International Journal of Research in Advent Technology, Special Issue, March 2019
E-ISSN: 2321-9637
International Conference on Technological Emerging Challenges (ICTEC-2019)
Available online at www.ijrat.org

Intellectual Automated Bug Patients with Concentrating Report Model

[1]G.Pavan Kumar, [2]B.Ramesh
[1]Asst.Professor, [2]Assoc.Professor
[1,2]Computer Science & Engineering
Tirumala Engineering College Narasaraopet, Guntur,Andhra Pradesh
pavan89.gonuguntla@gmail.com, ramesh.guts@gmail.com

Abstract: Automatic classification of crowd sourced test reports is important due to their tremendous sizes and large proportion of noises. The challenge is how to trade off the matching degree between users' expertise and the question topic, and the likelihood of positive response from the invited users. In this paper, we formally formulate the problem and develop a weakly supervised factor graph (WeakFG) model to address the problem. The model explicitly captures expertise matching degree between questions and users. We propose an approach which combines collaborative filtering and topic modeling techniques. In the collaborative filtering component, given a new app, our approach recommends libraries by using its similar apps. Triaging these expansive quantities of approaching bug reports is a troublesome and tedious errand. Part of the bug triaging procedure is relegating a recently arrived bug report to an engineer who could effectively resolve the bug. Appointing bug report to the applicable designer is a vital stride as it decreases the bug hurling. Trained with historical data including identified duplicate reports, it is able to learn the sets of different terms describing the same technical issues and to detect other not-yet-identified duplicate ones. To improve bug triage, many studies have proposed automatic approaches to recommend proper developers for resolving bugs. These approaches are based on machine learning algorithms, which treat bug triage like text classification. We further analyze the deep reason and find that industrial data have significant local bias, which degrades existing approaches. We aim at designing an effective approach to overcome the local bias in real industrial data and automatically classifying true fault from the large amounts of crowd sourced reports.

Index Terms: Crowd sourced testing; Report classification; Cluster, Topic Modeling, Collaborative Filtering, , Feature Information, Topic Model.

1. INTRODUCTION

Bugs are the programming blunders that cause noteworthy execution debasement. Bugs lead to poor client experience and low framework throughput [1]. Extensive open source programming improvement ventures. Mozilla and Eclipse get numerous bug reports. They more often than not utilize a bug following framework where clients can report their issues which happened in their separate undertakings [2]. Support Vector Machine (SVM) was utilized train an SVM classifier, all pairs of duplicate bug reports are formed and considered as the positive samples and all other pairs of non-duplicate bug reports are used as the negative ones. The key limitation of ML approaches is their low efficiency. The recent work by Sun et al. has shown an advanced IR approach [3]. Outperformed state-of-the-art ML approaches in term of both accuracy and time efficiency [4]. It is customized from account the long bug reports and the meta-data such as the reported product, component, and version [5]. We propose a method utilizing relevant search technique. Our method essentially builds a search application for bug reports [6]. With this search application, we can find relevant bug reports for a given bug then analyze them to get the corresponding developers [7]. We propose an automated approach AppLibRec which combines and collaborative filtering to recommends a list of third party libraries for mobile apps. Specifically, AppLibRec performs two kinds of analysis: README file based analysis (RM-based) and libraries based analysi In the RM-based component [8].

