IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

Analysis and Solutions for Bug

Akshay Khamkar[1], Aishwarya Jagtap[2], Neeraj Pattan[3], Anuja Raipure[4], Prof. Mrs. P.Y. Pawar[5]

Student, Department of Information Technology, Sinhgad Academy of Engineering, Pune, India[1,2,3,4]

Professor, Department of Information Technology, Sinhgad Academy of Engineering, Pune, India [5]

Abstract: In the recent years, as the number of lines of code has been increasing exponentially, and this has also increased the time to solve the bugs found in the source code of any software. To manage this surge in bugs, there is need to reduce the time required to solve these bugs i.e. to reduce the bug-fix time as well as to assign priorities to bugs, according to bug-fix time. Handling the bugs efficiently could prove to be cost saving for industries. In this paper we would be proposing model for reducing bug fix time and solution on bug fix time.

Keywords: Cosine Algorithm, Bug-Fix time, TF-IDF (Term Frequency - Inverse Document Frequency).

I. INTRODUCTION

The level of software quality depends on the number of bugs occurred throughout out the developing phase of software development process and the time required to fix them. Bug(s) can be defined as a fault or error which gives uncertain output. The time taken for solving the bug or fixing the bug starting from bug reported time is called as bug-fix time.

Generally, when the bug is reported, project management team members' approaches technical experts or experience team members to get bug fix times which estimated and the response would be very much personalized and subjective.

In software development phase, the resource management and planning should be done by considering bugs and bug fix times, that result in less bugs and decrease in bug fix time. And so quality of software can be improved if bug-fix time can be predicted accurately.

In previous years, there has been lots of research on software intelligence problems like bug prediction and testing efforts required. Thus, some similar techniques can be provided to predict bug fix time. While developing prediction models, historical data about bug fix times of previous bugs can be used.

Each time when the bug is reported, related data will be quite limited, but as the time passby there will be huge amount of data will be gathered. Such as, some attributes like number of developers who participated in fixing the bug also the comments of developers about the bug will be additional information about the bug. Many researchers working on this problem have considered features from thedata that is available not only at the time of reporting but during the life time of the bug.

II. RELATED WORK

In this section we consider and explain the related work on this issue by research community. R. Buse et al. introduced the concept of Software Analytics can be defined as a process of assisting decision maker in extracting important information. It can be also said as process of predicting bug and bug fix time, which ensure quality of product.

Ramarao et al. have classified bugs with an accuracy of 65.11%, whereas the classification done by α-kNN is 53.64%, concluding that the proposed model is more efficient than the algorithm. Prediction of bug fix times, immediately after the bug has been reported has a lot of challenges, hence Ramarao et al. have proposed score of a reporter as a prominent feature to predict the time required to solve bugs.

W. Abdelmoez et al. have compared 3 active open source projects, categorizing bugs into classes according to the time required to solve them. Therefore the developers can work on the bugs that would take more time.

Giger et al. using decision tree have built a prediction model. The results produced having accuracy between 60% and 70%, categorizing bugs into slow and fast classes. Post submission data have improved the efficiency by 5% to 10%.

Weiss et al. built a model using α-kNN, considering only two data points, viz title and description. This model gave better results than Naïve Bayes approach. Their predictions have a high similarity to that of bug reports, saying it is possible that estimate the effort required immediately.

III.PROPOSED SYSTEM

In this section we will be discussing the system which we are going to develop and test it with our testing data.

Firstly the administrator adds the dataset in the system; the system would parse and tokenize the bugs. The admin would add some filter words that are not relevant to analyze the bugs. Once the filtering of stop-words is done, then the data is passed to TF-IDF algorithm.

Term Frequency - Inverse Document Frequency is an algorithm used to help us determine the frequency of words in those documents. The values gained from TF-

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

IDF are used in Cosine Similarity, this algorithm is used to find the cosine values of individual words, and comparing it with others, the smaller the angular difference, the more similar words are. When the values are generated, they are scored, and sorted in descending order.
The diagram below shows the architecture of our proposed system:

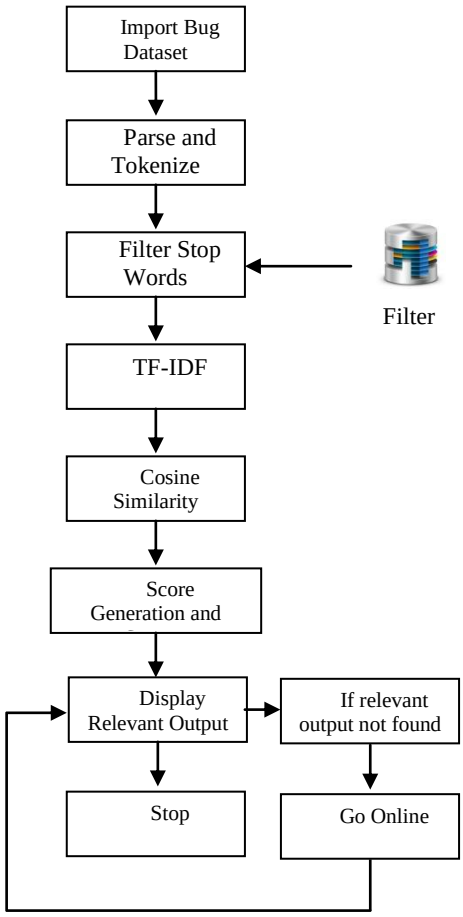

Fig. 1. Architecture of analysis and solution finding system

The relevant results are shown, the solution are also displayed accordingly. If the user is not satisfied with the results displayed, then he could go look for online solutions, even if the online solutions are not sufficient for him then he could forward this bug to the technical head, where the technical head would the provide the solution to it.

