Software Defect Data Mining: A Survey of Severity Analysis

Wenjie Liu*
School of Software Technology, Dalian University of Technology, China

* Corresponding author. Tel.: 13390500898; email: liuwj@dlut.edu.cn
Manuscript submitted June 22, 2019; accepted August 6, 2019
doi: 10.17706/jsw.14.10.457-478

Abstract: Open source software USES software defect tracking system, which can effectively manage the related information of software defects, and build software defect data warehouse in the form of defect report. The severity attribute of software defect report can determine the important indicators such as the repairers, solving time and repairing rate of software defect. Much research on software defects focuses on severity analysis. In order to evaluate the work in the field of severity analysis, this paper reviews the existing studies. In particular, this paper introduces the main methods of severity study, and expounds the statistical characteristics analysis of severity attribute in software defect report data set. According to the research status, the research of severity analysis is divided into qualitative analysis and quantitative analysis, and analyzed in detail. At the same time, each part is empirically analyzed based on data from the Mozilla project and Eclipse project datasets. On this basis, this paper summarizes the existing work of severity analysis of defect report, and points out some possible problems in the work.

Key words: Software defect report severity, qualitative analysis, quantitative analysis, feature selection, machine learning.

1. Introduction

In recent years, with the increasing of software size and complexity, the possibility of software defects increases sharply. In order to manage and maintain software more conveniently, some typical defect tracking systems, such as Bugzilla [1], JIRA [2], etc. are mainly used to collect information related to defects generated by users in the process of using software [3]. In fact, the severity of defect is an important attribute of defect report, which is highly subjective and uncertain for users [4]. In the task assignment of defect report, the severity of defect is an important reference attribute for determining the priority of defect repair and determining the repairman [5]. Large losses can be avoided by automatically identifying the severity of defect reports and fixing higher severity defects early. Therefore, defect reporting severity identification becomes an important problem in software maintenance.

The severity of a software defect report includes, for example, severe (i.e., blockers, critical, major), indicating that the defect is a severe error; The relatively low severity no-severe (e.g., normal, minor, trivial) represents the relatively superficial and minor defects; Or simply represent an enhancement [6].Qualitative analysis was carried out on the severity classification, and the severity prediction research was mainly realized by optimizing the machine learning method. At the same time, in order to solve the influence of severity attribute on other attributes of software defect report, the research on attribute classification, true and false problems, repairers and other aspects shows that severity attribute can seriously affect the processing efficiency of software defect report. Most studies will conduct text repeatability analysis and

sequencing method repeatability analysis on data, so as to improve the accuracy of severity analysis. We call this classification of severity a coarse-grained approach. Although this method can achieve a better classification effect in the process of machine learning, it cannot accurately express the true severity of defect reports, which will prolong the repair time of defects and reduce the repair rate [7].

It is a research field of quantitative severity analysis to classify the machine learning process and data processing process in a fine-grained manner according to 7 attributes of severity. According to the influence of other attributes on the effect of severity quantitative analysis, researchers generally conduct researches in machine learning methods, information extraction methods, defect reassigning problems and other aspects [8]. At the same time, some researches focus on feature optimization of data sets. These studies use feature optimization methods to reduce feature dimensions and obtain data sets that can express the meaning more clearly, thus improving the severity prediction efficiency [9].

Based on the existing research work, an empirical analysis was conducted based on the Eclipse project and Mozilla project defect report data of Bugzilla defect tracking system. Precision (P), Recall (R) and f-measure (FM), the most commonly used metrics in text classification, were introduced to measure the prediction effect of defect severity [10]. Experimental results compare the performance of different combinations and get the optimal method route. The experimental results show that the Correlation Coefficient (CC) feature selection method for feature optimization data sets, the Multinomial Naive Bayes theorem (MNB) performance of machine learning algorithms optimal [11]. This result also confirms the existing research content.

The rest of this article is organized as follows. Section 2 presents some provisions that are mainly used for defect reporting severity analysis, the role of defect reporting severity in the life cycle, and some statistics on existing defect reporting in practice. Section 3 reviews the main content and research direction of defect report data statistics; section 4 reviews the research status of qualitative analysis of defect report severity; section 5 reviews the current situation and development direction of quantitative analysis of defect report; section 6 reviews the research work of software defect report severity and summarizes the whole paper.

Main contributions:

1) Summarize the data processing methods commonly used in the research field of severity analysis of software defect reports;
2) Analyze the statistical characteristics of existing typical data warehouses according to the data sets, data collection and data statistics used in the research field of severity analysis;
3) Classify the research progress of qualitative analysis on the severity attribute of software defect report, and explain the reasons and problems of good classification effect of existing research;
4) By comparing existing studies on quantitative analysis of severity, find out the reasons for the low accuracy of quantitative analysis, and conduct empirical analysis on existing solutions.

2. Background

In this section, we explain the basic concepts in the area of defect reporting severity analysis research. Then, the role of severity attribute in software defect reporting life cycle is described. We also collected some software defect report data and conducted empirical research on the existing correlation analysis methods.

2.1. Relevant Knowledge of Severity Classification Research

The defect reporter will standardize the defect data and form defect report through the defect tracking system deployed on the Internet. Defect reporting attributes include defect current status, participant information, time data of each phase, repair software information, and severity. The current status of

defects includes descriptive data such as number, title, introduction and summary of defects; Participants include administrators, reporters, developers, fixers and other data; Each stage includes submission time, repair time, dispatch time, completion time, closing time and other information; Repair software information including product name, module name, component name and other information; Severity is mainly used to display the severity and priority of software defects selected by users [12].

1) Defect Report. The defect report also has multiple descriptions based on the defect content. These descriptions are provided by developers, testers, and users. The ID number, title, report, modification time, severity, fixer, current status and other information of the defect will be described [13]. These reports typically form a large software defect data warehouse in a defect tracking system.
2) Severity. Severity is an important attribute of defect report. In the process of defect assignment, it is necessary to sort defects by referring to this attribute and select appropriate repairers. The accuracy of this attribute will seriously affect the efficiency of defect repair, the ratio of resources, the solving time and other important results. Software defect reporting severity attribute mainly consists of seven categories: blockers, critical, major, normal, minor, trivial and enhancement, among which the first three are considered as relatively serious level, and the last four are considered as less influential or functional improvement level [14].
3) Classification and Ordering of defect reports. A defect report consists of a large number of attributes, each of which indicates a feature of the defect. Among them, priority and severity attributes can influence the repair process of defect reporting. Therefore, according to the existing data of defect report, analyzing and predicting the priority and severity of defects can help the repairers deal with defects accurately [15].
4) Repetition rate of defect report. Different descriptions of defects by different reporters will result in different description information and properties of the same defect. Therefore, judging whether the defect is repeated or not based on the information of defect report can effectively reduce the working intensity of the repairers and save the time of defect repair. If repeated defect reports are processed before defect prioritization and severity classification, the resource consumption of classification can be reduced [8].

2.2. Empirical Research Data Collection

In this section, we use the software defect report data warehouse of Bugzilla defect tracking system to conduct an empirical study of software defect research methods. Bugzilla is an open source defect tracking system developed and maintained by Mozilla. Through recording and tracking software defects, this system establishes a perfect defect tracking system for users, and can provide services such as submitting, dispatching, repairing and closing of defects in the whole life cycle of registered software, which is of great significance in the field of software defect management and analysis.

2.2.1. Mozilla project

Mozilla released its source code by Netscape on March 31, 1998, and its derivatives include Firefox, Thunderbird, and SeaMonkey.

The project used Bugzilla platform to systematically track its defect reporting and processing process, and established a complete report generation, query, record, resolution, and report generation process. By the end of 2017, Mozilla had more than 4,700 developers and submitted more than 1.1 million defect reports. The percentage of reported defects resolved is shown in Table 1.

2.2.2. Eclipse project

Eclipse is an open source, extensible java-based development platform. It was founded in April 1999 by the IDE product development teams of OTI and IBM.IBM provided the initial Eclipse code base, including

Platform, JDT, and PDE, to the open source community in November 2001.

As of January 1, 2017, Eclipse hosted 496,821 valid defect reports in the Bugzilla defect reporting repository. Since Eclipse users are mainly program developers, the software defect reports uploaded by Eclipse are more professional and have higher reference and representativeness. Meanwhile, due to the professional characteristics of the reporter, the accuracy of the description of different issues is higher. The number of severity defect reports is shown in Table 2.

