IJARCCE

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 11, November 2015

Towards Effective Troubleshooting With Data Truncation

Karishma Musale[1], Gorakshanath Gagare [2]

M.E.Student, Department of Computer Engineering, SVIT, Nasik, India [1]

Assistant Professor, Department of Computer Engineering, SVIT, Nasik, India [2]

Abstract: The process of fixing bug is bug triage or bug assortment. The aim of this, to correctly assign a developer to a new bug. Triaging these incoming reports manually is error-prone and time consuming.Software companies pay most of their cost in dealing with these bugs. For software repositories traditional software analysis is not completely suitable for the large-scale and complex data.To reduce time and cost of bug triaging,present an automatic approach to predict a developer with relevant experience to solve the new coming report. In proposed approach explain data reduction on bug data set which will reduce the scale of the data as well as increase the quality of the data.And also give domain specific bugs with their solution by developers. For implementing this use instance selection and feature selection for reducing bug of data. And Top-K pruning algorithms for tackling domain specific task.

Keywords: Bug,Bug Triage,repositories,instance selection.

I. INTRODUCTION

For managing software bugs bug repository or bug fixing plays an important role. Large of software which are open source projects have an open bug repository which allows developers as well as users to submit issues or defects in the software that suggest possible solutions and remark on existing bug reports. The number of regular occurring bugs for open source large-scale software projects is so much large that makes the triaging process very difficult and challenging .For fixing software bugs most of software companies pays a lot . The large scale and the low quality are main two challenges which are related with bug data that may affect the effective use of bug repositories in software development tasks. Bug is maintained as a bug report in a bug repository that records the reproducing bug in textual form and updates according to the status of bug fixing[1].

A. *Objectives:*

1) Simultaneously reduce the scales of the bug dimension and the word dimension.
2) Improve the accuracy of bug triage.
3) Improve the results of data reduction in bug triaging to explore how to prepare a high quality set of bug data and tackle a domain specific task.

II. LITERATURE SURVEY

Following are the existing papers name with their description:

- **Automatic bug triage using text categorization**

In this paper[2], authors used an application of supervised machine learning using a naive Bayes classifier for automatically assign bug reports to developers. For that they experimented their approach on bug reports from a large open-source project such as Eclipse.org. And get 30% classification accuracy.

- **Improving Bug Triage with Bug Tossing Graphs**

In this paper[3], authors studied on 445,000 bug reports as well as their overall activities from the Mozilla and Eclipse projects.This steps takes long time for assign and toss bugs. For improving the bug assignment process and reduce unnecessary tossing steps, they used tossing graph model which used existing tossing history.This results as model reduces tossing steps by up to 72% and up to 23 percentage points improving the accuracy of automatic bug assignment.

- **COSTRIAGE: A Cost-Aware Triage Algorithm for Bug Reporting Systems:**

In this paper[4], authors used COSTRIAGE technique. The experiments reduces the cost without significantly sacrificing accuracy. They used a proof-of-concept implementation by using cost of bug fixing time. Developer profile model is general enough to support other code indicators such as interests, efforts, and expertise to optimize for both accuracy and cost for automatic bug triage.

- **Towards more accurate retrieval of duplicate bug reports**

In this paper[8],improved the accuracy of duplicate bug retrieval in two ways. First, *BM25F* is an effective textual similarity measure which is originally designed for short unstructured queries, and extend it to *BM25Fext* specially or lengthy structured report queries by considering weight of terms in queries. Second, authors proposed a new retrieval function *REP* fully utilizing not only text but also other information available in reports such as product, component, priority etc: A two-round gradient descent contrasting similar pairs of reports against dissimilar ones, is adopted to optimize *REP* based on a training set. They experimented on 4 sizable bug datasets extracted from 3 large open-source projects like OpenOffice, Firefox and Eclipse; and find that *BM25Fext* and *REP* are able to improve the retrieval performance. The experiments on the this showed that *BM25Fext* improves recall rate by 3–13% and MAP by 4–11% over *BM25F*.

