A Multi-Source TrAdaBoost Approach for Cross-Company Defect Prediction

Xiao Yu[1], Jin Liu[1*], Mandi Fu[2,3], Chuanxiang Ma[2,3*],Guoping Nie[4], Xu Chen[1]
[1]State Key Lab. of Software Engineering, Computer School, Wuhan University, Wuhan, China
[2]School of Computer Science and Information Engineering, HuBei University, Wuhan, China
[3]Educational Informationalization Engineering Research Center of HuBei Province, Wuhan, China
[4]School of Mathematics, Huazhong University of Science and Technology, Wuhan, China
*Corresponding author email: jinliu@whu.edu.cn,mxc838@hubu.com

***Abstract*—Cross-company defect prediction (CCDP) is a practical way that trains a prediction model by exploiting one or multiple projects of a source company and then applies the model to target company. Unfortunately, larger irrelevant cross-company (CC) data usually makes it difficult to build a prediction model with high performance. On the other hand, brute force leveraging of CC data poorly related to within-company (WC) data may decrease the prediction model performance. To address such issues, this paper introduces Multi-Source TrAdaBoost algorithm, an effective transfer learning approach to perform CCDP. The core idea of our approach is that: 1) employ limited amount of labeled WC data to weaken the impact of irrelevant CC data; 2) import knowledge not from one but from multiple sources to avoid negative transfer. The experimental results indicate that: 1) our proposed approach achieves the best overall performance among all tested CCDP approaches; 2) only 10% labeled WC data is enough to achieve good performance of CCDP by using our proposed approach.**

Keywords*—software defect prediction;cross-company defect prediction; transfer learning; Multi-Source TrAdaBoost

I. INTRODUCTION

Software defect prediction is one of the most important software quality assurance techniques. It aims to detect the defect proneness of new software modules via learning from defect data. So far, many efficient software defect prediction approaches [1-6] have been proposed, but they are usually confined to within company defect prediction (WCDP). WCDP works well if sufficient data is available to train a defect prediction model. However, it is difficult for a new company to perform WCDP if there is limited historical data. Cross-company defect prediction (CCDP) is a practical approach to solve the problem. [1]It trains a prediction model by exploiting one or multiple projects of a source company and then applies the model to target company [7].

Most existing CCDP approaches [7-14] focus on using only cross-company (CC) data to build a proper prediction model. Unfortunately, larger irrelevant CC data usually makes it difficult to build a prediction model with high performance [15]. In fact, if there is limited amount of labeled WC data, the data is not enough to perform WCDP, but it may help a lot to improve the performance of CCDP. Another scenario is that companies may already have their defect prediction models in place and making use of CC data may improve the performance of models [16].

The challenges of performing CCDP with limited amount of labeled WC data usually include:

1) How to weaken the impact of irrelevant CC data to improve the performance of CCDP.

The ability to transfer knowledge from a source company to a target company depends on how they are related. The stronger the relationship, the more usable will be CC data. The performance of CCDP is generally poor because of larger irrelevant CC data. The irrelevant data has bad effects on the prediction outcome [17].

2) How to avoid negative transfer when leveraging multiple cross-companies data.

Brute force leveraging of CC data poorly related to WC data may decrease the prediction model performance. Lin et al. developed the double transfer boosting (DTB) approach [15] for CCDP. DTB approach merges all CC data as a source, relies on only the source, and therefore is intrinsically vulnerable to negative transfer.

Considering the above challenges, this paper introduces Multi-Source TrAdaBoost algorithm [17], an effective transfer learning approach to perform CCDP. We call the proposed approach for CCDP as MSTrA. The core insight of MSTrA is that: 1) in order to narrow the distribution gap between CC data and WC data, MSTrA firstly uses NN filter [10] to select the *k* most similar CC data to each WC data and then use data gravitation [18] for reweighting the whole distribution of CC data to fit WC data; 2) MSTrA then trains and combines a set of weak prediction models for building a stronger ensemble defect prediction model using not only reweighted CC data but also limited amount of WC data; 3) in each training round, MSTrA transfers knowledge from multiple sources and reduces the weights of irrelevant CC data continuously.