The system will store all the bugs reported by the reporter with its time, priority and status. Status may be solved or unsolved for bug. Our system will search bug description in database, which will quickly provide the reporter with optimized solution. If the solutions are not available or reported bug description is not available in dataset then such bug can be searched on Google using same application. Analysis of the reporters can be done, with parameters like title and description of bug as well as the time required to solve it would provide detailed reporting related to the bug, reporter and software respectively.

IV. METHODOLOGY

A. DATASET

The dataset used by the system to test is stackoverflow.com dataset. It is website which provides answers to programming language questions. Note that the programming language can be any language. The data from stackoverflow.com contains entries of past 8 years. The dataset is in the form of questions and answer. We will be filtering it, according to system requirement. The system only needs three parameters:

- Tittle
- Description
- Solution

Other parameters of dataset are omitted. The duplication of entities is avoided.

B. ALGORITHM

I.TF-IDF

Term weighting can be as simple as binary representation or as detailed as a mix of terms and existing datasets. TF-IDF is the most widely known and used weighting method, and it is remain even comparable with novel methods. The purpose of the text preprocessing stage is to refer to each document as a feature vector that is to divide the text into individual words.
In TF-IDF term weighting, the text documents are characterized as transactions. Choosing the keyword for the feature selection process is the main preprocessing process necessary for the indexing of documents. This study utilized two different TF-IDF methods, i.e. Global and normal, to weight the terms in term-document matrices of our evaluation datasets.
A common practice to avoid this variability or, at least, reduce the possible impacts resulting from it, is the normalization of the TF-IDF scores for each document in the collection by using the Euclidean norm is calculated by using Equations (1) and (2) respectively [5] as follows:

$$TF\text{-}IDF=\ln(TF)\times\ln(IDF) \quad (1)$$

$$TF\text{-}IDF=TF\times IDF \quad (2)$$

IiII.COSINE SIMILARITY

Cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any other angle. It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a Cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1].

Note that these bounds apply for any number of dimensions, and Cosine similarity is most commonly used

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

in high-dimensional positive spaces. For example, in Information Retrieval and text mining, each term is notionally assigned a different dimension and a document is characterized by a vector where the value of each dimension corresponds to the number of times that term appears in the document. Cosine similarity then gives a useful measure of how similar two documents are likely to be in terms of their subject matter. The technique is also used to measure cohesion within clusters in the field of data mining.
Cosine distance is a term often used for the complement in positive space, that is: It is important to note, however, that this is not a proper distance metric as it does not have the triangle inequality property and it violates the coincidence axiom; to repair the triangle inequality property whilst maintaining the same ordering, it is necessary to convert to Angular distance. One of the reasons for the popularity of Cosine similarity is that it is very efficient to evaluate, especially for sparse vectors, as only the non-zero dimensions need to be considered.

The cosine of two vectors can be derived by using the Euclidean dot product formula:

$$a.b = ||a|| \, ||b|| \cos \theta \qquad (3)$$

Given two vectors of attributes, A and B, the cosine similarity, cos(θ), is represented using a dot product and magnitude. The resulting similarity ranges from −1 meaning exactly opposite, to 1 meaning exactly the same, with 0 usually indicating independence, and in-between values indicating intermediate similarity or dissimilarity. For text matching, the attribute vectors A and B are usually the term frequency vectors of the documents. The cosine similarity can be seen as a method of normalizing document length during comparison. In the case of information retrieval, the cosine similarity of two documents will range from 0 to 1, since the term frequencies (TF-IDF weights) cannot be negative. The angle between two term frequency vectors cannot be greater than 90°.

V. CONCLUSION

We will be developing the system which will be more accurate in analysis and provides solution to bugs which may occur. The system may also generate report of each user to the system. The report may help in analyzing performance of user.

REFERENCES

[1] Pranav Ramarao et al. “Impact of Bug Reporter’s Reputation on Bug-fix Times”, International Conference on Information System Engineering, 2016.
[2] W. Abdelmoez et al. “Bug Fix-Time Prediction Model using Naïve Bayes Classifier”, ICCTA 2012, 13-15 October 2012.
[3] Emanuel Giger et al, “Predicting the Fix Time of Bugs”, RSSE 2010, May 4 2010.
[4] CathrinWeiß et al, “How Long will it Take to Fix This Bug?”, IEEE 2007.
[5] Raymond P.L. Buse& Thomas Zimmerman, “Analytics for Software Development”, FoSER, 2010.

BIOGRAPHIES

Akshay Khamkar Pursuing Bachelor of Engineering (B.E) in Information Technology from Sinhgad Academy of Engineering, Savitribai Phule Pune University (S.P.P.U)

Aishwarya Jagtap Pursuing Bachelor of Engineering (B.E) in Information Technology from Sinhgad Academy of Engineering, Savitribai Phule Pune University (S.P.P.U)

Neeraj Pattan Pursuing Bachelor of Engineering (B.E) in Information Technology from Sinhgad Academy of Engineering, Savitribai Phule Pune University (S.P.P.U)

Anuja Raipure Pursuing Bachelor of Engineering (B.E) in Information Technology from Sinhgad Academy of Engineering, Savitribai Phule Pune University (S.P.P.U)

Prof. Mrs. P.Y. Pawar, M.E. Computer Engineering, Sinhgad Academy of Engineering, Pune