- **Memories of bug fixes**

In this paper[7],authors used project-specific bug finding tool using memories of bug fixes. Potential bugs are detected by BugMem and which suggests corresponding fixes.They found that 19.3%-40.3% of bugs arrived repeatedly, and 7.9%-15.5% of bug and fix pairs arrived repeatedly in the history.To store histories and make backups, source code repositories such as CVS and Subversion are typically used.Their approach of computing memories of bug fixes provides a useful way to extract and deploy the knowledge latent in source code repositories. They tackle this information to improve the quality of source code and provide detailed guidance to developers.

- **Towards Effective Bug Triage with Software Data Reduction Techniques**

In this paper[1], For reducing the scale of bug data sets as well as improve the data quality combine feature selection with instance selection. For determinining the order of applying instance selection and feature selection for a new bug data set, This takes attributes of each bug data set and train a predictive model based on historical data sets. For experiments they use bug data set of Eclipse and Mozilla and get high quality bug data set.

III.PROPOSED SYSTEM

Manual Bug fixing is time consuming task and did't get accurate result.So that proposed system is provided.There is problem of getting accurate bug solution according to domain.In existing approach, get reduced bug dataset and high quality bug dataset. For that purpose, proposed system is provided.We used existing system instance selection and feature selection for reducing bug dataset.And additionaly use Top-K pruning algorithm for improving results of data reduction quality as compared to existing system and get domain wise bug solution.

A. *Architecture*

For fixing the bugs first we have to assign the bugs to developer.So,In this figure when there is new bugs arrived that time check this bug in bug repository, if this bug solution is already available, then fix this bug by already assign developers. But there is no bug solution that time assign this bug to new developer for fixing the bug based on the knowledge of historical bug fixing.For that purpose use instance selection and feature selection combinely for reducing the bug dataset and use Top-K pruning algorithm for solving the bug domain wise.

- **IS(Instance selection**) is for obtaining a subset of relevant instances (i.e., bug reports in bug data) .
  - ➢ Remove noise and redundant instances
  - ➢ Remove non-representative instances
- **FS(feature selection**) which aims to obtain a subset of relevant features (i.e.,words in bug data).
  - ➢ Sorting of words according to feature values

In that uses FS->IS to denote the bug data reduction, which first applies FS and then IS; on the other hand, IS->FS denotes first applying IS and then FS. After applying this get reduced dataset .When developer wants bug according to domain that time use Top-K Pruning algorithms.

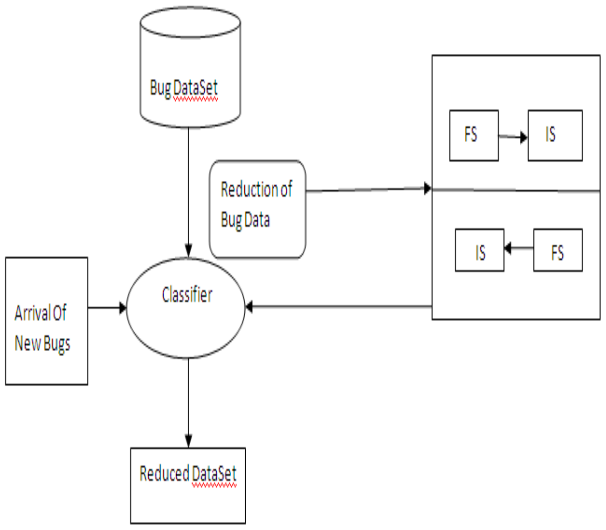
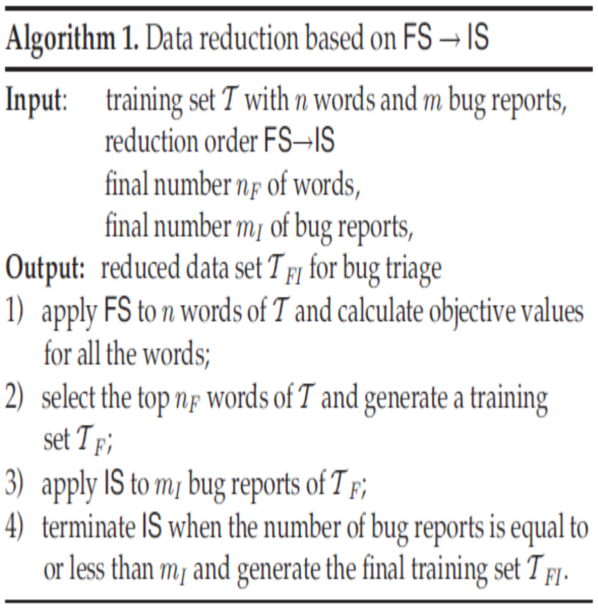