Table 1. Distribution of Resolved Bug Reports in the Mozilla Project

Severity	Count	Count Percentage	Resolved	Resolved Percentage
blocker	13576	1.14%	13458	99.13%
critical	82194	6.90%	78239	95.19%
major	81594	6.84%	77874	95.44%
minor	885328	74.27%	786968	88.89%
normal	45231	3.79%	40136	88.74%
trivial	18615	1.56%	17316	93.02%
enhancement	65537	5.50%	51100	77.97%

Table 2. Reports on the Different Levels of Severity of Eclipse Projects and the Distribution of Solutions

Severity	blocker	critical	major	minor	normal	trivial	enhancement
WORKSFORME	583	1291	3297	1246	19725	229	2248
MOVED	3	5	30	32	312	7	201
INVALID	670	1122	2736	962	18071	281	1972
DUPLICATE	1098	2253	5388	1871	35303	356	6768
WONTFIX	201	530	2105	1531	20524	353	9685
FIXED	4634	9058	25306	9251	194151	4104	31976
NOTECLIPSE	144	333	555	180	2825	56	369

Take the data for an Eclipse project, as shown in Fig. 1. An example of a defect report with Eclipse project ID 484884 that contains all the appropriate information, including ID, summary, description, status, resolution, priority, severity, comments, and other relevant information. Summary is a brief description of the defect report. Description is the detailed description of the defect report; Status is the current status of the defect report; Resolution is how defect reports are handled; Priority is the priority for defect reporting; Severity refers to the severity level of the defect report, which is also what we need to predict.

3. Statistical Analysis

The statistical information of software defect tracking system is quite different from each other. In view of many software defect report data warehouse, the statistical characteristic analysis of data warehouse has been carried out in the existing research. In this section, we summarize the current research situation on the problem of data collection and statistical analysis of defect reports, and conduct an empirical study on the Bugzilla software defect tracking system.

3.1. Data Collection of Defect Report

Due to different software defect tracking systems, software defect report data sets contain different attribute data. When you select data from defect reports to form an initial data set, you get a number of features. These characteristics will form a subset of the characteristics of the data set. Researchers need to construct classification models according to the characteristics of feature subsets and classification

characteristics. In the process of classification learning and prediction, researchers usually use the original data of existing classification results to train and predict classification algorithms. When the accuracy of the classification model is stable, the new defect report is predicted and analyzed.

1) Selection of defect report attribute data. Cubranic and Murphy used the summary property and description property of software defect report to complete the construction of data set [16]. Before extracting report attributes, Anvik et al. deleted reports of unconfirmed types, re-opened types and inexperienced repairers [17], [18]. Baysal *et al*. added comment attribute data on the basis of summary attribute and description attribute [19]. Bhattacharya et al and Tamrawi *et al* added report ID and fixer ID [20], [21] on the basis of summary attribute and description attribute. Park *et al*. [22] extracted the metadata of defect report by using the text content describing the attribute, and obtained the version attribute, platform attribute and milestone attribute of the defect. In order to improve the representation ability of feature subsets, Baysal and Aljarah *et al*. sorted the features in feature subsets by means of assignment to obtain the feature subsets with stronger expressive ability [23], [24].
2) Make use of repair ability of repairers to optimize data collection. Matter et al. sorted and classified software defect reports according to the repairers' ability, and obtained a software defect report classification model based on the contributions of repairers [25]. Servant and Jones combined the defect location attribute with the defect frequency attribute to predict the defect severity [26]. Shokripour *et al*. extracted and ranked the description content and severity of the new defect according to the ID, description property and annotation property of the new defect report and compared with the repaired defect report, and obtained the sorting method of code repair in a single defect report [27]. Based on previous studies, Kevic *et al*. obtained a classification method of similar defect report based on high repair rate, aiming at the role of repairers in the repair process of multiple similar defects, so as to improve the repair rate of new data sets [28]. This method relies too much on the attributes of fixer to construct the data set, which may lead to the reduction of the repair rate of some data.

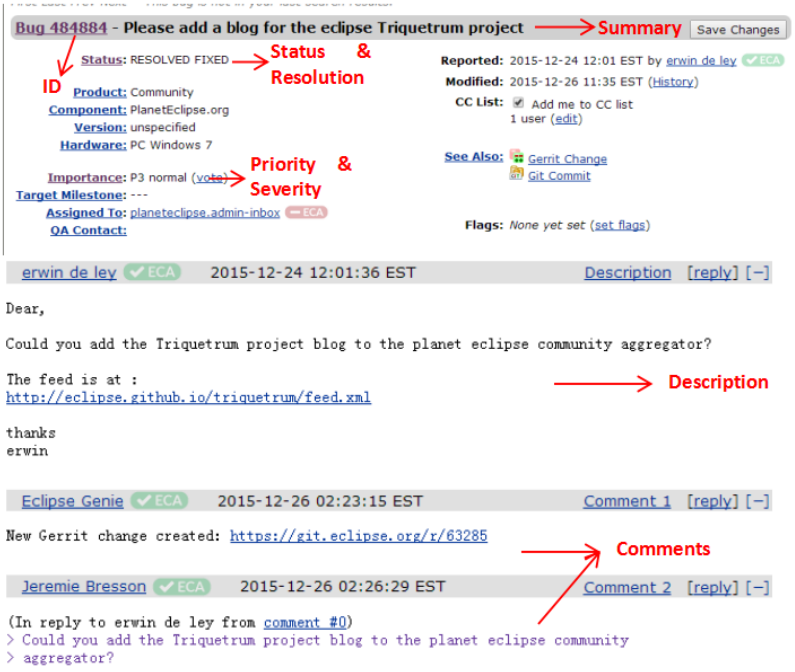

Fig. 1. Sample defect report.

3.2. Statistical Relationship between Severity and Reporting Attributes

With the increase of the amount of software defect report data, more and more people conduct statistical analysis for the mass software defect report. They get data associations between each software and project.

1) Relationship between severity and attribute of repairers. Bettenburg *et al*. conducted statistical analysis on software defect reports of Eclipse project in 2007, and found that there were close links among many attributes of defect reports [29]. They proposed an algorithm to evaluate the quality of defect reports using the content description of defect reports. According to the analysis of manually collected defect report data, code replication and software use tracking can determine the quality of software defect report, and incomplete data will lead to the quality decline of software defect report. Subsequently, they added Apache and Mozilla [30] to cover the project in order to prove the research results more realistically. In the same way, data affecting software quality and software defect report quality are analyzed by collecting software defect reports manually. They believe that important data affecting the quality of software defect report will be included in the process of software defect recurrence, defect tracking and defect repair testing. At the same time, the fixer will provide more defect tracking and defect repair test process data, and ordinary users will describe software defects. In order to evaluate the quality of software defect report automatically, they built a supervised learning model using the quality rating system. The reporter and fixer can use this model to evaluate new defects and recommend ways to improve them. However, no more reporters and fixers are using this model.
2) Relationship between severity and the duration attribute. Hooimeijer and Weimer [31] divided defect reports into CHEAP and EXPENSIVE types based on the solving time of defect reports.They determine whether the defect report is serious by determining whether the defect report has been resolved within a given time.Because this approach requires assumptions in advance, the severity of defect reporting is inversely proportional to the time it is resolved.As a result, this approach cannot accommodate more types of defect reporting.Based on this assumption, they constructed a model for the defect report severity prediction and made the prediction in the defect report data set they searched.
3) Relationship between severity and TRIAGER. Xie *et al*. conducted statistical analysis on Gnome and Mozilla projects of software defect tracking system and found that managers of software defect reports could dispatch defects, in which these managers were called TRIAGER [32]. TRIAGER reproduces the new defect report and manually dispatches it based on the existing fixer. The repaired defects are then inspected and the status of the software defect report is adjusted during the software defect report life cycle. But TRIAGER can only judge the severity of software defect reports based on professional experience, with low accuracy. It needs to be assisted by an automatic recommendation model.
4) Relationship between severity and repair process. Breu et al. extended the scope of software defect reporting to Mozilla and Eclipse projects [33]. But only 600 defect reports were selected for investigation. In the process of investigation, it is considered that the solution time of software defect report changes greatly. They suggest that users should refine the defect description to help reproduce the defect. During defect tracking and repair, they suggest that users should pay frequent attention to the process and evaluate the results produced.
5) Relationship between severity and reporter. John and Rajesh according to the survey data in 2016, the reporter, the role of development experience, software defect tracking system and other factors were analyzed, and think that the defect of experience in software defect report reporter reported the possibility of defect reports of high severity degree is higher, have experience in development of the

probability of high severity degree of defect report reporter is higher, software defect automation tools can better deal with software defect information [34].