To assess the MSTrA approach, this paper explores the following research questions.

RQ1: How effective is our proposed MSTrA approach when comparing to other approaches for CCDP?

DOI reference number: 10.18293/SEKE2016-120

RQ2: How much labeled WC data is enough to help the prediction model achieve better performance by using our proposed MSTrA approach?

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 describes the proposed MSTrA approach for CCDP. Section 4 demonstrates the experimental results. Finally, Section 5 addresses the conclusion and points out the future work.

II. Related Work

In this section, we briefly review the existing cross-company and cross-project defect prediction approaches. These approaches can be categorized into two main types: defect prediction using only CC data [7-14], defect prediction using not only CC data but also limited amount of labeled WC data [15-16].

A. Defect prediction using only CC data

In order to solve the problem that the new companies have too limited historical data to perform WCDP well, the cross-project and cross-company defect prediction appeared.

Briand et al.[8] used logistic regression and MARS models to learn a defect predictor, which is also the earliest work on CCDP. Zimmermann et al. [9] studied CCDP models on 12 real-world applications datasets. Their results indicate that CCDP is still a serious challenge. Turhan et al. [10] investigated the applicability of CC data for building localized defect predictors using 10 projects collected from two different companies including NASA and SOFTLAB. And they have proposed a nearest neighbor (NN) filter to select CC data. He et al. [11] investigates defect predictions in the cross-project context focusing on the selection of training data. Furthermore, they proposed an approach to automatically select suitable training data for projects without historical data so that the results of their experiments are comparable with WCDP, which indicated that some approach of CCDP can comparable to WCDP. They noted that learning predictors using the data from other projects can be a potential way to defect prediction without any historical data. In order to find data for quality prediction, Peters et al. [12] introduced the Peters filter to select training data via the structure of other projects. They compared the filter with two other approaches for quality prediction to assess the performance of the Peters filter, and found that 1) WCDP are weak for small data sets; 2) the Peters filter + CCDP builds better and more useful predictors. Zhang et al. [13] proposed sample-based methods for software defect prediction. For a large software system, they could select and test a small percentage of modules, and then built a defect prediction model to predict defect-proneness of the rest of the modules. They described three methods for selecting a sample and proposed a novel active semi-supervised learning method ACoForest to facilitate the active sampling. The results showed that the proposed methods are effective and have potential to be applied to industrial practice. Ma et al. [14] proposed a novel algorithm called Transfer Naive Bayes (TNB) to transfer cross-company data information into the weights of the training data and then build the predictor based on re-weighted CC data. The results indicated that TNB is more accurate in terms of AUC, within less runtime than the state of the art methods and can effectively achieve the CCDP task. The heterogeneous CCDP (HCCDP) task is that the source and target company data is heterogeneous. Jing et al. [7] provided an effective solution for HCCDP. They proposed a unified metric representation (UMR) for the data of source and target companies and introduced canonical correlation analysis (CCA), an effective transfer learning method, into CCDP to make the data distributions of source and target companies similar. Results showed that their approach significantly outperforms state-of-the-art CCDP methods for HCCDP with partially different metrics and for HCCDP with totally different metrics, their approach is also effective.

The approaches above are focus on using only CC data to build predictors. Considering there is limited amount of labeled WC data, the data is not enough to perform with company defect prediction, but it may help a lot to improve the performance of CCDP.

B. Defect prediction with limited amount of labeled WC data

Turhan et al. [16] introduced a mixed model of within and cross data for CCDP to investigate the merits of using mixed project data for binary defect prediction. Results showed that when there is limited project history, mixed model for CCDP can achieve good performance which can be comparable to WCDP. It provided a new idea to CCDP that the use of a small amount of labeled WC data would be very valuable to improve the performance of CCDP.

Lin et al. [15] introduced a novel approach named Double Transfer Boosting (DTB) to narrow the gap of different distributions between CC data and WC data and to improve the performance of CCDP by reducing negative samples in CC data. However, it merges all CC data as one source and the result only relies on the single source so that it is prone to negative transfer, which is exactly what we will solve in this paper.

