Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

A component recommender for bug reports using Discriminative Probability Latent Semantic Analysis

Meng Yan[b], Xiaohong Zhang[a,b,c,*], Dan Yang[b], Ling Xu[b], Jeffrey D. Kymer[b]

[a] State Key laboratory of Coal Mine Disaster Dynamics and Control, Chongqing 400044, PR China
[b] School of Software Engineering, Chongqing University, Chongqing 401331, PR China
[c] Key Laboratory of Dependable Service Computing in Cyber Physical, Society Ministry of Education, Chongqing 400044, PR China

A R T I C L E   I N F O

Article history:
Received 9 February 2015
Revised 15 January 2016
Accepted 16 January 2016
Available online 23 January 2016

Keywords:
Bug reports
Discriminative topic model
Component recommendation
Bug triage

A B S T R A C T

Context: The component field in a bug report provides important location information required by developers during bug fixes. Research has shown that incorrect component assignment for a bug report often causes problems and delays in bug fixes. A topic model technique, Latent Dirichlet Allocation (LDA), has been developed to create a component recommender for bug reports.
Objective: We seek to investigate a better way to use topic modeling in creating a component recommender.
Method: This paper presents a component recommender by using the proposed Discriminative Probability Latent Semantic Analysis (DPLSA) model and Jensen–Shannon divergence (DPLSA-JS). The proposed DPLSA model provides a novel method to initialize the word distributions for different topics. It uses the past assigned bug reports from the same component in the model training step. This results in a correlation between the learned topics and the components.
Results: We evaluate the proposed approach over five open source projects, Mylyn, Gcc, Platform, Bugzilla and Firefox. The results show that the proposed approach on average outperforms the LDA-KL method by 30.08%, 19.60% and 14.13% for recall @1, recall @3 and recall @5, outperforms the LDA-SVM method by 31.56%, 17.80% and 8.78% for recall @1, recall @3 and recall @5, respectively.
Conclusion: Our method discovers that using comments in the DPLSA-JS recommender does not always make a contribution to the performance. The vocabulary size does matter in DPLSA-JS. Different projects need to adaptively set the vocabulary size according to an experimental method. In addition, the correspondence between the learned topics and components in DPLSA increases the discriminative power of the topics which is useful for the recommendation task.

1. Introduction

Bug report is a fundamental artifact for informing developers about software problems. Research on mining bug report repositories has demonstrated success in a variety of software activities, such as tracing the evolution of the project [1], evaluating developer's expertise and contribution [2] and improving the software product quality [3]. However, the use of a bug report depends on correct triaging. Every new bug report has to be triaged to a component when submitted. This allows an efficient fixing by the appropriate development team which is responsible for the component [4]. Previous research reported that bug reporters frequently make inaccurate decisions in assigning the categorical fields such as the component in a bug report [5]. Unfortunately, bug reports are always triaged manually, which is time consuming and error prone. Take Eclipse as an example, approximately 25% of the bug reports need to be reassigned because of triaging mistakes [6], and it costs nearly two person-hours per day in triaging bug reports [7]. To assist correct bug triaging, this paper presents a novel Discriminative Probability Latent Semantic Analysis (DPLSA) based component recommender, in which a recommender is provided with a small list of component suggestions.

Across the popular issue tracking systems, such as JIRA[1] and Bugzilla[2], the bug reports submitted by reporters possess the following common features: first, there are a variety of mandatory and categorical fields like component, product, status and assigned to. Second, there are some other fields which are non-structured natural language text, including report title, detail description and

* Corresponding author at: School of Software Engineering, Chongqing University, Chongqing 401331, PR China. Tel.: +86 15923238399.
E-mail address: xhongz@cqu.edu.cn (X. Zhang).

[1] http://www.atlassian.com/software/jira/, verified 2015 /08/13.
[2] http://www.bugzilla.org/, verified 2015 /08/13.

http://dx.doi.org/10.1016/j.infsof.2016.01.005
0950-5849/© 2016 Elsevier B.V. All rights reserved.

comments which are written by developers or users. Third, each free-form text field in the bug reports contains rich information which reflects different facts about the bug. For example, the full free-form description text field reflects the detailed bug effects and provides indispensable conditions to track the bug. By utilizing these features, researches have proposed a variety of approaches to investigate different bug report related activities, such as duplicate bug report detection [8], bug report summarization [9] and bug triaging recommendations [6]. In this work, we focus on creating a component recommender by utilizing the full free-form text fields including title, description and comments.