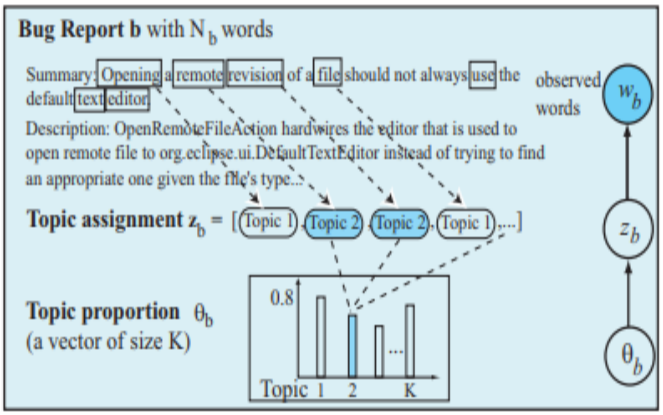

Figure 1: Document Modeling

2. RELATED WORK

To our best knowledge is most related to ours proposed combines association rule mining and a nearest-neigh borated collaborative filtering approach to recommend libraries for projects [9]. Recommender systems are widely utilized in software engineering many previous studies have proposed approaches to recommend various code using various information sources by various heuristics propose the problem of mining code fragments that satisfy the query which describes the input and output types [10]. The goal is to assign a list of experts to a list of queries by considering matching degree, load balance, and other constraints. There was a bunch of research on this topic, probably starting from conference-paper reviewer assignment [11]. Our observation on real industrial crowd sourced testing data reveals that previous approaches generate poor and unstable performances on these industrial data [12]. We further analyze the deep reason and find that industrial data have significant local bias data are heterogeneous within dataset and their distributions are often different from one project to another In general, testers in prepare packages for crowd sourced testing and distribute them online using their crowd sourced testing platform. Then, crowd workers could sign in to conduct the task and are required to submit crowd sourced test reports [13]. We develop Duplicate Bug report Topic Model, DBTM, a duplicate bug report detection model that takes advantage of both term-based and topic-based features. In DBTM, a bug report is viewed as a textual document describing one or more technical issues in a system [14].

3. SYSTEM MODELS

The framework consists of five steps. It first extracts necessary information such as summary, description, product and component from the original bug reports [15]. Then it indexes the extracted information. When we need to recommend developers for a new bug report, it first executes the same information extraction and indexing [16]. Duplicate reports describe the same technical issue reporters might discuss other relevant topics and phenomena, and provide the insights on the bug including suggested fixes and relevant technical functions. Those observations suggest that the detection of duplicate bug reports could rely not only on the technical terms, but also on the technical topics in the reports [17]

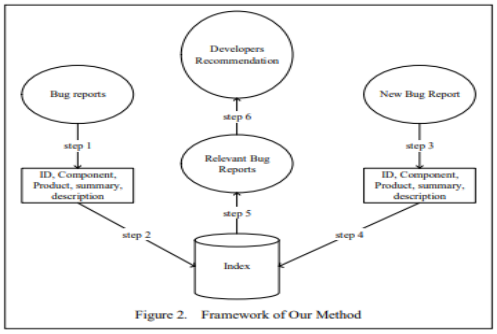

Figure 2. Framework of Our Method

4. PROPOSED SYSTEM

We first provide a description of our overall framework then propose method to solve the third party libraries recommendation we describe the RM-based component which uses LDA model. We present the Lib-based component which uses collaborative filtering [18]. Finally we aggregate the two components presented. We propose a cluster-based classification approach to overcome the local dilemma. The overview of our approach we carry out feature extraction, which is to obtain features from crowd sourced report to train machine learning classifiers. Secondly, we cluster similar reports together to construct more homogeneous training dataset we build classifiers based on most similar clusters and combine classification results from these classifiers [19]. Cluster Label Generation, which is utilized to produce the group marks utilizing the continuous terms present as a part of the bugs of a bunch. Mapping of the bunch names to the bug classifications utilizing the ordered terms, that are predefined for different classifications is completed next [20].