3.3. Empirical Analysis

To verify and summarize the statistical characteristics of existing software defect reports. According to the software defect report data of Bugzilla software defect tracking system, statistical analysis is conducted in this paper. The data of Mozilla project and Eclipse project of this system are analyzed in detail, because the standardized data of these two projects can truly reflect the data characteristics of software defect report.

Among them, the resolution time of incomplete information type was the longest, up to 734 days. Unable to reproduce the resolution time of type report, which can reach 343 days; The resolution time for irrecoverable type reports cannot be described; Fixed type report resolution time, up to 139 days; Duplicate type reports need to be confirmed and can be solved in up to 82 days. Illegal reports can be resolved within 150 days [9].

3.3.1. Mozilla product defect report data feature analysis

The Mozilla project was opened by NetSape in March 1998, so the data used in this experiment started from January 1, 1999 and ended on December 31, 2017, with a 10-day cycle to analyze the newly submitted defect report. The submission frequency of defect reports is higher in 2002 and 2010-2017, and the number of defect reports submitted in each cycle exceeds 60,000. In the meantime, Mozilla products are in the midst of a transition from Mozilla1.0 to Firefox4.0.

The original data set used in this experiment is all the data of Mozilla defect report database from 1999 to 2017, with a total of 8497 developers and 1192,075 defect reports. Reports of resolved, closed and verified defects accounted for 89.35%, of which 47.52% were repaired and 19.84% were duplicated; The defect report to be solved accounted for 10.65%, and the average solution time of defect report was 772 days. The data ratio is shown in Fig. 2. The blocker with the highest severity has the highest defect report repair rate, while the enhancement level with the lowest severity has the lowest defect report repair rate, and the normal level with the highest number of defects has the lowest average solution time.

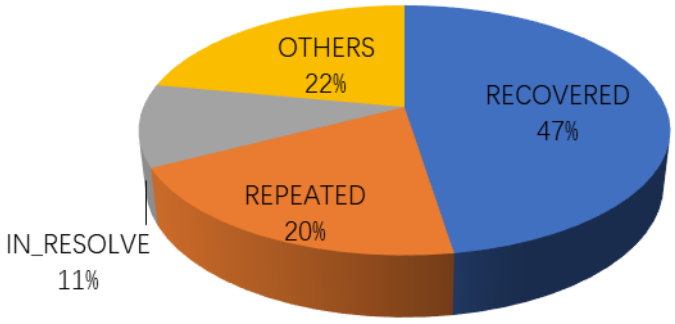

Fig. 2. Mozilla defect reports distribution.

3.3.2. Data feature analysis of eclipse product defect report

Eclipse was developed by IBM, opened in November 2001, and managed by the Eclipse foundation. From 2007 to 2015, it underwent nine revisions. The number of defect reports in the revision process fluctuated significantly.

From October 10, 2001 to January 1, 2017, the newly submitted defect report is analyzed in a 10-day

cycle. The most newly submitted defect reports were from October 10, 2001 to October 20, 2001, and 3,983 defect reports were submitted. After that, the most submitted defect reports were in late May and early June, 2006, early June, 2007 and mid-May, 2008.

The original data set used in this experiment is all Eclipse defect reports from 2001 to 2017, and the total number of defect reports involved is 496821, with 3407 developers participating in the repair. The quantitative distribution of each severity level is shown in figure 3. Among all Eclipse defect reports by January 1, 2017, the resolved, closed, verified defect reports account for 85.72% (425882/496821) and the pending defect reports account for 14.28%. Among the defect reports that have been solved, 65% are fixed and 12.45% are repeated. Defect report data set in the Eclipse project, report the degree of severity defects on defect repair rate, solve the length, the holder and the influence of product components and other attributes, and the statistical features of the Mozilla project defect report data set, the gap between the severity defect ratio not more than 0.1%, also determines the components, the holder and the defect fix rate, solve the length of a data set. However, in the minimum enhancement level defect, the amount addressed is around 1%.

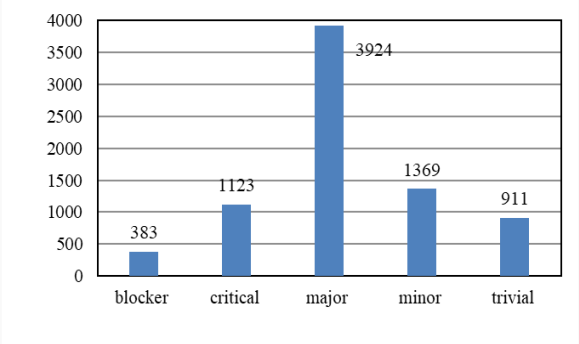

Fig. 3. Eclipse defect reports severity information.

3.4. Discussion

In the existing statistical analysis studies, most researchers believe that the repair time of software defects is closely related to the severity attribute of software defects report, and it is necessary to define the severity attribute by means of automatic classification. Such research is also gradually extended to software defect tracking system and the correlation of other attributes of software defect report.

In the Bugzilla defect reporting processing system, all reporting processing requires manual intervention. Therefore, the processing time of defect reports becomes its main attribute. In the defect report attribute of Bugzilla system, the resolution time is determined by the submission time and the resolved time. This attribute is mainly used to mark the duration of software defect resolution process. This is because the developer requests the reporter of the defect report for details of its report information and identifies the attributes of the report. Therefore, defect reports need to be typed.

According to the statistics, when a new version of the Mozilla product is released, the number of defect reports increases significantly. The severity of defect report will seriously affect the defect repair rate and solution time, as well as the defect repair rate and solution time of each component and its owner. Therefore, the severity analysis of defect reports is an important way to ensure the quality of Mozilla products.

4. Qualitative Analysis of Severity

The severity of software defect report mainly includes blockers, critical, major, normal, minor, trivial,

enhancement and other levels. In general, reports with severity levels of blockers, critical, major, and normal are considered severe, and the corresponding defect report is called severe. Minor, trivial, and enhancement are considered lower levels, and the corresponding defect report is called no-severe report. For the method of severity classification of software defect report, the evaluation is usually conducted according to the characteristics of solution duration, repair rate and holder. In the qualitative analysis of defect report, it is necessary to carry out the feature optimization process of the original data for the two types of severity classification, so as to achieve the strict matching between severity classification and feature subset, so as to obtain relatively accurate prediction results through machine learning method.

4.1. Qualitative Prediction of Severity

Automatic classification of software defect reports by text classification is the main method of relevant research. Using classifier to train and test the data of software defect report, the performance indexes of accuracy and recall rate need to be improved. Severity attribute is an important attribute of software defect report, and it is the required option submitted by the reporter. This property determines the order in which the fixer fixes the defect.

1) Use existing algorithms to realize prediction. In 2009, Matter D et al. achieved a defect prediction accuracy rate of 33.6% based on top-1 algorithm [25]. Alenezi M *et al*. solved the problem of software defect classification by text mining with the method of term selection [35]. In 2010, Lamkanfi *et al*. [36] conducted a further study on the classification of software defect reports by means of text mining, and analyzed the severity attribute of defect reports. Meanwhile, by comparing the accuracy of multiple machine learning algorithms in severity attribute prediction, they concluded that MNB algorithm had better effect. Later, in 2011, they classified and studied software defects from the perspective of software defect dispatch through the method of LDA theme model [37].
2) Improved machine learning method to achieve prediction. Menzies and Marcus et al. based on SEVERIS method combined with RIPPER method to predict the relevant text data of defect severity [38]. In 2015, Zhang *et al*. proposed a method based on text classification model CP to predict the severity of defect reports, and the experiment proved that the algorithm could effectively predict the severity of defect reports [39]. According to the results of these studies, naive bayesian algorithm and MNB algorithm are most suitable to classify defect reports into no-severe and severe[40].
3) Detailed severity progression study. Rajni *et al*., in 2017, refined the level of severity classification. They conducted machine learning through T01 vectors data set, IG method optimization feature subset and MMLR/MLR/DT model. The model conducts predictive performance analysis based on 10-fold validation for the five severity levels of NASA software defect data set, namely very high severity, high severity, medium severity, low severity, and very low severity, and compares the data subsets of various severity levels [41]. They rated MMLR and MLR as having better performance and consistency among the first four severity levels. DT algorithm performs well in subsets with very low severity.

4.2. Impact of Qualitative Analysis of Severity

Many researches focus on the influence of severity attribute on the quality of software defect report. These studies have delved into most of the attributes of defect reporting.

