A Novel Bug Report Extraction Approach

Tao Lin[1(✉)], Jianhua Gao[1], Xue Fu[1], and Yan Lin[2]

[1] Department of Computer Science and Technology, Shanghai Normal University, Shanghai 200234, China
l.t@acm.org, jhgao@shnu.edu.cn, fuxuee@hotmail.com

[2] Department of Information Systems and Operations Management, The University of Auckland, Auckland 92019, New Zealand
ylin688@aucklanduni.ac.nz

Abstract. There are more and more bug reports in software. Software companies and developers invest a large number of resources into the dramatic accumulation of reports. We introduce Bayes classifier into bug reports compression, which is the first effort in the literature. For this purpose, the vector space model as well as some conventional text mining values, such as tf-idf and chi-squared test, are designed to collect features for bug reports. The experiment proves that bug reports extraction by using Bayes classifier is outperformance to the method based on SVM through the evaluation of ROC and F-score.

Keywords: Bug report · Naïve Bayes classifier · Bug extraction · Tf-idf · Text mining

1 Introduction

The past decade witnessed a significant enhancement of software engineering. However, developers are increasingly bewildered by a great number of bug reports accumulated rapidly day by day. Admittedly, there are some excellent code management integrated development environments, such as Eclipse, and WingIDE. Besides some state of art tools can assist developers planning software projects, such as Microsoft Project, ProjectLibre and Openproj. On the other hand, it is obvious to note that there is little or no research on how to extract bug report to help developers as Table 1 shows.

Admittedly, there are many researchers have studied variety of documents summarization based on distinct methods. Goyal et al. investigated context-based extraction for general documents for improving traditional ways taking no consideration on contest [1]. There are a graph-based approach for documents similarity, and summarization introduced by Mills et al. [2].

As noted above, what is our motivation to research to extract bug report? Previously, there is little attention being paid on management of bug reports. Take waterfall model, one of primitive and key model in software engineering. The waterfall model divide a software entire life cycle into five main process, namely communication, planning, modeling, construction and deployment, of which construction chiefly focus on code and test. While test are concentration on unit testing, integration testing, system testing,

G. Wang et al. (Eds.): ICA3PP 2015 Workshops, LNCS 9532, pp. 771–780, 2015.
DOI: 10.1007/978-3-319-27161-3_70

and acceptance testing. As far as we are concerned, the whole software engineering pay little attention on bug reports management. There is a possible reason for maintenance being arduous is the lack of testing. To be precise, the short board on bug reports management in software engineering.

Table 1. Some tools to insist developers in variety of aspects in software engineering

Aspects of software engineering	Assist tools
Code management	Eclipse, WingIDE
Planning project	Microsoft Project, ProjectLibre, Openproj
Bug reports extraction	?

Yet bug reports are increasingly stand a significant role in software engineering, in terms of assistance of software reuse, update, upgrade, etc. Bertram and Greenberg suggested that bug reports should be taken consideration on software team development [3]. Some researchers conducted investigation on which bug reports are more critical to the entire project, by this making determination to arrange the sequence of fixing bugs [4].

While there is no denying that software code has essential position in a project, it is not too much usefulness in future after the project. In contrast, bug reports can influence the future software development fundamentally, for example as reference and review. None the less, there are some side effects in modern software development on bug reports, mainly because there are too many reports to archive, even the project participants can easily forget and ignore some important and major bug reports, let alone other managers not involve in the software project to arrange and trim the reports. Although some software companies demand that the developers summarize the bug reports just after fixing the bug. This is not an accept method for the following two reasons. To be admitted, most developers are good at fixing bugs, there are few developers can realize which bug reports are important in the future. The second reason is that developers do not have passion and responsibility to summary bug reports, due to company not always praising the task related to bug reports extraction.