III. Methodology

In this section, we present our MSTrA approach for CCDP. Its main steps are as follows: 1) in order to narrow the distribution gap between CC data and WC data, MSTrA first uses NN filter [10] to select the k most similar CC data to each WC data and then uses data gravitation [18] for reweighting the whole distribution of CC data to fit WC data; 2) MSTrA mixes limited amount of labeled WC data with reweighted CC data to build the prediction model by using Multi-Source TrAdaBoost algorithm.

A. Data Preprocessing

Previous work [10] found that using raw CC data directly would increase false alarm rates due to irrelevant instance in CC data, so several data preprocessing works should be done before building the prediction model. To decrease the negative effect of the irrelevant instance in CC data for building the prediction model, we employ NN filter proposed by Turhan et al. [10] to form the training set. Based on the widely used classification method KNN algorithm, NN filter can find out the most similar $K \times N$ instances from CC data while N is the number of WC instances and K is the parameter of the KNN method. Note that duplicate instances may exist in this filtered dataset, as some instances in WC data may have some common neighbors in CC data. Thus, the final filtered CC training data can be formed by using only unique ones.

Then we apply the data gravitation method [18] to change the entire distribution of CC data. Suppose that an instance x_i can be described by $x_i=\{a_{i1},a_{i2},...,a_{ik}\}$,where a_{ij} is the j-th attribute value of the i-th instance and k is the number of the attributes.

(1) We compute two vectors, Max= $\{max_1, max_2,…, max_k\}$ and Min= $\{min_1, min_2, …, min_k\}$ to represent the attribute value distribution of WC data, where max_i is the maximum value of the i-th attribute, min_i is the minimal value of the i-th attribute.

(2) For each instance x_i in CC data, the degree s_i of similarity to WC data is computed according to Eq.(1)

$$s_i=\sum_{i=1}^{k} h(a_{ij}) \tag{1}$$

where a_{ij} is the j-th attribute value of the instance x_i, $h(a_{ij}) = 1$, if $min_j \leqslant a_{ij} \leqslant max_j$; otherwise, $h(a_{ij}) = 0$.

(3) The weight w_i of instance x_i in CC data can be calculated by Eq.(2) according to the formulation of data gravitation [18].

$$w_i=s_i/(k-s_i+1)^2 \tag{2}$$

where k is the number of the attributes.

According to this formula, the weight w_i of instance x_i shows the similarity of x_i to WC data, and the greatest w_i will be assigned when $s_i = k$.

Though the above steps, the entire distribution of CC data is reweighted to be close to WC data.

B. *Multi-Source TrAdaBoost Approach*

Let $D^{SK}=\{(x_1^{S1},c_1^{S1}),…,(x_n^{SK},c_n^{SK})\}$ be the k-th cross-company data, where n is the number of instances in the k-th cross company data, $c_i^{SK}\in\{\text{true, false}\}$ is the class label of instance x_i^{SK}. Let $D^T=\{(x_1^T,c_1^T),…,(x_m^T,c_m^T)\}$ be limited amount of labeled WC data, where m is the number of instances in labeled WC data, c_i^T is the class label of instance x_i^T. During NN filter and data gravitation, filtered CC data $D_{S1},…D_{SN}$ and labeled WC data D_T are assigned different weight according Eq.(2).

In each training round, combine the k-th cross-company data and labeled WC data to train a candidate weak prediction model. In our paper, we choose Naïve Bayes [19] as the base prediction model due to its effectiveness in defects prediction [20]. The final weak prediction model $\boldsymbol{f_t}(x)$ in t-th iteration is one of the candidate weak prediction models which has the minimal prediction error on labeled WC data. In other words, every weak prediction model is selected from CC data that appears to be the most closely related to WC data. The prediction error function is defined according Eq.(3).

$$\varepsilon_t=\sum_{i=1}^{m}\frac{w_i^t|\boldsymbol{f_t}(x_i)-c_i|}{\sum_{i=1}^{m}w_i^t} \tag{3}$$

$$\text{Set}\quad \beta_t=\frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t} \tag{4}$$

In this way, we import knowledge not from one but from multiple sources, thus decreasing the risk for negative transfer. At t-th iteration, the instances in WC data are given more importance if the instances are misclassified. They are believed to be the "most informative" for the next round, so the weight of the misclassified instances are increased according Eq.(5).

