Information and Software Technology 62 (2015) 67–77

Contents lists available at ScienceDirect

Information and Software Technology

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

Negative samples reduction in cross-company software defects prediction

Lin Chen [a], Bin Fang [a,*], Zhaowei Shang [a], Yuanyan Tang [a,b]

[a] Department of Computer Science, Chongqing University, Chongqing 400030, China
[b] Faculty of Science and Technology, University of Macau, Macau, China

ARTICLE INFO

Article history:
Received 28 April 2014
Received in revised form 10 December 2014
Accepted 30 January 2015
Available online 19 February 2015

Keywords:
Cross-company defects prediction
Software fault prediction
Transfer learning

ABSTRACT

Context: Software defect prediction has been widely studied based on various machine-learning algorithms. Previous studies usually focus on within-company defects prediction (WCDP), but lack of training data in the early stages of software testing limits the efficiency of WCDP in practice. Thus, recent research has largely examined the cross-company defects prediction (CCDP) as an alternative solution.
Objective: However, the gap of different distributions between cross-company (CC) data and within-company (WC) data usually makes it difficult to build a high-quality CCDP model. In this paper, a novel algorithm named Double Transfer Boosting (DTB) is introduced to narrow this gap and improve the performance of CCDP by reducing negative samples in CC data.
Method: The proposed DTB model integrates two levels of data transfer: first, the data gravitation method reshapes the whole distribution of CC data to fit WC data. Second, the transfer boosting method employs a small ratio of labeled WC data to eliminate negative instances in CC data.
Results: The empirical evaluation was conducted based on 15 publicly available datasets. CCDP experiment results indicated that the proposed model achieved better overall performance than compared CCDP models. DTB was also compared to WCDP in two different situations. Statistical analysis suggested that DTB performed significantly better than WCDP models trained by limited samples and produced comparable results to WCDP with sufficient training data.
Conclusions: DTB reforms the distribution of CC data from different levels to improve the performance of CCDP, and experimental results and analysis demonstrate that it could be an effective model for early software defects detection.

1. Introduction

Software testing has become one of the most critical and costly phases in software development as size and complexity of software increases. Defect prediction based on static code metrics focuses on detecting defect-prone modules to help test teams accelerate searching for potential defects [1]. Existing studies [2–4] on this issue have proposed many effective prediction models, but they are usually confined to within-company defect prediction (WCDP), which attempts to train predictors based on historical data to detect future defects within the same source. However, a potential problem that cannot be avoided is that, in the early testing process, resources for practitioners are typically limited and may not be sufficient to build reliable predictors. Additionally, collecting historical training data is also time-consuming and difficult work [5]. On the other hand, many open source defect datasets can easily be found such as the PROMISE repository [6]. Can we use these abundant data to predict our specific software defects?

Cross-company defect prediction (CCDP) is an attractive solution for this issue. Different from previous methods, it tries to use cross-company (CC) datasets to build prediction models to detect defects in within-company (WC) data. Many studies have revealed that it can work as well as or even better than WCDP if CC data are carefully selected [7,8]. One of the most interesting methods is filter technology, for example, the Nearest Neighbor (NN) filter [8] returns the most similar samples from CC data as training samples. When the selected CC data have the same distribution as the WC data, it performs well (as shown in Fig. 1a).

However, since CC data are collected from different development environments or application fields, the labels of CC data may be in conflict with WC data even they are close in distance, which tends to generate false prediction results (Fig. 1b). More

* Corresponding author. Tel.: +86 023 65112784.
E-mail addresses: chenlincqu@cqu.edu.cn (L. Chen), fb@cqu.edu.cn (B. Fang), szw@cqu.edu.cn (Z. Shang), yytang@umac.mo (Y. Tang).

http://dx.doi.org/10.1016/j.infsof.2015.01.014
0950-5849/

specifically, if defect-free WC data have some contradictory defect-prone CC neighbors, they may be result in a high false alarm rate. On the other hand, if defect-prone WC data select negative defect-free CC data, it could lead to a low rate of recall.

If we have partial label information for WC data, these negative training samples can be reduced (Fig. 1c), so that misclassified instances can be rectified and the prediction result is moved into the right direction (Fig. 1d).

To overcome the effect of heterogeneity between CC and WC data, in this study, we propose a novel transfer learning method named Double Transfer Boosting (DTB) algorithm, which employs not only CC data but also a small amount of labeled WC data to improve the performance of CCDP.

The remainder of the paper is organized as follows: Section 2 gives a brief review of existing related work on CCDP. Section 3 describes the proposed DTB model. Section 4 reports experimental datasets and performance measures. Section 5 analyzes the experimental results in relation to the research questions. Section 6 discusses the potential threats to validity before we conclude this paper and suggest future work in Section 7.

2. Related work

As mentioned above, local historical training data from the same company is not always available or is difficult to collect in practice. To address this problem, researchers have turned to CCDP as an alternative solution in the last few years.

