Exploring the Influence of Time Factor in Bug Report Prioritization

Zhengjie Xu, Tieke He, Weiqiang Zhang, Yabin Wang, Jia Liu*, Zhenyu Chen
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
*liujia@nju.edu.cn

Abstract

Time factor has been widely applied into a wide range of data mining areas, such as social network and information retrieval. The main idea of taking time factor into consideration is that human activities may have some relations to time pattern. However, little attention has been pulled on the study of time factor in the area of software engineering. In this paper, we endeavour to explore to what extent time factor affects the prioritization of bug reports, a specified while important task in software engineering. Specifically, we test four time factors that may have some influence on this task, which are time of day, normal time, day of week, and days to major version. After the validation of relatedness of all these factors, we conduct an extensive set of experiments on two datasets to verify the effectiveness of these factors. The experimental results demonstrate that we can effectively improve the results by metrics of both Precision *and* Recall*, on two classical models, i.e., the* SVM *model and* Naive Bayes *model.*

Keywords: Time factor; bug report priorization; SVM

1. Introduction

Error or so-called bug reports are quite common during the process of software development and maintenance, which are critical to the stability as well as security of the software. Users, developers or any members who will use the software may submit a bug report. Thus, it is quite important for the developers to judge and analyze the bugs. A bug repository is then created to store great numbers of bug reports. Bugzilla, a bug repository came out during the development of Mozilla, is now widely used in most open-source software development. However, without being analyzed, bug reports are just none of use for the developers. Therefore, analyzing whether the bugs are valid or not, correct or not, important or not even unique or not becomes a necessary process. This is so-called bug triaging. For a bug triager, if he analyzes the bug reports one by one in time sequence, he will be overwhelmed by lots of data and thus, some important bugs may be ignored, which does great harm to the whole maintenance of the software. Hence, a way to prioritize the bug reports becomes an increasingly need for the developers.

In most researches in prioritization, adopted factors such as the textual content of the bug report, the author and so on are quite common. For example, the DRONE [12] framework takes textual, author, related-report, severity and product as their judging features. By judging these features, we can prioritize the bug reports to some extent. Kremenek et al. [9] used the successful as well as the failed checks found by the bug-finding tools to prioritize the bugs. Tools' analyzing decisions and Z-ranking scheme are used to rank the bugs' priority. These factors are basic and fundamental which are used widely in bug prioritization.

However, there is a sort of factors that most researchers did not notice or not pay too much attention to in the area of software engineering. It is the time factor, which includes the time the bug report is handed in as well as the existing time of the bug report. All the researches tend to ignore the factor of time, which, we believe, is cursory. Other subjects have given us the lesson that time factor can be critical and should be put emphasis on: in economy, time influences the value of money; in the subject of journalism and communication, time also affects the value of the news. As we can see, time factor is critical in many areas and therefore, a potential feature in deciding the priority of the bug report. Inspired from these thoughts, we have raised the question whether time factor has an impact on the bug prioritization.

On the other hand, time-based bug prioritization is difficult to accomplish. There are lots of challenges we have to face. Unlike other factors, such as author, textual content and so on, time factor is not as clear as those features and not so easy to extract from the bug report. Meanwhile, as almost none of the current prioritization model includes time factor, how to merge the time factor into the model becomes a problem. These are two of the basic problems we meet.

To solve this problem, we first adopt significance test in judging the relevance between time factor and bug reports' priority. This part of experiment will show the relevance

DOI reference number: 10.18293/SEKE2016-162

roughly, which determines whether time will affect the priority of bug reports or not. Afterwards, we have made a contrast experiment on to what extents time factor can influence the priority. A constructed model in past researches will be used in the experiment as well as the model merged with the factor of time. By this means, we can easily tell the effects of merging time factor into a prioritization model.

Briefly, we process the time factors as follows. 1) For the time of day (*TOD*) factor, we divide a weekday into 24 parts by hour, and fit every report into them accordingly. 2). For the normal time (*NT*) factor, we first aggregate every reporters' reporting time during the day, and then find his/her normal reporting time, after which a 0 or 1 is assigned to this factor that indicates a report is in normal time (0) or not (1). 3). For the day of week (*DOW*) factor, we divide a week into six parts, i.e., Monday to Friday, together with the Weekend, and then a number is used to label each report, e.g., 0 for Monday, and so on so forth. 4). And for the days to major version (*DTM*), we mainly mean the reporting days after a major version of the related project, and in this implementation, we roughly divide the time length by weeks, i.e., within one week after a major version, between one week and two weeks, and so on so forth, and reports that are over 8 weeks are gathered into one class, following the above style, we assign a number to each report with 0, 1 and so on.