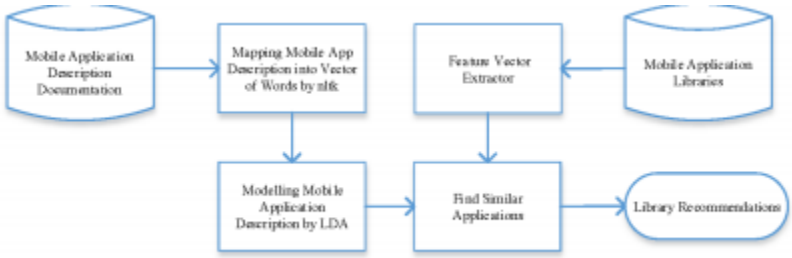

Figure 3: Third Party Libraries Recommendation Framework

5. METHODOLOGIES

Given a query our goal is to find experts with sufficient knowledge who are willing to answer the question or review the paper with limited amount of labeled data. Let us consider a set of N candidate experts E = {e1, · · · , eN }. The general problem can be then formalized as seeking a ranking function to quantify each expert ei's expertise degree and acceptance probability simultaneously [21]. The ranking score Sqi was defined by only considering the expertise matching degree between q and the expert ei One method is to use language models which estimates the generative probability of a query given each document using a smoothed word distribution. Afterwards topic models have been used to model latent topics in order to improve the matching accuracy [22].

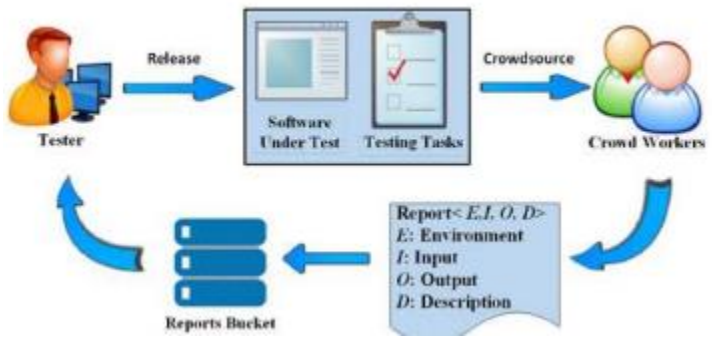

Figure 4: The procedure of crowd sourced testing [5]

A. Extracting Features

The goal of feature extraction is to obtain features from crowd sourced reports which can be potentially used as input to train machine learning classifiers. We abstract these features from the following three dimensions: textual, sentiment, and crowd worker experience. Each dimension may contain a set of features. For textual dimension, we first collect different sources of text description together [23]. Then we conduct word segmentation, as the crowd sourced reports in our experiment are written in Chinese. We adopt for word segmentation and segment descriptions into words. We then remove stop words (i.e., "the", "am", "on", etc.) to reduce noise. Note that, workers often use different words to express the same concept, so we introduce the synonym replacement technique to mitigate this problem. Synonym library of LTP5 is adopted. Each of the remaining word token corresponds to a feature. For each feature, we take the frequency it occurs in

the description as its value. We organize all the features under investigated into a feature vector. For each crowd sourced report, we extract the value for each feature and prepare the feature vector for building machine learning classification models.

B. Clustering Reports

For mitigating the local bias of our industrial crowd sauced data we perform clustering techniques to ensure the homogeneity within training data. Intuitively, the crowd sourced reports within the same cluster tend to homogeneous with each other. Specifically we apply K-means algorithm to separate all training data into different clusters. Note that, we break the boundary exerted by projects and reorganize all available crowd sourced reports into clusters [24].

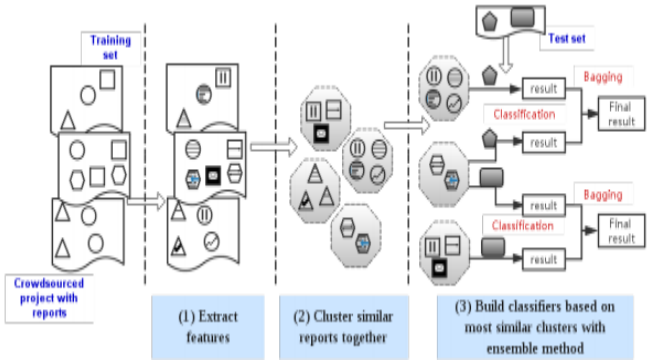

Figure 5: Overview of our cluster-based classification approach

C. Prediction Algorithm for T-Model

The goal of this algorithm is to estimate the topic proportion of a newly arrived bug report and calculate the topic similarity to other bug reports and duplicate groups. The algorithm uses the trained model from the previous algorithm to estimate the topic proportion and uses the Jensen-Shannon divergence to calculate the topic similarity between each bug report in all groups of duplicate reports. The similarity combination with BM25F-based similarity sim2, will be used to estimate how likely b can be a duplicate of the reports in the group [25].