An early attempt at CCDP reported by Zimmermann et al. [9] was based on large-scale experiments for 12 real-world applications datasets. To investigate whether the CCDP model is able to work well, 622 cross-project predictions were examined. However, the test results were somewhat discouraging with a low success rate of 3.4%. Thus researchers warned that CCDP is still a serious problem. Meanwhile, an interesting phenomenon also has been found: Firefox is a strong defect predictor for Internet Explorer. However, this does not work as well as the opposite direction, even though Firefox and IE are similar applications.

Turhan et al. conducted CCDP experiments using 10 projects collected from two different companies including NASA and SOFTLAB [8]. They found that defect predictors built on all available CC data dramatically increased defect detection ability, but with unacceptably high false alarm rates. They explained that false alarms were raised by irrelevancies in CC data. Furthermore, they proposed a Nearest-Neighbor filter method (named NN filter) to select training data close to WC data. This method achieved better performance than using the raw CC data, but still is worse than WCDP.

Similar to Zimmermann et al.'s work, He et al. investigated CCDP with 34 datasets obtained from 10 open source projects [7]. Rather than using CC data directly, a schema of dataset selection was added before building the prediction models. To assess the performance of CCDP, they created a criterion (recall $\geqslant$ 70% and precision $\geqslant$ 50%) for judging successful predictors. Experimental results indicated that their method achieved promising results, i.e., that 18 out of 34 cases met the criteria, showing that WCDP does not always perform better than CCDP. They pointed out that one potential way to predict defects in projects without local data is to learn predictors from data of other projects.

Rahman et al. found a different way to evaluate the feasibility of CCDP. They only inspected a partial set of files to find out defects and introduced a new performance measure called area under the cost effectiveness curve (AUCEC) [10]. They also drew an optimistic conclusion from experimental results, i.e., that CCDP is no worse than WCDP in terms of AUCEC. However, the process metrics used in their defect model at file level are difficult to obtain in the open repository.

Transfer learning is another solution for CCDP. Ma et al. [11] considered that predictions should be related to the distributional characteristics of datasets. They proposed a novel CCDP model using data transfer method called Transfer Naive Bayes (TNB). As a type of instance-transfer[1] in transfer learning approaches [12], TNB was trained based on re-weighted CC data according to the distribution of WC data. Their results showed that CCDP model built on datasets collected from NASA could find defects in SOFTLAB effectively.

More recently, He et al. shared the same idea of CC data selection based on data similarity to improve the performance of CCDP [13]. Unlike the NN filter [8] picking out some individual instances, their model selected closest cross training datasets with similar distributions. The distance of distribution between CC data and WC data was measured according to the classification accuracy. Feature subset selection was also employed to remove unstable features. Empirical study on datasets including open-source and proprietary projects was conducted to prove the feasibility of CCDP. The results indicated that their data selection method could perform relatively better than the NN filter.

Zhang et al. built a universal defect prediction model for CCDP from diverse datasets [14]. To address variations of software metrics between different projects, the original metrics values were discretized by context-aware rank transformation according to similar degree of context factors, then, the universal model was built on the transformed metrics in the same scales. The results of rank transformation showed a performance comparable to log transformation. When compared with WCDP, their universal model improved predictive performance with higher Recall and AUC values, but it also suffered from a higher false positive rate.

The studies above all focus on using only CC data to build proper prediction models. Recently, Turhan et al. introduced a mixed model for CCDP for if limited amounts of WC data exist [15]. In their study, the within and cross data were combined together as training data, and the Naive Bayes classifier with NN filter was applied to build the prediction model. Their results implied that the mixed model could be comparable to WCDP. However, their study is a post-facto method because they randomly mixed different amounts of (from 10 to all of NN filtered samples) CC data with a smaller ratio of WC data to train the predictor and only reported the best performance. Furthermore, how to select the best subsets of CC data is still unanswered in their paper. Despite these issues, the paper suggests that even small amounts of WC data could improve the results of CCDP.

3. Methodology

In this study, the proposed Double Transferring Boosting (DTB) algorithm is built based on mixed training samples consisting of CC data and partial WC data. However, rather than using a simple combination of these data as the mixed model [15], we employ two-levels of transfer learning methods to "reshape" CC data to build a high-performance predictor. Its main steps are as follows: First, training datasets from other sources are preprocessed by the NN filter and data oversampling (SMOTE). Next, these preprocessed data are re-weighted by the first transfer method with data gravitation. Finally, limited amounts of labeled WC data (i.e., 10% of the total WC data) are mixed with re-weighted data to build the prediction model using the transfer boosting learning algorithm. Fig. 2 gives the framework of the proposed model.

[1] According to Ref. [12], main approaches to transfer learning includes instance-transfer, feature-transfer, parameters-transfer and relational-knowledge-transfer. The instance-transfer method refers to learn the knowledge from different source data by re-weighting labeled instances.