$$w_i^T=w_i^T\, e^{\beta_t|f_t(x_i^T)-c_i^T|} \tag{5}$$

The instances in CC data are given less importance if the instances are misclassified. They are believed to be the most dissimilar to WC data, so the weight of the misclassified instances are decreased according Eq.(6) in order to weaken their impacts in the next round through multiplying the Hedge(β) defined in Eq.(7).

$$w_i^{SK}=w_i^{SK}\, e^{-\beta_s|f_t(x_i^{SK}-c_i^{SK})|} \tag{6}$$

$$\beta_s=\frac{1}{2}\ln(1+\sqrt{2\ln\frac{n_s}{M}}) \tag{7}$$

After several iterations, the instances in CC data that fit WC data will have larger training weights, while the instances in CC data that are dissimilar to WC data will have lower weights. The instances in CC data with larger training weights intend to build a better prediction model. The final prediction model $F(x)$ can be expressed as follows:

$$F(x)=\text{sign}(\sum_t \beta_t \boldsymbol{f_t}(x)) \tag{8}$$

Algorithm 1 presents the pseudo-code of MSTrA approach to perform CCDP.

Algorithm 1. MSTrA approach

```
Input: filtered CC data $D^{S1},…, D^{SN}$, limited amount of labeled WC data $D^T$, and the maximum number of iterations M
Output: a prediction model $F(x)$
1.  Initialize a weight vector ($\mathbf{w}^{S1},…,\mathbf{w}^{SN},\mathbf{w}^T$) using Eq.(2)
2.  for t=1,…,M do
3.      Empty the set of candidate weak prediction models
4.      Normalize to 1 the weight vector ($\mathbf{w}^{S1},…,\mathbf{w}^{SN},\mathbf{w}^T$)
5.      for k=1,…,N do
6.          Train the candidate weak prediction model $\boldsymbol{f_t}^K(x)$ over the combined data $D^{SK} \cup D^T$, using weight ( $\mathbf{w}^{SK},\mathbf{w}^T$)
7.          Compute the error of $\boldsymbol{f_t}^K(x)$ on $D^T$ using Eq.(3)
8.      end for
9.      Find the weak prediction model $\boldsymbol{f_t}(x)$ which has the minimal error
10.     Update weights vector ($\mathbf{w}^{S1},…,\mathbf{w}^{SN},\mathbf{w}^T$) for the next round using Eq.(5) and Eq.(6)
11. end for
12. return $F(x)=\text{sign}(\sum_t \beta_t \boldsymbol{f_t}(x))$
```

IV. Experiments

In this section, we evaluate our proposed MSTrA approach to perform CCDP empirically. We first introduce the experiment dataset and the performance measures. Then, in order to investigate the performance of MSTrA, we perform some empirical experiments to find answers to the research questions mentioned above.

A. Data set

In this experiment, we employ 15 available and commonly used datasets which can be obtained from PROMISE [21]. The 15 datasets have the same 20 attributes, so we can apply all attribute information directly. Table 1tabulates the details about the datasets.

TABLE I. Details of Experiment Dataset

Project	*Examples*	*%Defective*	*Description*
ant	125	16	Open-source
arc	234	11.5	Academic
camel	339	3.8	Open-source
elearn	64	7.8	Academic
jedit	272	33.1	Open-source
log4j	135	25.2	Open-source
lucene	195	46.7	Open-source
poi	237	59.5	Open-source
prop	660	10	Proprietary
redaktor	176	15.3	Academic
synapse	157	10.2	Open-source
systemdata	65	13.8	Open-source
tomcat	858	9	Open-source
xalan	723	15.2	Open-source
xerces	162	47.5	Open-source

B. Performance measures

In the experiment, we employ three commonly used performance measures including *pd*, *pf* and *g-measure*. They are defined in Table 2 and summarized as follows.

TABLE II. Performance measures

		Actual	
		yes	*no*
Predicted	*yes*	*TP*	*FP*
	no	*FN*	*TN*
pd	$\frac{TP}{TP+FN}$		
pf	$\frac{FP}{FP+TN}$		
g-measure	$\frac{2*pd*(1-pf)}{pd+(1-pf)}$		

● Probability of detection or *pd* is the measure of defective modules that are correctly predicted within the defective class. The higher the *pd*, the fewer the false negative results.