The main contributions of this article are summarized as follows.

1. We propose a new feature, time factor, to examine the priority of the bug raised by the reports. Past researches have only considered factors on other dimensions but failed to figure out the importance of time factor in prioritization.

2. We examine how time factors affect the prioritization, and tell the extents of the optimization by adding the time factors to the model.

3. We have experimented our solution on huge numbers of bug reports to determine the priority of the bugs. The result shows that our solution can significantly improve the correctness and effectiveness of prioritization.

The rest of this paper is organized as follows. Section 2 presents the background and some relate work, the intuitions of the proposed time factors is discussed in Section 3, and the comparing experiments are shown in Section 4, along with the discussion. Then finally, we conclude our work in Section 5.

2 Background and Related Work

Bug triage is needed in a bug repository to examine all the bugs reports entered into the repository. There are several features for the triagers to examine. Basically, whether the bug report is duplicate or not will be examined first. Then usually the validity of the bug report will be checked. The basic purpose of these judgments is to remove all the potential bug reports that are unnecessary to be resolved. After examining these features, triagers will take other features in the bug report to decide the severity and priority or change the former ones of the bug report in order to ensure that the most important bugs will be resolved quickly and well enough. Therefore, prioritization is meaningful in bug triage.

As we are going to prioritize the bug reports by judging their features, we decide to use two classification algorithms to build up our model in two different ways. The two algorithms are Support Vector Machine and Nave Bayes.

Support Vector Machines build non-linear models from training sets and are especially suitable for text classification. For instance, for two different categories in a plane, what SVM does is to draw a graph to separate these two categories as far as possible. The model it builds assigns new entities into one category or the others, which is a so-called non-probabilistic binary linear classifier. For prioritization, we have 5 classes to categorize (from P_1 to P_5), which means this is a multi-class classification. Naive Bayes, on the other hand, is just a simple probabilistic based on the Bayes' theorem. All the models are based on the hypothesis that every feature is independent from others. As the Bayes' theorem considers the probability of an event based different parameters or conditions, Naive Bayes simply uses this theorem to classify different categories according to several features given by the training set. Also, Naive Bayes has a good compatibility of classifying the textual information.

Since bug reports prioritization is put great emphasis on and studied a lot, what we have to do is to try to advance the precision of the prioritization. After reviewing the researches on prioritization, we find that most researchers neglect one of the factors that we are interested in. That is the time factor. Many researchers, however, take it unimportant and think that it is no use in predicting the prioritization. We, on the contrary, opposed to this idea and supposed that time factor can be critical to the prioritization, especially according to the influence of time on other factor. In the study of social network, time is an important feature to be considered. Since human activities are immensely influenced by the time, time should not be ignored. Similar to this thought, since bug reports are submitted by people who will be easily influenced by the factor of time, we think that the severity or the priority of the bug report is relevant to the time factor. Thus, we raised this research to find the relationship

between time factor and prioritization.

According to several researches made on bug triage, many attributes of the bug reports are used to determine the severity and priority of them. For example, Cubranic [5] has worked on recommending the bug reports according to their descriptions. By using the text categorization, they can decide which developer is responsible for a certain bug. Ahsan et al. [1] uses other features such as the titles and author of the bug reports in the classifier. These factors help the prediction become more accurate. Also, different measures for classification are taken to accomplish the goal. Kanwal et al. [7] used SVM and Naive Bayes to classify the bug reports and compare two different ways for their efficiency and precision. Anvik et al. [2, 3, 4] develops different classification models due to several machine learning techniques. After comparing each of the models, they gave out a brief overview of the advantages and disadvantages of using these models.

Also, there are many related works in studying the influence of time factor on other areas. For instance, Koutsonikola et al. [8] proposes a clustering framework which groups users according to their preferred topics and the time locality of their tagging activity. By this means, they can reveal the topic-domain of users interests and significantly contributes in a profile construction process. Sutton et al. [11] assigns credit by means of the difference between temporally successive predictions. They prove their convergence and optimality for special cases and relate them to supervised-learning methods. Wang et al. [13] presents an LDA-style topic model that captures not only the low-dimensional structure of data, but also how the structure changes over time, showing improved topics, better timestamp prediction, and interpretable trends.

