The Journal of Systems and Software 113 (2016) 296–308

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Automatically classifying software changes via discriminative topic model: Supporting multi-category and cross-project

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

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

ARTICLE INFO

Article history:
Received 14 November 2014
Revised 11 November 2015
Accepted 7 December 2015
Available online 22 December 2015

Keywords:
Software change classification
Multi-category change
Discriminative topic model

ABSTRACT

Accurate classification of software changes as corrective, adaptive and perfective can enhance software decision making activities. However, a major challenge which remains is how to automatically classify multi-category changes. This paper presents a discriminative Probability Latent Semantic Analysis (DPLSA) model with a novel initialization method which initializes the word distributions for different topics using labeled samples. This method creates a one-to-one correspondence between the discovered topics and the change categories. As a result, the discriminative semantic representation of the software change messages whose largest topic entry directly corresponds to the category label of the change message which is directly used to perform single-category and multi-category change classification. In the evaluation on five open source projects, the experimental results show that the proposed approach achieves a more accurate performance than the four baseline methods. Especially with the multi-category classification task which improves the recall rate. Moreover, the different projects share the same vocabulary and the estimated model so that DPLSA is well applicable to cross-project software change message analysis.

1. Introduction

To aid further software analysis, it is necessary to classify software change as corrective, adaptive, or perfective. The proportion of each category provides a valuable window into the software development practices. Project managers need to be well informed to enhance their decision making process. For example, if 90% of the changes in a project are corrective, then it may mean that now is the time to intensify the quality assurance work like code reviews and unit tests. It has been applied to many important software engineering activities, such as software maintenance (Mockus and Votta, 2000) and defect prediction (Kim et al., 2008). Various change cues have been used for classifying software changes, for example, change author (Hindle et al., 2009a), change file (Alali et al., 2008), change size (Hattori and Lanza, 2008) and change messages (Hassan, 2008). In particular, change messages are attractive for software change classification because it does not require retrieving and then analyzing the source code of the change. Moreover, retrieving only the message is significantly less expensive, and allows for efficient browsing and analysis of the changes and their constituent revisions. These characteristics are useful to anyone who needs to quickly categorize or filter out irrelevant revisions (Hindle et al., 2009a). However, to the best of our knowledge, there has been little work classifying a multi-category change; but, there have been researches that have found this to be a realistic activity (Fu et al., 2015; Mauczka et al., 2012). In this work, we aim to accurately understand the category distribution by classifying both single-category and multi-category changes.

Researchers have proposed a variety of approaches for retrieving keywords in change messages to classifying software changes (Hassan, 2008; Mauczka et al., 2012; Mockus and Votta, 2000). Despite the great success achieved, there are some unsolved issues remaining in this research, such as the ambiguity coming from subjective interpretations of the relationship between relevant words and categories of changes. A similar work has demonstrated success in automatic software change classification by using semi-supervised Latent Dirichlet Allocation (LDA) (Fu et al., 2015). We noticed that both Mauczka et al. (2012) and Fu et al. (2015) found that single-category changes are not necessarily realistic. A major challenge which remains is how to automatically classify multi-category changes. To address this challenge, we focus on automatically classifying software changes by developing a novel discriminative Probability Latent Semantic Analysis, referred to DPLSA. The main difference from Fu et al. (2015) is that the three topics in this work have a one-to-one correspondence to

* Corresponding author. Tel.: +86 15923238399.
E-mail address: xhongz@cqu.edu.cn (X. Zhang).

http://dx.doi.org/10.1016/j.jss.2015.12.019
0164-1212/

corrective, adaptive and perfective software change categories, such that the change message categorization comes down to finding the single maximum entry (single-category) or multi maximum entries (multi-category) in the topic-document distributions, and we provide a method of cross-project classification without the need of re-learning. In particular, we motivated our investigation with three research questions:

RQ1 *What is a better way to evaluate the relationship between relevant words and the categories of software changes?* A change message often is a short description written by developers and the VCS does not enforce how to write a change message. Consequently, change messages are non-structured free format text. There are many salient words relevant to categories in change messages, such as "fix", "create" and "correct". The relationship between the relevant words and the categories is the key issue in the classification step. Mauczka et al. (2012) assigned weights to the salient words which is a subjective interpretation. We wish to perform a cross-project training using labeled messages to automatically determine a probabilistic relationship.