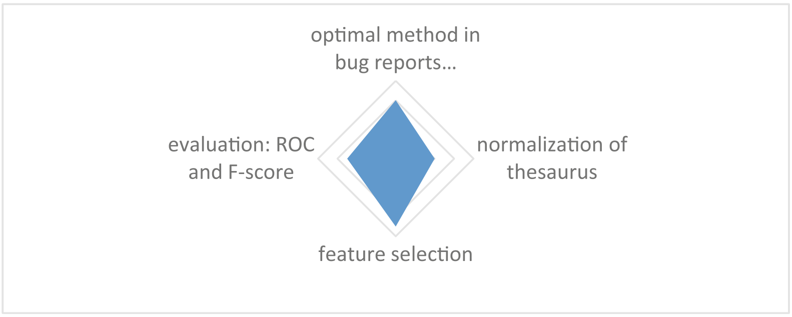
optimal method in bug reports...
evaluation: ROC and F-score
normalization of thesaurus
feature selection

Fig. 1. Four contributions in our work

Therefore, it is urgent to research automatic archive methods on bug reports. It is wise to apply pattern reorganization approach to the area.

This paper makes four contribution as following Fig. 1.

According to our contributions, this paper mainly divided four sections:

In the first section, we briefly explain some core theory about naïve Bayes classier, and why this is optimal method to apply in the circumstance of bug reports.

In the second section, though we mainly focus on extractive approach, it is useful to process original bug report by abstractive approach, to be precise, we use a method of normalization of thesaurus.

In the third section, we want present the features in our classier, which is based on vector space model.

At last, we give an experiment and compute ROC curve and related evaluation values, such as precision, recall, F-score.

Though there are some work for document mining [5] and bug reports summarization based on support vector machine previously. This research maybe is the first time to extract bug reports by using Bayer classifier in the literature.

2 The Reason to Choose Naive Bayes Classifier

In machine learning, there are already a large number of classifiers, for instance, support vector machine and Logistic regression. In previous research, Rastkar et al. apply support vector machine methods to summary bug reports [6]. To be admitted, support vector machine method is an effective approach in many areas, such as geophysics analysis and face recognition. On the other hand, it is not a judicious and advisability way in bug report analysis for the following reasons. It is necessary to compute a great number of 'vectors' (word weight) in bug reports, by this, it is easily to depend on a tendency using a very high level vectors. Though it maybe increases the accuracy in training set, there is a marked plunge rate in test set. On the contrary to that, naïve Bayes classifier can avoid the problem of support vector machine by statistical method.

Therefore, how can we use naïve Bayes classier in bug report? Suppose the bug reports can be represented as $C_i = (C_1, C_2, C_3, \ldots, C_n)$. The attribute of bug reports, if have m attributes, can be noted as $a_i = (a_1, a_2, a_3, \ldots, a_m)$, besides the important and possessing profound historical significance bug reports are a subset of a_i. Therefore, the extraction of bug reports is a problem of posterior probability. By this we mean that according to Bayes formula, bug reports a_i is part of C_i, if and only if

$$\mathrm{P}\left(c_i|a_x\right) = \mathrm{p}\left(c_i\right)\frac{\mathrm{p}\left(a_x|c_i\right)}{\mathrm{p}(a_x)} \tag{1}$$

$P(c_i)$ is the priori probability of one specific bug report?

One of significant rational to select naïve Bayes classifier is that it is simple and convenience to avoid the terminology of involving in specific software projects bug reports, in other words, naïve Bayes classifier can be generalized to more software projects.

3 Abstract Thesaurus

It is known to us all that one of inconvenience problems on natural language is that English like most languages in the world being ambiguity. In this paper, for a better result, firstly, we intend to eliminate the ambiguity by merge different thesaurus into one. For the best consequence, we try to setup a merge Table 2.

Table 2. Part of merge table to normalization of thesaurus

Core word	Alternative word
Problem	Difficulty, drawback, issue
Fix	Tackle, arrange, solve
Good	Fine, ok, not bad
Suggestion	Proposal, proposition, submission, idea, recommendation
Agreement	Contract, arrangement, promise

4 Bug Reports Features

One of the major task in this work is classification process, which includes word segmentation, abstract thesaurus, feature selection, and vector space model as following graph (Fig. 2).

In this section, we introduce vector space model firstly, then what we intend to emphasize is feature selection, which is based on text mining methods, but maybe is the first time use in bug reports extraction.

4.1 Vector Space Model

There are a large number of text mining methods in the literature, such as latent semantic analysis, probabilistic latent semantic analysis, latent dirichlet allocation, and the correlated topic model [7]. None the less, it is vector space model suggested by Salton et al., one of the most popular and widely used, which is possible and optimal for preprocessing of bug reports [8].