Some researches also do a lot in the field of machine learning, especially in SVM and Naive Bayes which we use in our research. Joachims et al. [6] did a lot in making large scale SVM learning practical. It presents algorithmic and computational results developed for SVMlight V2.0, which make large-scale SVM training more practical. The results give guidelines for the application of SVMs to large domains. McCallum et al. [10] simply empirically compared performance of two different ways of classification on five text corpora and shows the advantage and disadvantage of different Naive Bayes models, which gave us ideas about which way to choose for modeling the prioritization of the bug reports.

3 Intuitions

In this research, we divide the time factor of the bug reports into 4 different parts, i.e., *TOD*, *NT*, *DOW* and *DTM*. At first we chose these factors basically from our experience and intuition. Then we made several observations on these factors and found some evidence about them.

Intuition 1: the severity or the priority is related to the time of the day.

This thought comes from our own experience, when we were suddenly called late in the night to solve an important bug. This bug report was handed in the mid-night, which is a rare time period for bug reports. Through this experience, we began to notice the time when a bug report is submitted. From our observation, those bug reports handed in a rare time period are more likely to be marked as severe and important. Thus, we believe that the severity or the priority is related to the time of the day

Intuition 2: the severity or the priority is related to the time whether a reporter submit a bug report in the time period he normally do.

Similar to our intuition 1, we suppose that a reporter would have different habits of reporting a bug, e.g., normally, he would report during the morning, when we notice a report that is reported in the afternoon, there is a high probability that this bug report has a high priority.

Intuition 3: the severity or the priority is related to the day of the week.

We suppose that the day in a week is also critical to the priority of a bug report. For instance, many developers may meet the case when they have to work overtime during the weekends, basically solving a bug reported on Friday or weekends. Some time especially when holidays and festivals will make the bug report rank higher and need to be solved at once. From this observation, we believe that severity and priority is related to the day of the week, and specifically, we divide a week by Monday to Friday, and the Weekend.

Intuition 4: the severity or the priority is affected by the days related to the latest major version.

When a new version of the software is released, it is quite common to see patches being installed soon after the release. Usually, the bug reports which are submitted most closest to the release date are thought the most important, mainly because these bugs are supposed to be the most obvious and easiest to be found. Thus, those bug reports submitted soon after the new major version released are always considered necessary to solve immediately.

4 Experiment

4.1 Data

In this paper, we collected our dataset from the Trac[1] Open Source Project. It is an enhanced wiki and issue tracking system for software development projects. It uses a minimalistic approach to web-based software project management. Mainly, we collected data of two projects, the

[1] https://trac.edgewall.org

first is Trac itself, and the other is the Wordpress[2] project. And specifically, the collected bug reports for Wordpress was between $June, 2004$ and $March, 2013$, while the bug reports for Trac project was between $August, 2003$ and $July, 2013$. The statistics of these two datasets is shown in Table 1.

Table 1. Descriptive statistics of dataset

Project	# Bug reports	# Reporter	# Versions
Wordpress	23,848	6,013	18
Trac	10,416	4,701	11

From these reports, we extract several basic features, which include id, current status, title, type, reporter, owner, priority, milestone, component, severity, keywords, cc and description. Also, as we take time factor as the testing factors, other features that may be ignored by other models, such as open time, version, are also included in our features. To note, only those with status values such as 'resolved', 'closed', 'confirmed', 'fixed' or 'duplicate' will be used to train our model, others may be supplementary for our results. Figure 1 shows an example of the used bug report.

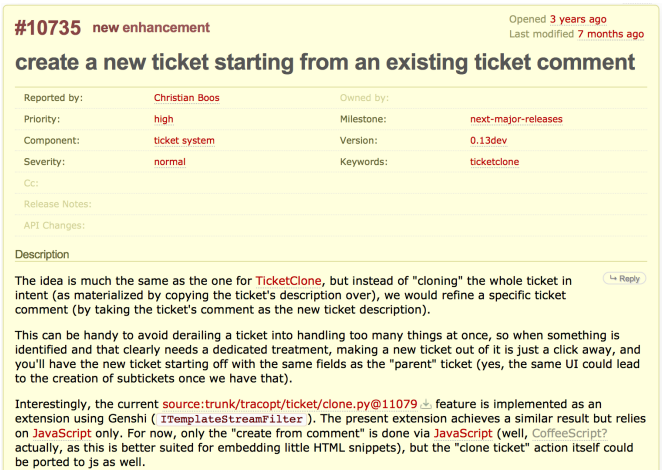

Figure 1. An example of *Trac* bug report

4.2 Metrics

We mainly use *Precision*, *Recall* to evaluate the effectiveness of our proposed approach.