6. EXPERIMENT RESULTS

We describe our dataset, and present our evaluation measures. And then, we demonstrate our research questions and the results of our experiments. Finally, we discuss the threads to validate. Filtering. This method recommends libraries by investigating the set of libraries that are used by similar applications, using a nearest neighbor based collaborative filtering approach. We tune each algorithm to its best parameter, compared with the state-of-the-art work and the improvement of over it. The statistically significant improvements are highlighted in bold. From Table 1, the improvement of our approach over it is significant. commonly used latent factor based model for text similarity. Our RM-based component recommends libraries by finding similar applications, using LDA model to find the similar applications. Our Lib-based component recommends libraries by investigating the set of libraries that are used by similar applications, also using a nearest neighbor based collaborative filtering approach. It uses PCC as the metric to compute this distance.

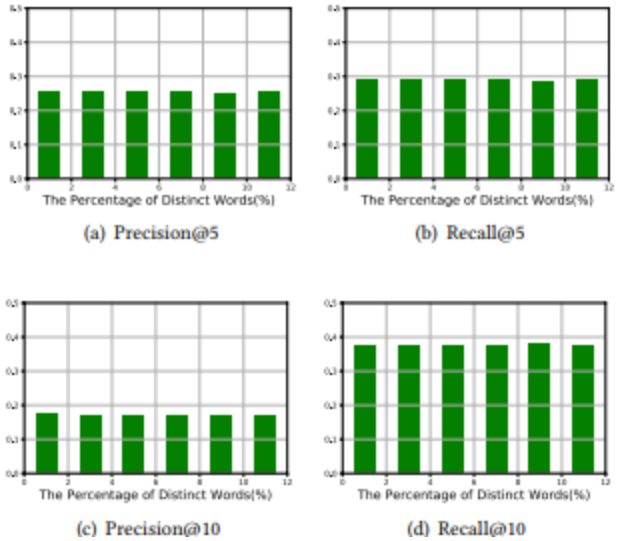

Figure 6: Recall Different Numbers of Topics of RM-based

7. CONCLUSIONS AND FUTURE WORK

The cluster-based classification model is classify true fault from the large amounts of crowd sourced reports. This can potentially help to reduce the effort required for manually inspecting the reports and facilitate project management in crowd sourced testing. To improve efficiency and effectiveness of bug triage our method recommends developers for bugs to be assigned. Besides we utilized component and product information of bug reports to improve the performance of our method. A single error takes too much time and companies need to spend huge amount of money on single bug. It is not affordable for companies where time and money matters a lot. So time and money can be utilized by providing all solution in developer's desk even if he is not facing this bug. However these mobile applications may not contain the sufficient co-occurrence of third party libraries. We propose a new method AppLibRec to automatically recommend third party libraries for the developers. Future work will also include exploring transfer learning and other techniques to further improve the model performance and stability.