1) Classification of severity attributes. Lankanfi *et al*. found that component attribute information in software defect report data would affect the effect of qualitative analysis of severity, and based on the correlation between component attribute and severity attribute, obtained data mining model, and completed qualitative prediction of severity of software defect report [42]. Herraiz *et al*. 's data on the Bugzilla defect tracking system [43]. Compared with the completed software defect data, it is

concluded that the severity attribute of software defect report can be reduced to 3, and the data collective product can be greatly reduced. However, this method cannot determine the true severity level of software defect report and cannot provide accurate information of software defect report assignment for repairers and administrators.

2) Data range of qualitative analysis. Wu *et al*. found from the description attribute of defect report that the description accuracy of defect by the reporter would seriously affect the severity classification [44]. They proposed a data mining method based on defect description to extract defect report attributes from defect report database and complete qualitative analysis on whether defect report is completed [45]. Xia *et al*. analyzed the data of four open source projects (i.e., OpenOffice, Netbeans, Eclipse, and Mozilla) based on the change attributes of defect reports. They believe that the repair time of defects will increase with the number of defect changes. Rastkar *et al.* believed that the summary property of defect reports could determine whether the defect was correct or not [46]. Therefore, in the subsequent research, they reconstructed the summary attribute of the defect report to ensure the accuracy of the summary data to describe the defect.
3) Relationship between authenticity and severity. Antoniol *et al* first proposed the classification of defect reports in 2008 [47]. They simply divided defect reports into YES or NO, and did not classify the severity of defect reports classified in this way. In this process, they use three classification models to filter out the UNDEFECT reports from the three datasets. The three classification model is Alternating Decision Trees, Naive Bayes algorithm and logistic regression algorithm. These three datasets are Mozilla, Eclipse, and JBoss. The results of qualitative analysis are all over 75%. In order to improve the accuracy of qualitative analysis, Pingclasai *et al.* selected the theme model method [48]. However, in the data set they constructed, and the qualitative analysis effect of the thematic model method was lower than that of Naive Bayes classification model. Regarding the authenticity of defect reports, Herzig *et al*. conducted qualitative analysis on some defect reports of Bugzilla and Jira defect tracking systems [49]. More than 30 percent of defect reports they said were untrue. Of all defect reports, 40 percent were classified incorrectly and 39 percent could not be reproduced. Most of the defect reports stored in software defect tracking systems are open source software. Zanetti et al. discovered an automatic classification algorithm based on support vector machine by analyzing the defect report data of open source software [50]. This algorithm is used to automatically determine the authenticity of defect reports.
4) Relationship between fixer and severity. Canfora and Cerulo classify defect repair experience for defect reports based on repairers' attributes [51]. In this algorithm, the repairers are ranked in similar defect sets, and the repairers' capability database is constructed by information retrieval of repairers' information. However, their proposed approach may overlook some fixers who may have the ability to fix the above defect classifications. This approach may result in an increase in the workload of restorers who have experience with restoration.

4.3. Repeatability Analysis

Due to the uneven level of software defect report reporters, they cannot describe defects completely and accurately when uploading defect report data. Therefore, in the software defect reporting database, there will be a large number of duplicate reports. Before qualitative analysis of the severity of software defect reports, duplicate defect reports need to be eliminated. There are a lot of research contents in this field, including text comparison, ranking comparison, attribute enhancement and so on.

1) Text repetition analysis. The main description method for defect reports USES text. Therefore, the text needs to be split, cleaned and reconstructed when data analysis is carried out. Hiew breaks down the description text of defect report, and after removing the noise data, takes a single word as the

feature of the data set, and obtains the similarity order of the defect report tested according to the comparison of similar features in the feature sub-set [52].On this basis, Runeson et al. added other text attributes of defect report [53], so as to enhance the representation ability of feature subset of defect report. This method achieved good results in the repeatability detection of defect report in industrial software. Later, Jalbert and Weimer proposed a method of repeating attribute sequencing for the repeatability of defect reports [54]. This method can classify the repeatability of existing defect reports. After training with existing data, it is possible to predict whether new defect reports will be repeated or not. Sureka and Jalote combine n characters to obtain more ideographic subset of features and propose a new method of repeated defect reporting based on n-item character detection [55]. Such an approach can handle defect reports written in Asian languages. Sun et al. [56] proposed a new method based on recognition method, which assigns different weights to words in defect reports when calculating text similarity. Their research shows that their method is superior to all technologies that use textual information. Later, they further improved their method by introducing a text similarity measure called BM25 [57]. Later, based on BM25, Tian *et al*. proposed a classification method to consider the similarity between new defect reports and multiple existing defect reports [58].This method is able to determine whether the defect represented by the new defect report is really a duplicate defect. Nyugen et al. proposed to use the theme model to detect repeated defect reports [59], which measures the semantic similarity between words. Unlike previous efforts to model defect reports as a set of words, Banerjee *et al*. proposed further consideration of word sequences when calculating text similarity between error reports [60].

2) Repeatability analysis of sorting methods. When judging the repeatability of the defect report, the defect report can be sorted by means of supervision and semi-supervision. The vectors with similar sorting positions have high similarity. Liu et al. proposed a supervised machine learning technology called Learn To Rank (LTR), which was applied To defect reporting analysis based on sequencing [61]. Meanwhile, Zhou and Zhang also proposed an ltr-based approach [62] that USES nine more complex defining features. Podgurski et al. started their research work by proposing a method to classify software defect reports [63]. This method marks the properties of life cycle phase and realizes the classification of defect report by undirected clustering. Later, Wang *et al*. proposed the method of using defect tracking to realize repeated defect report detection [64]. This method USES the adaptive weight allocation method to assign the weight of the natural language information of the defect report by learning the existing and repeated defect reports. In the end, they say, the combined technique is more accurate than a single technique. Based on this work, Song *et al*. made some modifications [65] and achieved better results in the IBM Jazz project data. Lerch et al. proposed a method to identify the stack trace in defect report [66], converted the stack trace into a group of methods, and sorted by calculating the similarity between methods to detect repeated defect reports.
3) Repeatability analysis of other attributes. There is a strong correlation with repeatability due to other attributes of defect reporting. Starting from the aspect of attribute correlation, many researchers conducted qualitative analysis on repeatability of software defect report. Feng *et al.* proposed the use of defect reporter profile information to enhance the effectiveness of existing techniques in detecting duplicate defect reports [67]. Alipour *et al*. proposed to use domain knowledge for repeatability detection, and the experiment showed that this research method can effectively improve the accuracy of repeated defect report detection [68]. Kim *et al*. proposed to use repair markers to describe multiple related defect reports [69], and to detect repeated defect reports by measuring the similarity between repair markers. Dang et al proposed a new model to measure the similarity between two defect data [70]. Specifically, in their work, they further considered the distance from the matching

function to the top frame and the offset distance between the matching functions.

4.4. Empirical Study on Qualitative Analysis of Severity

This experiment collected all software defect report data of Eclipse project from 2001 to 2017.According to the general practice in the field of software defect report, this defect report data according to the severity categories can be divided into serious (severe) and not (no - severe) two kinds big, serious defect report severity degree including blockers, critical, major, such as level, no serious defect report severity degree including normal, minor, trivial, level enhancement, etc. The data set includes 102,905 no-severe reports and 393,916 severe reports. Among which, the first 20 feature words with occurrence frequency in no-severe category are: Dialog, view, editor, file, error, page, prefer, project, type, new, select, eclip, javadoc, name, work, code, show, Java, miss, set, text, wizard, chang, task, message, method, class, button, wrong, The menu. The first 20 feature words with severe frequency are:Eclip, file, error, project, work, view, fail, npe, Java, build, test, open, creat, except, report, new, crash, problem, updat, org, plugin, import, run, compil, class, cau, set, gener, chang.

In order to guarantee the objectivity of the experiment, the comprehensive embodiment of vectors method in the current data set, the severity of software defect report forecast accuracy, the experiment selects four kinds of commonly used classical classification algorithms (NB, MNB, SVM and KNN), and the four classical vectors method (T01, TF, DF, TF - IDF), cross matching use, according to the software defect report confusion matrix severity prediction data content, in view of the classification algorithm forecast accuracy, Precision and Recall, F-Measure evaluation index, such as comprehensive comparison, Objectively judge and predict the effect. Before the training and prediction of classification algorithm, the feature subset of the data set is not optimized, and the same number of features (all features) is selected, and different classification algorithms are used for experiments.