Precision is defined as the ratio of relevant items that are predicted to all the predicted ones. In our experiments, relevant items are bug reports with priority that match the predicted priorities, so the *Precision* can be calculated by following equation:

$$Precision = \frac{Relevant \cap PredictedOnes}{Numberof Predicted}$$

Recall is the defined as the ratio of relevant items that are predicted to all the relevant ones. In our experiments, for one bug report, we have ground truth data of what priority of that bug report, so *Recall* can be defined as follows:

$$Recall = \frac{Relevant \cap PredictedOnes}{Numberof Relevant}$$

4.3 Significance Test

In order to find whether time factor have an influence on the prioritization of the bug reports, we performed the significance test to test the correlation between the 4 different kinds of time factors and the priority of the bug reports. Specifically, we adopt the Spearman correlation in our work, it is a non-parametric test that is used to measure the degree of association between two variables. Spearman correlation does not assume any assumptions about the distribution of the data and is the appropriate correlation analysis when the variables are measured on a scale that is at least ordinal. The formula used to calculate the Spearman correlation is as follows:

$$\rho = 1 - \frac{6 \sum {d_i}^2}{n(n^2 - 1)} \quad (1)$$

where ρ stands for the Spearman correlation, d_i is the difference between the ranks of corresponding values, and n is the number of values in each data set.

As a result, we got the following correlation for the four proposed time factors, as depicted in Table 2.

Table 2. The Spearman correlation of each time factor

Project	TOD	NT	DOW	DTM
Wordpress	0.022**	-0.014	0.006	-0.028**
Trac	-0.008	0.052**	-0.010	0.057**

** indicates the p-value < 0.01, which is significant.

From the result, we can see that time factor *DTM* is significant for both projects, and the *TOD* is effective for the Wordpress project, while the *NT* factor is effective for the Trac project. In our two chosen projects, the proposed time factor *DOW* is not significant according to the generated result.

4.4 Experimental results

Following the result of the significance testing, we evaluate to what extent these proposed time factors can affect

[2]https://wordpress.com

the task of bug report prioritization. First, we build up the model by using the basic features of the bug reports, which has been done by many researches before. Then, we build another model by incorporate the time factor as well as the basic features to see what are the differences. From the resulting data, we can know the influence of the time factor on the prioritization on each project.

For ease of understanding, in this experiment we only use the following basic features as to build the common classification models, i.e., the status, type, component, severity and reporter. And the classification models adopted in this paper are the SVM model and Naive Bayes.

In detail, a Support Vector Machine constructs a hyperplane or set of hyperplanes in a high or infinite dimensional space, then it can be used for classification. A good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. Naive Bayes is a simple technique for constructing classifiers, models that assign class labels to problem instances, represented as vectors of the feature values, where the class labels are drawn from some finite set. All different versions of Naive Bayes algorithms are based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable.

For our case, there are mainly three sets of comparisons, i.e., 1). for the Wordpress project, we need to compare between basic features and after taking in the *TOD* time factor, 2). for the Trac project, we need will compare between the basic features based models and the with the *NT* time factor included, 3). and finally, for both projects, we should compare between the basic features based models and when the *DTM* time factor is considered. And for all these comparisons, the experiments are conducted on the above-mentioned two models.

Following we present the results of our comparisons.

Table 3 depicts the comparing result for the Wordpress project on the SVM model and Naive Bayes model, between only adopting the basic features and incorporating the *TOD* time factor. From which we can see that after taking the *TOD* time factor into consideration, the *Precision* and *Recall* are both improved by about 3%.

Table 3. *TOD* **in Wordpress**

Models	***Project***	Wordpress	
	Metrics	*Precision*	*Recall*
SVM	*Basic*	0.694	0.833
	With TOD	**0.711**	**0.847**
Naive Bayes	*Basic*	0.787	0.826
	With TOD	**0.802**	**0.841**

Table 4 illustrates the comparing result for the Trac project on the SVM model and Naive Bayes model, between simply adopting the basic features and incorporating the *NT* time factor. From which we can see by taking into account the *NT* time factor, the effectiveness of prioritization is evidently improved.