This proposed method is motivated by the recent success of topic modeling in mining bug reports [4,10-13]. The most similar work is a Latent Dirichlet Allocation (LDA) [14] based component recommender proposed by Somasundaram and Murphy [4]. Although they achieved a state-of-art performance, there are still several unsolved issues in this line of research. First, a critical issue in topic modeling is how many topics should be sought [15]. There is no agreed-upon method for choosing the right number of topics among different datasets. In the similar work proposed by Somasundaram and Murphy [4], the number of topics varied from 10 to 120 and it was determined by using a series of experiments. Second, it is rarely discussed that the comments in bug reports decrease the performance of the recommendation or improve it. It is hasty to decide whether or not to consider comments. There are different decisions in related works. For example, Naguib et al. [16] did not adopt the comments in building the recommender. Shokripour et al. [17], Xuan et al. [18] and Zhang et al. [19] used the comments in their recommendation works. Third, the vocabulary in bug reports has an impact on the bug assignment accuracy [20]. Meanwhile, vocabulary plays a significant role in topic modeling [14]. However, it is rarely discussed whether the vocabulary size impacts the component recommender based on topic modeling.

To address the aforementioned issues, we present a Discriminative Probability Latent Semantic Analysis (DPLSA) model with discriminative power to create a recommender that assists with the component assignment task. We divide our recommender into three phases, namely the training phase, the testing phase and the recommendation phase. In the training phase, we use past assigned bug reports as our training datasets. The number of topics is simply set to the number of components. We develop a novel method to initialize the word distributions for different topics. This introduces the past component labels to initialize the topic-conditional probability of a particular word. Thus, the topics are estimated in a supervised way. As a result, it creates a correspondence between learned topics and components. This increases the discriminative power of the learned topics and overcomes the difficult in determining an appropriate topic number in LDA. In the testing phase, given a new bug report, we fix the obtained word-topic distribution in the training phase and compute the test sample's topic distribution by the standard EM algorithm of Probability Latent Semantic Analysis (PLSA). In the recommendation phase, the component recommendations are decided by ranking the divergence with each component's centroid topic representation in the training set. A small list of top-k most suitable components are recommended. The similarity with Somasundaram and Murphy [4] is that both of us use the topic modeling and divergence measuring technique. The main differences from Somasundaram and Murphy [4] lie in: first, we investigate the impacts of comments and vocabulary size which is not discussed in their work. Second, we use DPLSA instead of LDA. In summary, the contributions in our paper are threefold:

- We propose DPLSA, which performs an initialization method in the PLSA modeling step. It assigns the category-conditional probability of a specific word conditioned on the corresponding topic by using the assigned bug reports. Such that a correspondence between components and topics is created. It is utilized to enhance the discrimination of the estimated topics and overcomes the difficulty in choosing the right number of topics. (See Section 3.2.1).
- We evaluate our DPLSA-JS recommender on five open source projects to validate the effectiveness. Compared with two LDA based methods [4], we show that DPLSA-JS outperforms LDA-KL and LDA-SVM on the component recommendation problem by a substantial range. (See Section 4.5.1).
- We explore the impact of vocabulary size and comments on the recommendation performance by conducting a comparative study on five datasets. As a result, we find that the vocabulary size and whether or not using the comments impact the recommendation performance (See Sections 4.5.2 and 4.5.3).

The paper is structured as follows: Section 2 presents the related work of our research, including bug triage recommenders, topic modeling in mining bug reports and similar supervised topic models. We describe our research preparation, models and techniques in Section 3. Section 4 presents the research questions, sketches the experiment design, results and comparisons. Section 5 provides the threats to validity, including internal validity and external validity. Then at last in Section 6, we draw a conclusion about our findings and provide our future plans.

2. Related work

In this section, we discuss related literature from three aspects: bug triage recommenders, topic modeling in mining bug reports and similar supervised topic models.

2.1. Bug triage recommenders

A variety of techniques have been investigated to guide the bug triaging process, such as duplicate bug report detection [8,21,22], developer recommenders [7,12,13,16,17,23-27] and component recommenders [4-6].

Many researches have focused on automating the process of detecting duplicate bug reports by analyzing the free-text fields which is also used in this work. Several kinds of methods were employed, such as the statistical model, vector space model (VSM) [28], and TF-IDF [21]. In 2005, Anvik et al. [22] built a statistical model, using cosine similarity and detected 28% of all duplicate reports on Firefox. Hiew [29] improved on it by applying VSM to detect duplicate bug reports. The TF-IDF term weighting measurement was employed in their work and achieved a 50% recall and 29% precision on Firefox and 20% and 14% for Eclipse, respectively. The similarity between their work and this work is the textual processing on non-structured natural language fields of bug reports. While they differ from our work in how the training data is labeled and what data is used.

There have been several researches on creating developer recommenders which automatically assign a bug report to a particular developer. Cubranic [30] first modeled a bug report developer recommender by using a text classification method. After that, Anvik et al. [6,7] enhanced the above work by removing inactive and less active developers (according to the count of recent bug fixes). In their experiments on five open source projects, they increased the precision accuracy up to 64% by using three machine learning classifiers, namely C4.5, Naive Bayes and SVM. Based on these works, several approaches such as bug tossing (i.e., bug reports are reassigned to other developers) and source location based methods were proposed to enhance the existing performance. For example, Jeong et al. [24] presented an enhanced