As can be seen from the experimental results in Fig. 3, for the data set of this experiment, the MNB classification algorithm gets the best results, with weighted average Precision, Recall and F-Measure of 42.88%, 43.13% and 49.97%, respectively.

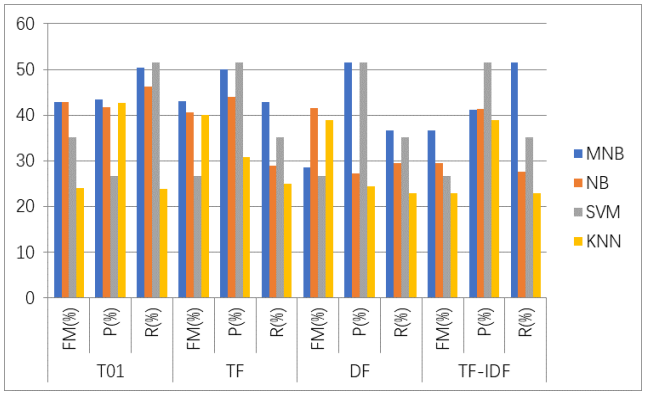

Fig. 3. The predicted results of different feature representation methods are Weighted F value comparison of different classification algorithms.

4.5. Discussion

In the process of qualitative analysis, the severity of software defect report can be divided into two categories: severe and no-severe. Many studies have focused on the methods and effectiveness of severity analysis. At the same time, some methods do preliminary processing on the repeatability of data sets, and they conduct severity analysis on the optimized data sets. However, all methods can only describe the

severity of the defect report in a summary manner, and cannot accurately indicate the severity.

5. Quantitative Analysis of Severity

The study of severity attribute has become the focus of software defect report. Studies that divide the 7 severity degrees of software defect reports into two categories (severe and no-severe) can achieve a better classification effect in the process of machine learning, but the failure to accurately express the true severity degree of defect reports will prolong the repair time of defects and reduce the repair rate. In order to review the research status of severity attribute, in this section, we summarize the research content of quantitative analysis of software defect reporting severity; secondly, the influence of other attributes on severity quantitative analysis is introduced. Thirdly, the function of feature selection algorithm in severity quantitative analysis is introduced. Finally, this paper classifies the six attributes of severity in a fine-grained way, and conducts a comprehensive empirical analysis of the machine learning process and the feature optimization process in the data processing.

5.1. Quantitative Study on Severity

Due to the significant influence of software defect severity analysis on defect repair, the research on defect severity is gradually increasing. Most of the researches are based on the data of open source defect tracking system for specific optimization and comparison research, and most of them adopt the improved classification, data set optimization method and priority determination method.

1) Machine learning methods. Tian *et al*. proposed a machine learning method based on information retrieval to predict the severity of defect reports [71]. In this algorithm, they used the derivative algorithm of BM25 to analyze the similarity of defect report. KNN algorithm is used in learning and prediction. Thus, the similarity-based severity prediction of defect reporting was obtained and applied to all 7 severity levels. In contrast, Menzies and Marcus work with fewer severity levels and a lower representation of defect reporting severity. Tian et al. further clarified the study of quantitative analysis of severity.
2) Research methods for specific data sets. To be find a severity prediction method suitable for data from the Bugzilla defect reporting tracking system. In 2016, Zhang et al. proposed a prediction algorithm based on the improved REP model and KNN classification algorithm to find the most similar defect reporting method [72], and verified the effectiveness of the algorithm through experiments. In 2017, Gao et al. proposed a method to predict severity using the thematic model and multiple features of existing defect reports, and proved the effectiveness of the method through experiments [73].
3) Severity quantitative analysis based on priority. In order to facilitate the prioritization of defect reporting, Yu et al. adopted neural network technology [74]. In addition, they reused data sets from similar software systems to accelerate evolutionary training to automate systems to solve defects. Kanwal and Maqbool proposed a defect priority recommendation method based on SVM and Naive Bayes classifier [75]. In addition, they also compared the accuracy of the results of these two classifications and found that SVM performed best when making text features recommend defect priority, while Naive Bayes performed better when using classification features.

5.2. Repairers Attributes of Severity Classification

As a result of the severity qualitative analysis, it is intended to identify the fixer. So, a lot of research has focused on predicting and assigning work to restorers. This process can be realized through machine learning, space vector model, information retrieval and other technologies.

1) Machine learning methods. Cubranic and Murphy proposed the first classification method of defect

report fixers based on machine learning in 2004 [16]. Based on the supervised learning method, the algorithm USES the historical information of defect assignment to train the machine learning model and predict the repairers of new defect report data. They used Naive Bayes model to predict the repairers and collected 15,859 defect reports. The final prediction accuracy reached 30%. Later, Anvik et al. stripped noise data from the data set for the above work [17], [18]. Their method filters out reports marked as Wontfix and Worksforme status, and also removes repairers who are no longer working and those who contribute less. Experiments show that this method improves the prediction accuracy by 20%. By comparison; it is found that SVM has the strongest defect reporting ability among Naive Bayes, support vector machine and C4.5 algorithm. In order to improve the accuracy of defect reporting classification, Bhattacharya and Neamtiu rely on feature optimization to improve the prediction accuracy of restorers [20]. Hu *et al*. started from the data set to the quantization process and improved the performance of similarity classification of defect reports by using the optimization processing of spatial vector model [76]. This method is superior to the prediction ability of Naive Bayes and Support Vector Machine.

Lin *et al*. analyzed Chinese text defect report data [77]. They use support vector machine-based classification algorithms to learn the title, description and process of defect reports. Finally, they complete the automatic dispatch of defect reports. The experimental results show that text data can describe the defect report more effectively in the defect report data warehouse. Xuan et al. first proposed to use semi-supervision method to deal with the problem of defect allocation [78]. Based on the semi-supervised learning method of expectation maximization; this method generates a weighted recommendation list for defect report classification, and USES the marked and unmarked data to implement defect assignment. Alenezi and Magel found a new entry point for defect dispatch from the perspective of fixer workload [35].

2) Information extraction method. Due to the defect report assignment process, after the comprehensive evaluation of the repairman, the current defect report description information analysis is added, and all factors are comprehensively analyzed to determine the repairman. Therefore, the extraction of useful information of defect report is an important content of defect dispatch. In order to find a better term selection technique, Alenezi and Magel compared and analyzed the influence of five word selection algorithms on the accuracy of defect report dispatch [35]. Matter *et al*. evaluated the contribution ability of the fixer, and based on the existing data, formed a vocabulary to implement the fixer assignment in a defect report [25]. This method mainly relies on the description information of defect report and the information extraction of repairers' repair ability. By comparing the coupling information between the two, the repair ability ranking of all repairers on the current defect can be obtained. This solves the experience value problem between experienced repairers and new repairers.
3) Defect reassigning. Since fixers may not handle defect reports assigned to them, these defects need to be reassigned. Therefore, in addition to the research on the automatic assignment of defect report, there is also some research on the re-dispatch analysis of defect report. According to the process principle of defect re-dispatch, Jeong *et al*. proposed the method of defect re-dispatch [79]. This method can effectively reduce the error redistribution by 72% and improve the accuracy of automatic error reporting by 23%. In addition, Bhattacharya and Neamtiu extended their work by invoking other attributes during the dispatch process to enhance the dispatch accuracy [80]. After evaluation, the prediction accuracy of this method reached 83.62% in defect report allocation. Tamrawi et al. reorganized the corresponding relationship between the description information of defects and the description information of repairers, and proposed a new method of repeated defect dispatch by referring to the description information of the repair experience of repairers [81]. This method USES

a fuzzy set approach to the Bugzie defect tracking tool data. Through the evaluation, the accuracy and efficiency are higher than other work. Xie *et al.* studied the Mozilla project and found that the probability of redistributing defect reports was related to the past activities and social networks of fixers [82]. Instead of studying the redistribution of defect report distributors, they did an empirical study of the redistribution of defect report fields, including the redistribution of reports to new fixers. From their research, fixing a redistributed defect usually takes more time than fixing a non-redistributed defect. Guo *et al*. conducted a large-scale study on the redistribution of defect reports in Microsoft Windows Vista regarding the redistribution of defect reports in non-open source software [83]. The study divided the distribution into five categories (i.e., root cause finding, ownership determination, poor quality of error reporting, difficulty in determining correct fixes, and workload balancing).