REFERENCES

[1] Eclipse bug tracking system.
[2] Anvik J, Hiew L, Murphy GC. Who should fix this bug?. InProceedings of the 28th international conference on Software engineering 2006 May 28 (pp. 361-370). ACM.
[3] Jeong G, Kim S, Zimmermann T. Improving bug triage with bug tossing graphs. InProceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium.
[4] Zhang T, Jiang H, Luo X, Chan AT. A Literature Review of Research in Bug Resolution: Tasks, Challenges and Future Directions. The Computer Journal. 2016 May 1;59(5):741-73.
[5] Hu H, Zhang H, Xuan J, Sun W. Effective bug triage based on historical bug-fix information. InSoftware Reliability Engineering (ISSRE), 2014 IEEE 25th International Symposium on 2014 Nov 3 (pp. 122-132). IEEE.
[6] Bhattacharya P, Neamtiu I. Fine-grained incremental learning and multifeature tossing graphs to improve bug triaging. InSoftware Maintenance (ICSM), 2010 IEEE International Conference on 2010 Sep 12 (pp. 1-10). IEEE.
[7] Bhattacharya P, Neamtiu I, Shelton CR. Automated, highly-accurate, bug assignment using machine learning and tossing graphs. Journal of Systems and Software. 2012 Oct 31;85(10):2275-92.
[8] Matter D, Kuhn A, Nierstrasz O. Assigning bug reports using a vocabulary-based expertise model of developers. InMining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on 2009 May 16 (pp. 131-140). IEEE.
[9] Tamrawi A, Nguyen TT, Al-Kofahi JM, Nguyen TN. Fuzzy set and cachebased approach for bug triaging. InProceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering 2011 Sep 5 (pp. 365-375). ACM.
[10] Xuan J, Jiang H, Ren Z, Zou W. Developer prioritization in bug repositories. InSoftware Engineering (ICSE), 2012 34th International Conference on 2012 Jun 2 (pp. 25-35). IEEE.
[11] Ensemble averaging. http://en.wikipedia.org/wiki/Ensemble averaging.

[12] M. Fischer, M. Pinzger, and H. Gall. Analyzing and relating bug report data for feature tracking. In WCRE'03. IEEE CS, 2003.
[13] Gibbs sampling. wikipedia.org/wiki/Gibbs sampling
[14] L. Hiew. Assisted detection of duplicate bug reports. Master's thesis, University of British Columbia, 2006.
[15] P. Hooimeijer and W. Weimer. Modeling bug report quality. In ASE'07, pages 34–43. ACM, 2007.
[16] N. Jalbert and W. Weimer. Automated duplicate detection for bug tracking systems. In Int. Conf. on Dependable Systems and Networks, pp. 52–61. 2008.
[17] S. Kim and E. J. Whitehead, Jr. How long did it take to fix bugs? In MSR'06, pages 173–174. ACM, 2006.
[18] A. J. Ko, B. A. Myers, and D. H. Chau. A linguistic analysis of how people describe software problems. In VLHCC'06, pages 127–134. IEEE CS, 2006.
[19] T. Menzies and A. Marcus. Automated severity assessment of software defect reports. In ICSM'08.
[20] A. T. Nguyen, T. T. Nguyen, J. Al-Kofahi, H. V. Nguyen, and T. N. Nguyen. A Topic-based Approach for Narrowing the Search Space of Buggy Files from a Bug Report. In ASE'11, pp. 263-272. IEEE CS, 2011
[21] X. Xia, D. Lo, Y. Ding, J. M. Al-Kofahi, T. N. Nguyen, and X. Wang, "Improving automated bug triaging with specialized topic model," IEEE Transactions on Software Engineering, vol. 43, no. 3, pp. 272–297, 2017.
[22] X. Xia, D. Lo, X. Wang, and X. Yang, "Who should review this change?: Putting text and file location analyses together for more accurate recommendations," in ICSME. IEEE, 2015, pp. 261–270.
[23] X. Xia, D. Lo, X. Wang, and B. Zhou, "Dual analysis for recommending developers to resolve bugs," Journal of Software: Evolution and Process, vol. 27, no. 3, pp. 195–220, 2015.
[24] S. Wang, W. Zhang, and Q. Wang. Fixercache: Unsupervised caching active developers for diverse bug triage. In ESEM 2014, pages 25:1–25:10.
[25] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun. An approach to detecting duplicate bug reports using natural language and execution information. In ICSE 2008, pages 461–470.