypes of ARB/NAM. The empirical study was mainly performed on the manual classified dataset which includes 5741 bug reports from Linux kernel Bugzilla. Seven different classifiers were used for comparison, it is shown that our method performs well and results obtained by each classifier are different at various granularities. Moreover, the impact factors of classification results were investigated. It is found that with the increasing of word2vec model size, the performance of automatic classification would improve either. Furthermore, the domain of word2vec training corpus would also influence the classification result, which indicates that the more relevance domain of training corpus of word2vec model, the better classification performance.

In the future, we plan to further improve the effectiveness of our approach by using more bug reports to train the model. We will also explore more factors that would affect the classification results.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (Grant No. 61772055).

References

[1] M. Grottke and K. S. Trivedi, "A classification of software faults," *Journal of Reliability Engineering Association of Japan*, vol. 27, no. 7, pp. 425–438, 2005.

[2] R. Chillarege, "Understanding bohr-mandel bugs through odc triggers and a case study with empirical estimations of their field proportion," in *Software Aging and Rejuvenation (WoSAR), 2011 IEEE Third International Workshop on*. IEEE, 2011, pp. 7–13.

[3] G. Carrozza, D. Cotroneo, R. Natella, R. Pietrantuono, and S. Russo, "Analysis and prediction of mandelbugs in an industrial software system," in *Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on*. IEEE, 2013, pp. 262–271.

[4] M. Grottke, D. S. Kim, R. Mansharamani, M. Nambiar, R. Natella, and K. S. Trivedi, "Recovery from software failures caused by mandelbugs," *IEEE Transactions on Reliability*, vol. 65, no. 1, pp. 70–87, 2016.

[5] Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton, "Software rejuvenation: Analysis, module and applications," in *Fault-Tolerant Computing, 1995. FTCS-25. Digest of Papers., Twenty-Fifth International Symposium on*. IEEE, 1995, pp. 381–390.

[6] M. Grottke, A. P. Nikora, and K. S. Trivedi, "An empirical investigation of fault types in space mission system software," in *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*. IEEE, 2010, pp. 447–456.

[7] D. Cotroneo, M. Grottke, R. Natella, R. Pietrantuono, and K. S. Trivedi, "Fault triggers in open-source software: An experience report," in *Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on*. IEEE, 2013, pp. 178–187.

[8] G. Xiao, Z. Zheng, B. Yin, K. S. Trivedi, X. Du, and K. Cai, "Experience report: Fault triggers in linux operating system: From evolution perspective," in *Software Reliability Engineering (ISSRE), 2017 IEEE 28th International Symposium on*. IEEE, 2017.

[9] X. Xia, D. Lo, X. Wang, and B. Zhou, "Automatic defect categorization based on fault triggering conditions," in *Engineering of Complex Computer Systems (ICECCS), 2014 19th International Conference on*. IEEE, 2014, pp. 39–48.

[10] X. Yang, D. Lo, X. Xia, L. Bao, and J. Sun, "Combining word embedding with information retrieval to recommend similar bug reports," in *Software Reliability Engineering (ISSRE), 2016 IEEE 27th International Symposium on*. IEEE, 2016, pp. 127–137.

[11] Y. Uneno, O. Mizuno, and E.-H. Choi, "Using a distributed representation of words in localizing relevant files for bug reports," in *Software Quality, Reliability and Security (QRS), 2016 IEEE International Conference on*. IEEE, 2016, pp. 183–190.

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

[13] D. Rahmawati and M. L. Khodra, "Word2vec semantic representation in multilabel classification for indonesian news article," in *Advanced Informatics: Concepts, Theory And Application (ICAICTA), 2016 International Conference On*. IEEE, 2016, pp. 1–6.

[14] M. Hughes, I. Li, S. Kotoulas, and T. Suzumura, "Medical text classification using convolutional neural networks," *arXiv preprint arXiv:1704.06841*, 2017.

[15] F. Qin, Z. Zheng, X. Li, Y. Qiao, and K. S. Trivedi, "An empirical investigation of fault triggers in android operating system," in *Dependable Computing (PRDC), 2017 IEEE 22nd Pacific Rim International Symposium on*. IEEE, 2017, pp. 135–144.

[16] F. Frattini, R. Pietrantuono, and S. Russo, "Reproducibility of software bugs," in *Principles of Performance and Reliability Modeling and Evaluation*. Springer, 2016, pp. 551–565.

[17] F. Sebastiani, "Machine learning in automated text categorization," *ACM computing surveys (CSUR)*, vol. 34, no. 1, pp. 1–47, 2002.

[18] M. Lan, C.-L. Tan, H.-B. Low, and S.-Y. Sung, "A comprehensive comparative study on term weighting schemes for text categorization with support vector machines," in *Special interest tracks and posters of the 14th international conference on World Wide Web*. ACM, 2005, pp. 1032–1033.

[19] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[20] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167.

[21] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[22] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[23] J. Lilleberg, Y. Zhu, and Y. Zhang, "Support vector machines and word2vec for text classification with semantic features," in *Cognitive Informatics & Cognitive Computing (ICCI* CC), 2015 IEEE 14th International Conference on*. IEEE, 2015, pp. 136–140.

[24] M. Y. Javed, H. Mohsin *et al.*, "An automated approach for software bug classification," in *Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on*. IEEE, 2012, pp. 414–419.

[25] Y. Zhou, Y. Tong, R. Gu, and H. Gall, "Combining text mining and data mining for bug report classification," *Journal of Software: Evolution and Process*, vol. 28, no. 3, pp. 150–176, 2016.