Duplicate Bug Report Detection

Tarun Chhabra

Computer Science

Department

North Carolina State University

Raleigh, NC, 27606, USA

Devika Desai

Computer Science Department

North Carolina State University

Raleigh, NC, 27606, USA

Sudipto Biswas

Computer Science Department

North Carolina State University

Raleigh, NC, 27606, USA

There are many users interacting with a system and reporting issues concerned with it in terms of a bug report. Bug reports are then used to guide software corrective maintenance activities so that it can result in more reliable software systems. Bug repositories are maintained as a collection of bug reports. Despite the advantages of a bug report system, it does cause some challenges. As bug reporting process is often ad-hoc, often the same bugs are reported by different users, or a different bug caused by the same potential software defect result in duplicate bug reports. A number of studies have attempted to address this issue by automating bug-report deduplication. The following review provides an overview of all the techniques used for duplicate bug detection before the concept of detection by information retrieval [1] and how the field has changed after that. A number of past studies have proposed a number of automated approaches to detect duplicate bug reports. We comment on how this approaches can be improved by integrating two or more techniques or considering even more possible factors for accurate detections.

INTRODUCTION

Due to system complexity and inadequate testing, many software systems are often released with defects or have some unknown bugs. Bugs occur for a variety of reasons, ranging from ill-defined specifications, to carelessness, to programmers misunderstanding of the problem, technical issues, nonfunctional qualities, corner cases, etc. To overcome such situations developers often need proper feedback on the bugs that are present in the systems. For this, they allow users to report such bugs using bug report systems such as Bugzilla, Jira or other propriety systems. With such systems, end users and testers could report bugs that they encounter and developers could triage, track, and comment on the various bugs that are reported. Bug reporting is standard practice in both open source software development and closed source software development.

For several reasons, such as lack of motivation of users and defects in the search engine of the bug tracking systems, the users of software systems may report bugs that are already present in the bug tracking system. These bug reports are called “duplicates”. The word duplicate may also represent the bug reports referring to different bugs in the system that are caused by the same software defect.

To automate the detection of duplicate bug reports, several approaches have been introduced so far in the papers we studied. Early approaches have applied information retrieval (IR) to this problem with Vector Space Model (VSM) in which a bug report is modeled as a vector of textual features computed via Term Frequency-Inverse Document Frequency (Tf-Idf) term weighting measurement. To improve the detection accuracy, natural language processing (NLP) has been combined with those IR methods. Execution trace information on the reported bugs in the bug reports is also used in combination with NLP. However, execution traces might not be available in all bug reports. Another predominant approach to this problem is machine learning (ML). Jalbert use a binary classifier model and apply a linear regression over textual features of bug reports computed from their terms’ frequencies. To 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 has shown that REP, an advanced IR approach, outperformed state-of-the –art ML approaches in term of both accuracy and time efficiency. It is customized from BM25F to take in account the long bug reports and the meta-data such as the reported product, component, and version. The key assumption in REP is based on high textual similarity between duplicate bug reports. However, in practice, it is popular that the bug reports can be filed by multiple reporters who could describe about the same technical issues in different phenomena via different terms. With different input data, usage environments and scenarios, an erroneous behavior might be exposed as different phenomena (e.g. different outputs, traces, or screen view). Moreover, different reporters might use different terminologies and styles, or write about different phenomena to describe the same issues. Thus, duplicate bug reports might not be very textually similar. In those cases REP does not detect them well.

Another approach introduced is DBTM, a duplicate bug report detection model that takes advantage of not only IR-based features but also topic based features from novel-topic model, which is designed to address textual dissimilarity between duplicate reports.

Bug fixing is important in producing high quality software. Bug fixing can be carried out in both development and post-release time. In both the cases the developers carry out necessary testing and find the incorrect behaviors that do not conform with their expectations and software requirements.

REFERENCES.

[1] Anh Tuan Nguyen, Tung Thanh Nguyen, Tien N. Nguyen, David Lo, Chengnian Sun. 2012. Duplicate Bug Report Detection with a Combination of Information Retrieval and Topic Modeling. Automated Software Engineering (ASE), 2012 Proceedings of the 27th IEEE/ACM International Conference.