5.3. Feature Selection Algorithm in Quantitative Analysis

Feature selection is an important part of machine learning process which can simplify training sets. At present, there have been some studies on the severity prediction of software defects [7]. Most of these researches are based on text classification and text similarity [8]. Due to the large amount of software defect report data, high-dimensional feature sets will be generated, resulting in poor prediction effect, and the improvement of classification algorithm cannot greatly improve the prediction effect [9]. Therefore, how to reduce the feature dimension of prediction defect severity is an important problem for the recognition of the severity of missing item reports.

4) Function of feature selection algorithm. Zou *et al*. [84] first combined feature selection (i.e., CHI feature selection algorithm) with instance selection (i.e., ICF instance selection algorithm) to improve the accuracy of defect report classification and reduce training sets. From their experimental results, their method eliminated 70 percent of the words and 50 percent of the error reports when applying machine learning and the reduced training set provided better accuracy. Park et al. [35] redefined the defect assignment problem as one with better accuracy and cost, and proposed a cost-sensitive classification algorithm, which extracted defect features to train SVM models and reduced the cost by 30% after evaluation.
5) Optimization of feature selection algorithm. In 2015, Rajni *et al.* conducted data mining on the attributes of existing Andriod software defect report by using MMLR method and based on DCRS software defect tracking system. The method optimized the characteristics based on IG method and obtained the description document of software defect report [85]. At the same time; they describe the current research classification of software defect report in the relevant work section. Yang et al. studied performance of four feature selection algorithms in defect prediction, including information gain, chi-square test and correlation coefficient [86]. The experimental results showed that the feature selection algorithm could effectively improve the severity prediction effect. Since the main method of predicting the severity of software defect report needs to be realized by machine learning algorithm, how to choose better feature selection method and machine learning method to work together and improve the prediction efficiency is a new content of this research direction.
6) Feature subset optimization. In 2018, Anna carried out feature detection of software defect report by means of clustering [87]. In this method, tf-idf algorithm is used to realize the vectors process of text data. Meanwhile, two feature subsets are obtained by feature selection using clustering method. 8 features are selected from the first data set and 10 features are selected from the second data set. The algorithm achieves good results in the classification learning process.2018 using the cluster analysis for characteristics detection in software defect reports. Liu et al., in 2018, by means of optimal feature subset based on the gravity of the fine-grained classification, according to the characteristics of the

feature selection algorithm, change based on the characteristics of the Max common sequence of construction method, is put forward based on the characteristics of the sorting sequence reconstruction method, get the optimal feature subset, sharply reduce the feature dimension [88].In the process of quantitative analysis of severity prediction, the method obtains 76% prediction accuracy by inheritance feature selection algorithm.

5.4. Empirical Study on Quantitative Analysis of Severity

During the experiment, text data of summary attribute of Eclipse project was used to form the data set. The original text data is denoised to obtain a simplified text data set. By comparing the data sets generated in the vector-quantization process with multiple classification algorithms, it is concluded that the data sets extracted by T01 vector-quantization algorithm have the best prediction effect on the fine-grained severity of software defects in MNB classification algorithm. Get stable vectors method and the combination of classification algorithm, through multiple feature selection algorithm to the optimization of quantitative data set for fine-grained severity of 5 kinds of classification and three kinds of performance evaluation standard, to conduct a comprehensive performance comparison and evaluation after that feature selection method can significantly improve the predictive accuracy of software defect report severity of CC algorithm to predict the best effect.

Based on the commonly used vectors methods T01, TNUM, TF and tf-idf, four vectors data sets were obtained, and the number of features of the obtained feature subsets was 4,890.MNB, NB, KNN and SVM, commonly used classification algorithms, were used to conduct training and prediction based on 10-fold verification with 4 data sets respectively, and the evaluation results of prediction effect of severity of software defect report based on accuracy, recall rate and F value were obtained. Thus, the matching effects of four vectors methods and four classification algorithms are compared and evaluated. The prediction results of different vectors methods corresponding to different classification algorithms are shown in Fig. 4.

According to the above experimental results, the data set obtained by T01 vector quantization method has the best prediction effect in MNB classification algorithm, with F value of 42.91%.The data set obtained by TF method has the best prediction effect in NB classification algorithm, with F value of 42.88%.The data set obtained by DF vectors method has the best prediction effect in SVM and KNN classification algorithm, with F value of 35.09% and 25%.The data set obtained by tf-idf vectors method has the best prediction effect in NB classification algorithm, with F value of 36.66.The most effective matching method is to use T01 for vectors, and to predict the severity in MNB classification algorithm.

In order to comprehensively evaluate the performance indexes of all the data, and take the results as the reference value of the subsequent optimization process. In this experiment, all features of the data set obtained by T01 vectors method were selected to classify and predict the severity degree of fine granularity. This paper compares and analyzes the prediction effects of MNB, NB, KNN and SVM, commonly used classification algorithms, in 5 software defect reporting severity degrees (blocker, critical, major, minor and trivial). The evaluation of prediction effect was marked by P, R, FM and other indicators. KNN classification algorithm works best for data processing with blocker and critical severity levels; SVM classification algorithm is the best for data processing with the severity of major. NB classification algorithm works best for minor and trivial data processing. According to the prediction results, the F value of NB and MNB algorithm in this experimental data set is relatively high. Meanwhile, MNB algorithm has higher accuracy and recall rate than other algorithms.

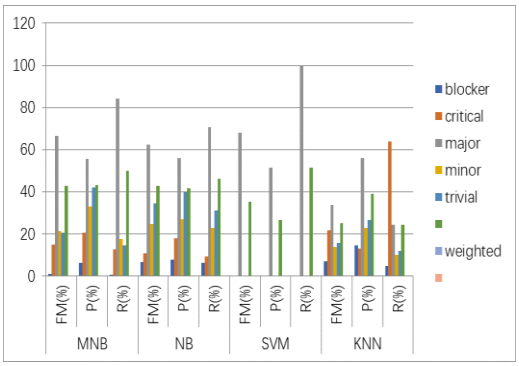

Fig. 4. Comparison of prediction results between different classification algorithms.

5.5. Comparison with Existing Algorithms

For the prediction of fine-grained severity of software defect report, some researchers have proposed better methods.This experiment compares the prediction effect of the existing most efficient processing method with the Eclipse project data set and the feature selection optimization method adopted in this paper.The accuracy (P), recall rate (R) and FM values of the predicted results of blocker, critical, major, minor and trivial as well as weighted average are compared, and the results are shown in Table 3.

According to the experimental result shows that the Eclipse project data set, the feature selection methods are introduced to the prediction accuracy for blocker in severity degree, major, minor, trivial level, and a weighted average, etc., are more than the existing results the best classification algorithm, only in the degree of severity to critical level of predicted results is slightly lower than the existing algorithm is low (0.17%).Among them, the weighted FM value increased from 44.50% to 48.62%, and the effect increased by 9.3%.Therefore, the experimental results show that CC feature selection algorithm can effectively improve the severity prediction effect of defect report.

Table 3. Comparison of the Effectiveness of Existing Severity Prediction Algorithms

Approaches	MNB-CC			Approach in		
Severity	P（%）	R（%）	FM（%）	P（%）	R（%）	FM（%）
blocker	32.48	3.94	6.65	0.38	3.33	0.69
critical	34.22	13.7	18.72	22.22	16.43	18.89
major	58.9	88.41	70.60	71.17	56.31	62.87
minor	41.30	23.38	29.21	19.80	29.22	23.61
trivial	49.31	25.68	31.88	19.98	51.98	28.87
weighted	50.65	55.02	48.62	45.74	43.32	44.50

5.6. Discussion

In the long-term research process, researchers generally believe that quantitative analysis with stronger ideographic ability should be introduced into the severity analysis of defect report. Such an analysis process can more accurately represent the importance of defects, thus affecting the important attributes such as defect reporting life cycle, resolution time and fixer. Although existing studies can prove the effectiveness of feature selection algorithm in identifying the severity of predicted defects, it is necessary to conduct in-depth research on which feature selection algorithm performs best and how to obtain better results.

6. Conclusion

There have been numerous studies on the severity of reported software defects. After introducing the

background and main methods of severity classification research, this paper analyzes the significance of severity attribute of defect report in the field of defect report research and makes targeted statistical analysis. It is considered that severity attribute analysis is an important part of software defect report, which can determine such important attributes as the resolution time, repair rate and repairers of software defect report life cycle, and it needs to be studied in depth. Then, the existing research field is divided into quantitative analysis field and qualitative analysis field. The main research achievements in two fields were investigated. It mainly includes the current situation of severity analysis, the relationship between severity attribute and other attributes, the application of feature optimization method, and the empirical analysis of relevant research results. The main techniques of machine learning and information retrieval are discussed.