RQ2 *How well do the discovered topics correspond to software changes with multi-category?* A change message indicates a particular maintenance task, such as fixing a defect or adding a new feature, despite the fact that there exists a few change messages, which indicate multiple purposes as Mauczka et al. (2012) and Fu et al. (2015) presented in their validation step. We wish to create a one-to-one correspondence between discovered topics and categories by using the discriminative topic model. After that, the discovered topics can be directly used to perform the classification task including single-category and multi-category.

RQ3 *What is an accurate way to automatically obtain the distribution of software changes?* A project manager would be interested in knowing the distribution of categories of software changes. We wish to quantify a more accurate distribution by classifying both single-category and multi-category changes.

We address our research questions by proposing a topic modeling method. It is inspired by the recent success of topic modeling in mining software repositories (Grant et al., 2012; Hindle et al., 2011; Hindle et al., 2009b; Pollock et al., 2013; Thomas, 2012). Topic models, such as Probability Latent Semantic Analysis (Hofmann, 2001), Latent Dirichlet Allocation (Blei et al., 2003), Correlated Topic Models (Lafferty and Blei, 2006) and their variants and extensions, have been applied to various software engineering research questions, such as software evolution and software defect prediction (Chen et al., 2012; Gethers and Poshyvanyk, 2010; Grant et al., 2012). Despite the great success achieved, there are some unsolved, important issues that still remain in this line of research. First, in the original topic models, some words which are fully connected to different topics are noisy and irrelevant for model construction. Disconnecting the irrelevant words is helpful for generating a sparse representation over different topics of a document (Chien and Chang, 2014). In fact, the sparsity of the topic-document distribution (i.e. with a small number of dominant entries and most zero or close to zero entries) is helpful for directly performing the classification task. Second, a critical issue in understanding the latent topics uncovered from software repositories is how many topics should be sought (Grant et al., 2013). There is not a one-to-one correspondence between topics and category labels in the traditional models. The topic-document distributions are only used to decide which topics are important for a particular document and cannot determine which category a particular document belongs to. This is also the limitation in solving the multi-category problem.

The process of the proposed DPLSA is divided into three phases as illustrated in Fig. 1. We select single category change messages as our training datasets and use the semantically salient words derived from the work of Mauczka et al. (2012) to form the vocabulary as illustrated in Fig. 1(a). Moreover, the training messages from the same category are employed to initialize the category-conditional probability of a specific word conditioned on the corresponding topic. Hence, semantically salient words are forced to connect to the topic partially with a dominated probability. Such that, it creates a one-to-one correspondence between topics and categories. Due to the special initialization approach, the sparsity is achieved for the corresponding words to the corresponding topics (Chien and Chang, 2014). Finally, the topic representation of a test sample is sparse and its maximum entry directly determines the category to which the test sample belongs because the topic is the same as the category. When multiple topic entries of a change message reach the same maximum, the change message is regarded as a multi-purpose one.

In our experiments, change messages of five open source projects are extracted by using the CVSAnalY (Robles et al., 2004) tool. The change message is normalized by WordNet (Miller, 1995) and Gate (Cunningham et al., 2002). The five different projects in the experiment shared the same vocabulary and the estimated model, and moreover the sparse probabilistic representation of software change messages were directly used to assign software changes into Swanson's maintenance categories (Swanson, 1976) by finding the maximum topic entry. The proposed approach is proved capable of classifying changes well through manual validation performed by professional developers. Especially, the multi-category change classification task that improves the recall rate. In summary, the contributions of this paper can be summarized as follows:

- We explore the discovered word-topic distributions learned from labeled change messages and find they provide an ordered probabilistic relationship between relevant words and the categories of software changes. As a result, this overcomes the ambiguity coming from manually subjective weights.
- We explore the discovered topic-document distributions and find a one-to-one correspondence between these discovered topics and change categories. The maximum topic entry directly determines the category to which a change belongs. If multiple topic entries reach the same maximum, this indicates a change is a multi-category one.
- We evaluate our approach on five projects and compare the performance with four baselines. The results indicate that our performing multi-category classification improves the classification performance. As a result, this work provides a more accurate distribution of each category in a project. Besides, we provide a method of cross-project software change classification without the need for re-learning. The different projects share the same vocabulary and the estimated model.

The structure of this paper is as follows. In Section 2 we present the related work of our research, including previous software change classification methods, software change classification rules, topic modeling in mining software repositories (MSR), and PLSA. We describe our research preparation, models and techniques in Section 3. In Sections 4 and 5, we provide the experiment design, results and validation. Then at last in Sections 6 and 7, we discuss the potential threats to our findings and draw a conclusion.

2. Related work

In this section, we discuss related literature from several aspects: previous software change classification methods, software