Fig. 1 System Architecture

B. *Algorithm*

Following algorithm is used for data reduction in bug fixing,which is based on feature selection and instance selection.



```
Algorithm 1. Data reduction based on FS → IS
Input:  training set $T$ with $n$ words and $m$ bug reports,
        reduction order FS→IS
        final number $n_F$ of words,
        final number $m_I$ of bug reports,
Output: reduced data set $T_{FI}$ for bug triage
1) apply FS to $n$ words of $T$ and calculate objective values
   for all the words;
2) select the top $n_F$ words of $T$ and generate a training
   set $T_F$;
3) apply IS to $m_I$ bug reports of $T_F$;
4) terminate IS when the number of bug reports is equal to
   or less than $m_I$ and generate the final training set $T_{FI}$.
```

IV.CONCLUSION

Software Companies spend most of their money for fixing bug.This is necessary for companies to solve the bugs.And this task is time consuming.So,In this paper we use existing system instance selection and feature selection method for getting reduced bug dataset.And additionaly use Top-K pruning algorithm for improving results of data reduction quality as compared to existing system and get domain wise bug solution.This work provides the accurate high quality bug dataset as well as provide domain specific task.

REFERENCES

[1] B Jifeng Xuan, He Jiang, Yan Hu, Zhilei Ren, Weiqin Zou, Zhongxuan Luo, and Xindong Wu," Towards Effective Bug Triage with Software Data Reduction Techniques" ieee transactions on knowledge and data engineering, vol. 27, no. 1, january 2015.

[2] D. Cubranic and G. C. Murphy, "Automatic bug triage using text categorization," in Proc. 16th Int. Conf. Softw. Eng. Knowl. Eng.,Jun. 2004, pp. 92–97.

[3] G. Jeong, S. Kim, and T. Zimmermann, "Improving bug triage with tossing graphs," in Proc. Joint Meeting 12th Eur. Softw. Eng. Conf. 17th ACM SIGSOFT Symp. Found. Softw. Eng., Aug. 2009,pp. 111–120.

[4] J. W. Park, M. W. Lee, J. Kim, S. W. Hwang, and S. Kim,"Costriage: A cost-aware triage algorithm for bug reporting systems,"in Proc. 25th Conf. Artif. Intell., Aug. 2011, pp. 139–144.

[5] A. E. Hassan, "The road ahead for mining software repositories,"in Proc. Front. Softw. Maintenance, Sep. 2008, pp. 48–57.

[6] J. Xuan, H. Jiang, Z. Ren, and W. Zou, "Developer prioritization in bug repositories," in Proc. 34th Int. Conf. Softw. Eng., 2012, pp. 25–35.

[7] S. Kim, K. Pan, E. J. Whitehead, Jr., "Memories of bug fixes," in Proc. ACM SIGSOFT Int. Symp. Found. Softw. Eng., 2006, pp. 35–45.

[8] J. Xuan, H. Jiang, Z. Ren, and W. Zou, "Developer prioritization in bug repositories," in Proc. 34th Int. Conf. Softw. Eng., 2012, pp. 25–35.

[9] H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," Data Mining Knowl. Discovery, vol. 6, no. 2, pp. 153–172, Apr. 2002.

[10] C. Sun, D. Lo, S. C. Khoo, and J. Jiang, "Towards more accurate retrieval of duplicate bug reports," in Proc. 26th IEEE/ACM Int. Conf. Automated Softw. Eng., 2011, pp. 253–262.