In vector space model, bug reports can be presented as vector. Every bug report can be regarded as a two-valued feature vectors, as follows,

$$\mathrm{b} = \{(t_1, w_1), (t_2, w_2), \ldots, (t_n, w_n)\} \tag{2}$$

t_i is the feature item, w_i is the weight of corresponding t_i, and n is the length feature space. In our work, the length feature space in bug reports is definite. Therefore, for the simplification of related work, the above formula can be simplified as

$$\mathrm{b} = \{w_1, w_2, \ldots, w_n\} \tag{3}$$

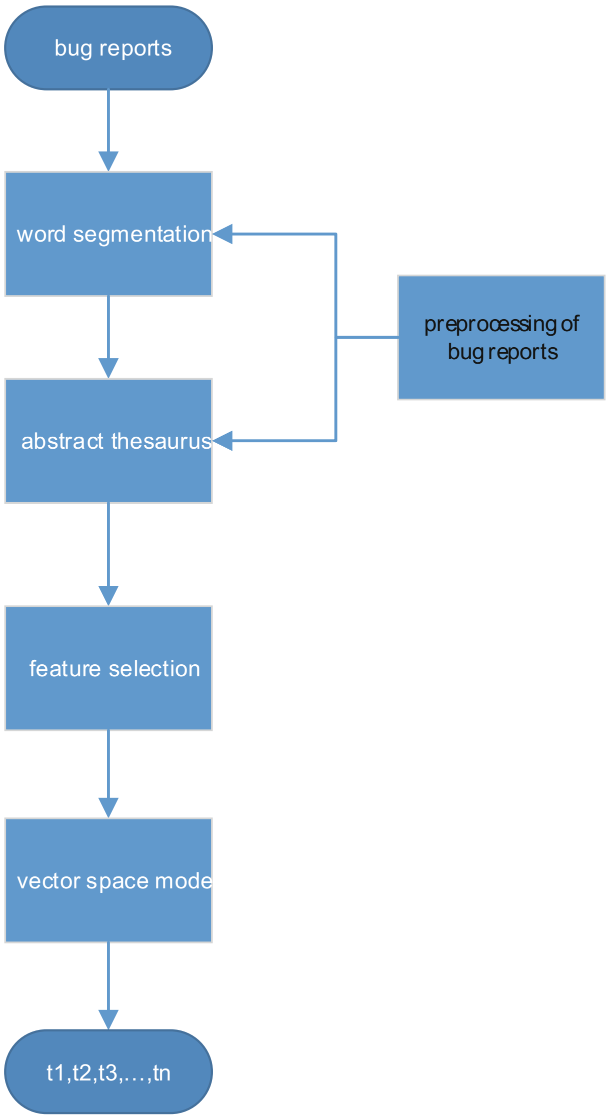

Fig. 2. Bug reports classification process

By this, it is convenient to get every bug report's vector space presentation by computing every items weight.

4.2 Term Frequency–Inverse Document Frequency

Though there are some extensive research on text feature extraction. As regards bug report, we mainly extract two main features. The first is key words, as we just put it in

Sect. 3. Then, it is obvious that the core words should be selected as features. While, we try to use some more objective methods. We compute term frequency–inverse document frequency (tf-idf), which is a value evaluate a word whether or not having key position in the total texts, and this value comprise two independent values term frequency and document frequency. Term frequency in bug reports corpus can be defined as

$$\text{term frequency} = \frac{\textit{specific word appeared in total bug reports corpus}}{\textit{total words in bug reports copus}} \tag{4}$$

Document frequency in bug reports corpus can be defined as

$$\text{document frequency} = \log \frac{\textit{number of reports include specific word}}{\textit{total number of reports in corpus}} \tag{5}$$

In our opinion, the higher document frequency, the more value in the specific bug reports. On the other hand, it is possible that the lower document frequency includes more information, in other words, it is should be preserved in bug reports extraction. Consequently, in regard of the above analysis, tf-idf is trade-off.

tf-idf is the multiple of tf and idf, in other words,

$$\text{tf} - \text{idf} = \text{tf*idf} \tag{6}$$

Therefore, for instance, there are 20458 words in the corpus in our experiment. ‘Problem’ appeared 73 times total. There are total 36 bug reports. And ‘problem’ appeared in 35 bug reports. Therefore, the term frequency is 0.003568, and document frequency is -0.02817. From above, we can compute the term frequency–inverse document frequency of ‘problem’ is $-1.005\ e^{-4}$.