Table 4. *NT* **in Trac**

Models	***Project***	Trac	
	Metrics	*Precision*	*Recall*
SVM	*Basic*	0.571	0.756
	With NT	**0.593**	**0.787**
Naive Bayes	*Basic*	0.668	0.745
	With NT	**0.689**	**0.782**

Table 5 presents the comparing result between only using the basic features and adopting the *DTM* time factors, on the two models, for both of the two projects. From the result we can conclude that for both project, and using both SVM and Naive Bayes, after bringing in the *DTM* time factor, the ability of prioritization is improved.

Table 5. DTM in both projects

Models	***Project***	Wordpress		Trac	
	Metrics	*Precision*	*Recall*	*Precision*	*Recall*
SVM	*Basic*	0.694	0.833	0.571	0.756
	With DTM	**0.726**	**0.853**	**0.597**	**0.799**
NB	*Basic*	0.787	0.826	0.668	0.745
	With DTM	**0.793**	**0.838**	**0.692**	**0.783**

4.5 Threats to validity

In our research, all the threats to the validity come basically from the experimental errors. Though we checked the process and implementation of our experiments, there are still some errors that we could not avoid. As all the data we use are triaged and prioritized by human, many severity and priority may be subjective and vary from person to person, or even worse, sometimes they could be wrong. Relatively, time factor is objective and does not have too much influence on the validity. This is the threat to the internal validity. For external validity, the threat is that all the data we use come from one source. Though we used more than ten thousand reports, we track all these bug reports from the Trac Open Source Project, which means that we could not be comprehensive to all the sources of bug reports. In the future, if time permits, we would probably track more bug reports from other sources, which can increase the reliability and sustainability of our research.

5 Conclusion

This paper introduces the time factor in the bug report prioritization. Specifically, we investigated four time factors, i.e., the time of the day, the normal time, day of week and days to the latest major version. For our specific case, the *TOD* time factor and *DTM* time factor is effective for project Wordpress, while for the Trac project the *NT* time factor and *DTM* time factor is effective. Our experimental results demonstrate that incorporating the time factors can effectively improve the *Precision* and *Recall* for bug report prioritization. In future, we will consider evaluating our proposal on more sources of data, as well as adopting more sorts of classification models.

References

[1] S. N. Ahsan, J. Ferzund, and F. Wotawa. Automatic software bug triage system (bts) based on latent semantic indexing and support vector machine. In *Software Engineering Advances, 2009. ICSEA'09. Fourth International Conference on*, pages 216–221. IEEE, 2009.

[2] J. Anvik. Automating bug report assignment. In *Proceedings of the 28th international conference on Software engineering*, pages 937–940. ACM, 2006.

[3] J. Anvik, L. Hiew, and G. C. Murphy. Coping with an open bug repository. In *Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange*, pages 35–39. ACM, 2005.

[4] J. Anvik, L. Hiew, and G. C. Murphy. Who should fix this bug? In *Proceedings of the 28th international conference on Software engineering*, pages 361–370. ACM, 2006.

[5] D. Čubranić. Automatic bug triage using text categorization. In *In SEKE 2004: Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering*. Citeseer, 2004.

[6] T. Joachims. Making large scale svm learning practical. Technical report, Universität Dortmund, 1999.

[7] J. Kanwal and O. Maqbool. Bug prioritization to facilitate bug report triage. *Journal of Computer Science and Technology*, 27(2):397–412, 2012.

[8] V. Koutsonikola, A. Vakali, E. Giannakidou, and I. Kompatsiaris. *Clustering of social tagging system users: A topic and time based approach*. Springer, 2009.

[9] T. Kremenek and D. Engler. Z-ranking: Using statistical analysis to counter the impact of static analysis approximations. In *Static Analysis*, pages 295–315. Springer, 2003.

[10] A. McCallum, K. Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

[11] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[12] Y. Tian, D. Lo, and C. Sun. Drone: Predicting priority of reported bugs by multi-factor analysis. In *2013 IEEE International Conference on Software Maintenance*, pages 200–209. IEEE, 2013.

[13] X. Wang and A. McCallum. Topics over time: a non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433. ACM, 2006.