However, many problems remain unsolved or even unstudied. Most researchers focus on the optimization of classification algorithms, but do little work on the selection of data sets, and do not study enough on the influence of data sets on prediction results. Many research methods fail to achieve satisfactory accuracy in quantitative analysis. In the aspect of data set optimization, the researchers put little effort into it. And the optimization of this aspect can bring better results. In the future, researchers may consider ways to improve the accuracy of existing automated methods.

Reference

[1] Sliwerski, J., Zimmermann, T., & Zeller, A. (2005). When do changes induce fixes? *International* Workshop *on Mining Software Repositories*.

[2] Schröter, A., Zimmermann, T., Premraj, R., *et al*. (2006). If your bug database could talk. *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*.

[3] Zimmermann, T., Premraj, R., & Zeller, A. (2007). Predicting defects for eclipse. *International Workshop on Predictor Models in Software Engineering.*

[4] Wu, R., Zhang, H., Kim, S., *et al*. (2013). Relink: Recovering links between bugs and changes. *ACM SIGSOFT Symposium on the Foundations of Software Engineering*.

[5] Bissyande, T. F., Thung, F., Wang, S., Lo, D., & Reveillere, L. (2013). Empirical evaluation of bug linking. *Proceedings of the 2013 17th European Conference on Software Maintenance and Reengineering.*

[6] Nguyen, A. T., Nguyen, T. T., Nguyen, H. A., & Nguyen, T. N. (2012). Multi-layered approach for recovering links between bug reports and fixes. *Proceedings of the ACM Sigsoft International Symposium on the Foundations of Software Engineering.* ACM.

[7] Bird, C., Bachmann, A., Rahman, F., & Bernstein, A. (2010). Linkster: Enabling efficient manual inspection and annotation of mined data. *Proceedings of the ACM Sigsoft International Symposium on Foundations of Software Engineering.*

[8] Bird, C., Bachmann, A., Aune, E., Duffy, J., & Devanbu, P. T. (2009). Fair and Balanced? Bias in bug-fix datasets. *Proceedings of the 7th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering*.

[9] Bachmann, A., Bird, C., Rahman, F., Devanbu, P. T., & Bernstein, A. (2010). The missing links: Bugs and bug-fix commits. *Proceedings of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering*.

[10] Nguyen, T. H. D., Adams, B., & Hassan, A. E. (2010). A case study of bias in bug-fix datasets. *Proceedings of the 2010 17th Working Conference on Reverse Engineering*.

[11] Panjer, L D. (2007). Predicting eclipse bug lifetimes. *Proceedings of the International Workshop on Mining Software Repositories*.

[12] Giger, E., Pinzger, M., & Gall, H. C. (2010). Predicting the fix time of bugs. *International Workshop on Recommendation Systems for Software Engineering*.

[13] Zhang, H., Gong, L., & Versteeg, S. (2013). Predicting bug-fixing time: An empirical study of commercial software projects. *Proceedings of the International Conference on Software Engineering*.

[14] Weiss, C., Premraj, R., Zimmermann, T., & Zeller, A. (2007). How long will it take to fix this bug?. *Proceedings of the International Workshop on Mining Software Repositories*.

[15] Anbalagan, P., & Vouk, M. (2009). On predicting the time taken to correct bug reports in open source projects.

[16] Čubranić, D. (2004). Automatic bug triage using text categorization. *Proceedings of the International Conference on Software Engineering and Knowledge Engineering*.

[17] Anvik, J., Hiew, L., & Murphy, G. C. (2006). Who should fix this bug?. *Proceedings of the 28th International Conference on Software Engineering*.

[18] Anvik, J. (2006). Automating bug report assignment. *Proceedings of the International Conference on Software Engineering*.

[19] Baysal, O., Godfrey, M. W., & Cohen, R. (2009). A bug you like: A framework for automated assignment of bugs. *Proceedings of the IEEE International Conference on Program Comprehension*.

[20] Bhattacharya, P., & Neamtiu, I. (2010). Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging. *Proceedings of the IEEE International Conference on Software Maintenance*.

[21] Tamrawi, A. (2011). Fuzzy set-based automatic bug triaging (NIER track). *Proceedings of the International Conference on Software Engineering*.

[22] Park, J., Lee, M., Kim, J., Hwang, S., & Kim, S. (2016). Cost-aware triage ranking algorithms for bug reporting systems. *Knowledge and Information Systems, 48(3)*, 679-705.

[23] Baysal, O., Godfrey, M. W., & Cohen, R. (2009). *Proceedings of the* I*EEE 17th International Conference on Program Comprehension: A Framework for Automated Assignment of Bugs.* 297-298.

[24] Aljarah, I., Banitaan, S., Abufardeh, S., Jin, W., & Salem, S. (2011). Selecting discriminating terms for bug assignment: A formal analysis. *Proceedings of the International Conference on Predictive Models in Software Engineering*.

[25] Matter, D., Kuhn, A., & Nierstrasz, O. (2009). Assigning bug reports using a vocabulary-based expertise model of developers. *Proceedings of the 6th International Working Conference on Mining Software Repositories*.

[26] Servant, F., & Jones, J. A. (2012). Whosefault: automatic developer-to-fault assignment through fault localization. *Proceedings of the International Conference on Software Engineering*.

[27] Shokripour, R., Anvik, J., Kasirun, Z. M., & Zamani, S. (2013). Why so complicated? Simple term filtering and weighting for location-based bug report assignment recommendation. *Proceedings of the 2013 10th Working Conference on Mining Software Repositories.*

[28] Kevic, K., Muller, S. C., Fritz, T., & Gall, H. C. (2013). Collaborative bug triaging using textual similarities and change set analysis. *Proceedings of the International Workshop on Cooperative & Human Aspects of Software Engineering*.

[29] Bettenburg, N., Just, S., Adrian Schröter, Weiss, C., Premraj, R., & Zimmermann, T. (2008). What makes a good bug report?. *Acm Sigsoft International Symposium on Foundations of Software Engineering*.

[30] Bettenburg, N., Just, S., Schr?Ter, A., Wei?, C., Premraj, R., & Zimmermann, T. (2007). *Proceedings of the 2007 OOPSLA Workshop on Eclipse Technology Exchange*.

[31] Hooimeijer, P., & Weimer, W. (2007). Modeling bug report quality. *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*.

[32] Xie, J., Zhou, M., & Mockus, A. (2013). Impact of Triage: A study of mozilla and gnome. *Proceedings of the 2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*.

[33] Breu, S., Premraj, R., Sillito, J., & Zimmermann, T. (2010). Information needs in bug reports: Improving cooperation between developers and users. *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work*.

[34] Yusop, N. S. M., Schneider, J. G., Grundy, J., & Vasa, R. (2017). What influences usability defect reporting? — A survey of software development practitioners. *Proceedings of the Software Engineering Conference*.

[35] Alenezi, M., Magel, K., Banitaan, S. (2013). Efficient bug triaging using text mining. *J Softw*.

[36] Lamkanfi, A., Demeyer, S., Giger, E., & Goethals, B. (2010). Predicting the severity of a reported bug. *Proceedings of the 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*.

[37] Lamkanfi, A., Demeyer, S., Soetens, Q. D., & Verdonck, T. (2011). Comparing mining algorithms for predicting the severity of a reported bug. *Proceedings of the 15th European Conference on Software Maintenance and Reengineering*.

[38] Menzies, T., & Marcus, A. (2015). Automated severity assessment of software defect reports. *Proceedings of the IEEE International Conference on Software Maintenance*.

[39] Zhang, T., Yang, G., Lee, B., & Chan, A. T. S. (2015). Predicting severity of bug report by mining bug repository with concept profile. *ACM Symposium on Applied Computing*.

[40] Davies, R. (2012). Using bug report similarity to enhance bug localisation. *Reverse Engineering*.

[41] Jindal, R., Malhotra, R., & Jain, A. (2016). Prediction of defect severity by mining software project reports. *International Journal of System Assurance Engineering & Management*, 1-18.

[42] Lamkanfi, A., Demeyer, S. (2013). Predicting reassignments of bug reports — An exploratory investigation. *Proceedings of the European Conference on Software Maintenance and Reengineering*.

[43] Herraiz. (2008). Towards a simplification of the bug report form in eclipse. *Proceedings of the International Working Conference on Mining Software Repositories*.