4.3 Chi-Squared Test

Besides term frequency–inverse document frequency, we try to introduce chi-squared test (χ^2 test) into bug reports core words features extraction. To be precise, χ^2 test in debug reports show the relevance of one specific and the extraction from debug reports.

$$\chi^2\,(\text{m}, \text{e}) = \frac{T\,(AD - BC)^2}{(A + C)\,(B + D)\,(A + B)\,(C + D)} \tag{7}$$

T is the total bug reports in corpus. A is the number of extraction including specific word m. B is the number of exclude extraction including specific word m. C is the number of extraction not including specific word m. D is the number of exclude extraction not including specific word m. Obviously, $\chi^2\,(\text{m}, \text{e}) = 0$ when feature m and extraction are independent from each other. On the other hand, the larger $\chi^2\,(\text{m}, \text{e})$, the more possibility including the word m in extraction. We use the ‘problem’ as example. We suppose every sentence as a dependent text. In the corpus as the experiment, there are total N = 2361 sentences in bug reports. There are 49 sentences including ‘problem’ (A = 59) in extraction. There are 24 sentences including ‘problem’ not in extraction (B = 14). There are

465 sentences in extraction, in which there are C = 406 sentences not including 'problem'. D equals 1882, which means there are 1882 sentences not including 'problem' not in extraction. Therefore the 'problem' of χ^2 (m, e) = 66454.31.

In bug reports, χ^2 test hold attention on the relation specific words between the extractions of bug reports.

4.4 Information Gain

The third indicator to determine a word whether or not being selected as core words is the value of information gain, which means the specific words have how much information quality.

$$\mathrm{IG}\,(\mathrm{t}) = -\sum_{i=1}^{2} p\left(c_i\right) * log_2 p\left(c_i\right) + p\,(t)\sum_{i=1}^{2} p\left(c_i|t\right) * log_2 p\left(c_i|t\right) + p\,(\bar{t})\sum_{i=1}^{2} p\left(c_i|\bar{t}\right) * log_2 p\left(c_i|\bar{t}\right) \tag{8}$$

Information gain can note specific words bring how much information to the bug reports extraction. The more information, the more vital to the specific words in extraction. In bug reports extraction, information gain of every words is computed, throwing the words, which is lower the specific threshold. By this we mean that it is the words above the threshold can be regarded as features.

4.5 Sentence Complication

The other features is sentence complication. As we investigated, the higher complication sentence include much more crucial information than simple sentences, especially some sentences in the bug report only have one to three words, therefore, sentences bearing this characteristic should be excluded from the extraction. Take coups in our experiment for example. Some sentences, like 'Good point.', 'But go ahead', and 'How does it sound?' being little message, it is eligible not in extraction.

5 Experiment

In this paper, we use the Sarah Rastkar et al' bug report corpus.[1] One of a good reason to use this corps is that there is an annotation for this bug reports. By this we can train a naïve Bayes classifier based on Python Textblob.[2] It is the convention that we need to compute true positive rate and false positive rate to plot receiver operator characteristic curve, ROC and compute the Area-under-the-ROC curve, AUC.

[1] www.cs.ubc.ca/cs-research/software-practices-lab/projects/summarizing-software-artifacts, verified 2015/09/04.

[2] http://textblob.readthedocs.org/en/dev/, verified 2015/09/04.

6 Receiver Operator Characteristic Curve

First, we need to compute true positive rate.

$$\text{true positive rate} = \frac{\textit{bug sentences in classifier from annotation}}{\textit{total sentences in annotation}} \tag{9}$$

Then, we have to calculate false positive rate.

$$\text{false positive rate} = \frac{\textit{bug sentences in classifier not from annotation}}{\textit{total sentences in bug report not in annotation}} \tag{10}$$

At last, we can plot the ROC as shown in Fig. 3, and compute AUC.

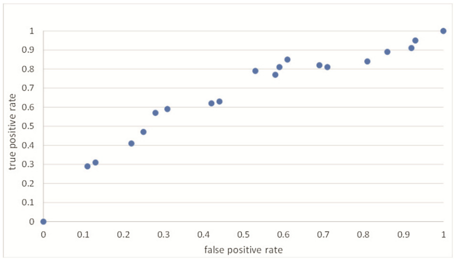

Fig. 3. ROC for Bayes classifiers

According to ROC, Area-under-the-ROC curve is 0.711, which is competitive to support vector machine, while Bayes classifier is much more less dimensionality

6.1 F-Score

We further analyses the effectiveness of our Bayer classifier by compute some standard value, namely, precision, recall, and F-score.

$$\text{precision} = \frac{\textit{bug sentences in classifier from annotation}}{\textit{total senctences in classifier}} = 0.61$$

$$\text{recall} = \frac{\textit{bug sentences in classifier from annotation}}{\textit{total sentences in annotation}} = 0.33$$

In fact, the value of recall equals true positive rate.

Because we cannot guarantee the two value both high, so it is common to use F-score to determine the quality of classifiers.

$$\text{F} - \text{score} = 2 * \frac{\textit{precision} * \textit{recall}}{\textit{precision} + \textit{recall}} = 0.43$$

There are some problem hard to solve to increase F-score value, for the inherent factors of natural language, such as the idea of modification, revocation. For example, one of developers in corpus said: “Once I started messing around with what is ticked, the problem went away.” But later, he made supplementary: “Now I am also unable to reproduce it.” These two examples are typical contradiction presentation in natural language. However, neither support vector machine classifier nor naïve Bayes classifier can make wise and right decision on this situation. As far as we are concerned, on the one hand, it is a necessity to conduct much more research on classifier. On the other hand, it is requisite that establish some essential standard in bug reports on the condition that not to confine developers creative idea and effective team cooperation.

7 Summary

Developers need excellent bug reports to enhance software development. However, there is little and no research in this area until now, except some complicated methods using support vector machine.

In this paper, may be initiated, we present a simple, but effective Bayes classifier, which is competitive to support vector machine classifier. One of main threats in this research is the corpus which we selected as experiment. To be precise, the corpus is not large enough, the annotation in the corpus is subjective, rather than objective. In other words, it is not easy to compare any bug reports classifier whether or not effectiveness in an objective standard. In the future work, it is one of our main target to setup a relative large enough software bug reports corpus. In addition, the bug reports in the corpus coming from Eclipse Platform, Gnome, Mozilla and KDE, which all are part of open-source software projects. By this we mean that we wonder how our classifier’s effectiveness in commercial environment. However, this defect scarcely to solve straightforward, for almost all commercial software bug reports are confidential.

There are two main areas which need further research. The first is that we intend to design an automatic feature abstract classifier to eliminate thesaurus specifically for bug reports. Besides, there is possible more advantages if combining various classifiers to extract the bug reports, for instance, Bayes classifier and Decision Tree combination.

Acknowledgments. This work was supported by the National Science Foundation of China, No. 61073163, and Enterprise Innovation Special Fund of Shanghai Municipal Commisiion of Economy and Informatization, China, No. CXY-2013-88.

References

1. Goyal, P., Behera, L., McGinnity, T.M.: A context-based word indexing model for document summarization. IEEE Trans. Knowl. Data Eng. **25**, 1693 (2013)
2. Mills, M.T., Bourbakis, N.G.: Graph-based methods for natural language processing and understanding—a survey and analysis. IEEE Trans. Syst. Man, Cybern. Syst. **44**, 59 (2014)
3. Bertram, D., Greenberg A.V.S.: Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams. In: Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW 2010), vol. 291 (2010)

4. Alenezi, M., Banitaan, S.: Bug reports prioritization: which features and classifier to use? In: 12th International Conference on Machine Learning and Applications (ICMLA), vol. 2, p. 112, Miami, FL (2013)
5. Kastner, C., Dreiling, A., Ostermann, K.: Variability mining: consistent semi-automatic detection of product-line features. IEEE Transactions on Software Engineering **40**, 67 (2014)
6. Rastkar S., Murphy G.C., Murray G.: Automatic Summarization of Bug Reports. IEEE Transactions on Software Engineering. 40. 366 (2014)
7. Sangno, L., Baker, J., Song, J., Wetherbe, J.C.: An empirical comparison of four text mining methods. In: 43rd Hawaii International Conference on System Sciences (HICSS), vol. 1, Honolulu, HI (2010)
8. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM **18**, 613–614 (1975)