[44] Wu, L, L., Xie, B., Kaiser, G, E. *et al*. (2011).BugMiner: Software reliability analysis via data mining of bug reports. *Proceedings of the International Conference on Software Engineering & Knowledge Engineering.*

[45] Xia, X., Lo, D., Wen, M., Shihab, E., & Zhou, B. (2014). *An Empirical Study of Bug Report Field Reassignment*.

[46] Rastkar, S., Murphy, G. C., & Murray, G. (2010). Summarizing software artifacts: a case study of bug reports. *Proceedings of the ACM/IEEE International Conference on Software Engineering*.

[47] Antoniol, G., Ayari, K., Penta, M. D., Khomh, F., & Yann-Gaël, G. (2008). Is it a bug or an enhancement?: A text-based approach to classify change requests. *Proceedings of the 2008 conference of the Centre for Advanced Studies on Collaborative Research*.

[48] Pingclasai, N., Hata, H., & Matsumoto, K. I. (2013). Classifying bug reports to bugs and other requests using topic modeling. *Proceedings of the 2013 20th Asia-Pacific Software Engineering Conference.*

[49] Herzig, K., Just, S., & Zeller, A. (2013). It's not a bug, it's a feature: How misclassification impacts bug prediction. *Proceedings of the International Conference on Software Engineering*.

[50] Zanetti, M. S., Scholtes, I., Tessone, C. J., & Schweitzer, F. (2013). Categorizing bugs with social networks: A case study on four open source software communities.

[51] Canfora, G., & Cerulo, L. (2006). Supporting change request assignment in open source development. *ACM Symposium on Applied Computing*.

[52] Hiew, L. (2006). *Assisted Detection of Duplicate Bug Reports.* The University of British Columbia, Vancouver.

[53] Runeson, P., Alexandersson, M., & Nyholm, O. (2007). Detection of duplicate defect reports using natural language processing. *Proceedings of the International Conference on Software Engineering*.

[54] Jalbert, N., & Weimer, W. (2008). Automated duplicate detection for bug tracking systems. *Proceedings*

of the IEEE International Conference on Dependable Systems & Networks with FTCS & DCC.

[55] Sureka, A., & Jalote, P. (2010). Detecting duplicate bug report using character n-Gram-based features. *Proceedings of the Asia Pacific Software Engineering Conference*.

[56] Sun, C., Lo, D., Wang, X., Jiang, J., & Khoo, S. C. (2010). *Proceedings of the 32nd ACM/IEEE international Conference on Software Engineering*.

[57] Sun, C., Lo, D, Khoo, S. C., & Jiang, J. (2011). Towards more accurate retrieval of duplicate bug reports. *Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering*.

[58] Tian, Y., Sun, C., & Lo, D. (2012). Improved duplicate bug report identification. *Proceedings of the European Conference on Software Maintenance & Reengineering*.

[59] Nguyen, A. T., Nguyen, T. T., Nguyen, T. N., Lo, D., & Sun, C. (2012). *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*.

[60] Banerjee, S., Cukic, B., & Adjeroh, D. (2012). Automated duplicate bug report classification using subsequence matching. *Proceedings of the IEEE International Symposium on High-assurance Systems Engineering*.

[61] Liu, K., Tan, H. B. K., & Chandramohan, M. (2012). *Proceedings of the ACM SIGSOFT 20th international Symposium on the Foundations of Software Engineering*.

[62] Zhou, J., & Zhang, H. (2012). Learning to rank duplicate bug reports. *Proceedings of the ACM International Conference on Information and Knowledge Management*.

[63] Podgurski, A., Leon, D., Francis, P., Masri, W., Minch, M., & Sun, J., *et al*. (2003). *Proceedings of the 25th International Conference on Software Engineering.*

[64] Wang, X., Zhang, L., Xie, T., Anvik, J., & Sun, J. (2008). An approach to detecting duplicate bug reports using natural language and execution information. *Proceedings of the 2008 ACM/IEEE 30th International Conference on Software Engineering.*

[65] Song, Y., Wang, X., Xie, T., Zhang, L., & Mei, H. (2010). JDF: Detecting duplicate bug reports in Jazz. *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*.

[66] Lerch, J., & Mezini, M. (2013). Finding duplicates of your yet unwritten bug report.

[67] Feng, L., Song, L., Sha, C., & Gong, X. (2013). Practical duplicate bug reports detection in a large web-based development community.

[68] Hindle, A., Alipour, A., & Stroulia, E. (2016). A contextual approach towards more accurate duplicate bug report detection and ranking. *Empirical Software Engineering, 21*(*2*), 368-410.

[69] Kim, S., Zimmermann, T., Nagappan, N. (2011). Crash graphs: an aggregated view of multiple crashes to improve crash triage. *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE.

[70] Dang, Y., Wu, R., Zhang, H., Zhang, D., & Nobel, P. (2012). ReBucket: A method for clustering duplicate crash reports based on call stack similarity. *Proceedings of the 34th International Conference on Software Engineering*.

[71] Tian, Y., David, L. O., & Sun, C. (2012). Information retrieval based nearest neighbor classification for fine-grained bug severity prediction.

[72] Zhang, T., Chen, J., Yang, G., Lee, B., & Luo, X. (2016). Towards more accurate severity prediction and fixer recommendation of software bugs. *Journal of Systems and Software*.

[73] Gao, Yan., Yang, Chun-hui., Liang, Li-xin. (2017). Software defect prediction based on geometric mean for subspace learning. *Proceedings of the IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference*.

[74] Yu, L., Tsai, W. T., Zhao, W., & Wu, F. (2015). Predicting defect priority based on neural

networks. *Advanced Data Mining & Applications-International Conference*.

[75] Kanwal, J., & Maqbool, O. (2012). Bug prioritization to facilitate bug report triage. *Journal of Computer Science and Technology (English Language Edition), 27*(2), 397-412.

[76] Hu, H., Zhang, H., Xuan, J., & Sun, W. (2014). Effective bug triage based on historical bug-fix information. *Proceedings of the IEEE International Symposium on Software Reliability Engineering*.

[77] Lin, Z., Shu, F., Yang, Y., Hu, C., & Wang, Q. (2009). An empirical study on bug assignment automation using Chinese bug data. *Proceedings of the Third International Symposium on Empirical Software Engineering and Measurement*.

[78] Xuan, J., Jiang, H., Ren, Z., Yan, J., & Luo, Z. (2017). Automatic bug triage using semi-supervised text classification.

[79] Jeong, G., Kim, S., & Zimmermann, T. (2009). *Proceedings of the Joint Meeting of the European Software Engineering Conference and the ACM Sigsoft Symposium on the Foundations of Software Engineering*.

[80] Bhattacharya, P., & Neamtiu, I. (2010). Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging. *Proceedings of the IEEE International Conference on Software Maintenance*.

[81] Tamrawi, A. (2011). Fuzzy set-based automatic bug triaging (NIER track). *Proceedings of the International Conference on Software Engineering*.

[82] Xie, J., Zheng, Q., Zhou, M., & Mockus, A. (2014). Product assignment recommender. *Companion Proceedings of the International Conference on Software Engineering*.

[83] Guo, P. J., Zimmermann, T., Nagappan, N., & Murphy, B. (2011). "Not my bug!" and other reasons for software bug report reassignments. *Proceedings of the ACM Conference on Computer Supported Cooperative Work*.

[84] Zou, W., Hu, Y., Xuan, J. *et al*. (2011). Towards training set reduction for bug triage. *Proceedings of the Annual IEEE International Computer Software and Applications Conference*.

[85] Jindal, R., Malhotra, R., & Jain, A. (2015). Mining defect reports for predicting software maintenance effort.

[86] Yang, Chen-zen., Hou, Chun-chi., Kao, W. C. Chen, Wei, C., Xiang, I. (2012). An empirical study on improving severity prediction of defect reports using feature selection. *Proceedings of the Asia-pacific Software Engineering Conference*.

[87] Gromova, A. (2017). Using cluster analysis for characteristics detection in software defect reports.

[88] Liu, W., Wang, S., Chen, X., & Jiang, H. (2018). Predicting the severity of bug reports based on feature selection. *International Journal of Software Engineering and Knowledge Engineering, 28*(*4*), 537-558.

Wenjie Liu received his master's degree in software engineering from Dalian University of Technology in 2006. At present, he is a faculty member in the School of Software Technology, Dalian University of Technology. And has been the main person in charge of the IT Department. He holds the title of senior network administrator in China. He has over 12 years of experience in software engineering and network engineering. He wrote two books and 22 journal articles. His research interests include software engineering, network engineering, and in particular software metrics and quality models. He has published many related papers.