● Probability of false alarm or *pf* is the measure of non-defective modules that are incorrectly predicted within the non-defective class. Unlike *pd*, the lower the *pf* value, the better the results.

● *g-measure* is a trade-off measure that balances the performance between *pd* and *pf*. A good prediction model should have high *pd* and low *pf*, and thus leading to a high *g-measure*.

C. Results for Q1

In order to confirm whether the MSTrA approach can perform better than other CCDP approaches, we compared our approach with four state-of-the-art CCDP approaches. More details are provided below:

● **NN filter** [10] is based on the widely used classification method K-Nearest Neighbors (KNN) algorithm to filter irrelevant CC data. It can find out the most similar *K*×*N* instances from CC data while *N* is the number of instances in WC data and *K* is the parameter of the KNN method. In our experiment, we choose *K* as 10. After NN filter, Naïve Bayes classifier is chosen as the basic prediction model.

● **TNB** [14] first reweights the CC data by the data gravitation method, then builds a transfer Naïve Bayes classifier on reweighted CC data.

● **NN+WC** (Nearest-Neighbor filter with WC data) [16] mixes *p%* WC data with CC data which was processed by NN filter as training data. In our experiment, we choose *p* as 10. Then Naïve Bayes classifier is chosen as the basic prediction model on the training data.

● **DTB** [15] first uses NN filter ,SMOTE [22] and data gravitation to process CC data. Then limited amount of labeled WC data and reweighted CC data are mixed to build prediction model using the transfer boosting algorithm.

In every experiment, one dataset is selected as WC data and the rest are regarded as CC data to conduct the experiment. The CC data is considered as basic training data which will be adjusted in every experiment. WC data will be randomly divided into two parts: 10% labeled WC data as training data with CC data in our MSTrA approach, DTB approach and NN+WC approach, and the remainder is taken as test data for all CCDP approaches in order to be fair. Then the values of the performance measures for our MSTrA approach, DTB approach and NN+WC approach are calculated.

The comparison results are summarized in Table 3 with three performance measures mentioned above. It shows that the NN approach often achieves the best *pd* but the worst *pf* so that it usually ends up with low *g-measure* value. The performance of NN+WC approach seems sometimes have lower *pf* than the NN approach but mostly have similar result with NN approach. The effect of WC data seems not very obvious.

It's very clear that the transfer learning models including TNB, DTB and MSTrA have lower *pf* than the other two models. In the aspect of *pf* value, the MSTrA approach reduces the *pf* to a large extent. The *pf* results of 7 projects are better than others. On more than half tests, MSTrA achieves higher *g-measure* than other models.

In total, the MSTrA approach has acceptable *pd* value and can obtain better *pf* values in most experiments we conducted, and it almost always achieve the higher *g-measure* value than other models. In other words, the MSTrA approach outperforms other CCDP approaches, therefore it can be an effective approach for CCDP.

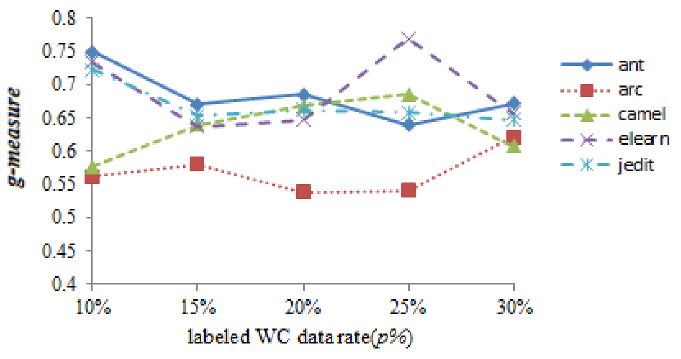
TABLE III. PD,PF AND G-MEASURE VALUES, THE REASULTS OF FIVE APPROACH

No.	Test data	MSTrA			DTB			TNB			NN			NN+WC		
		PD	PF	G	PD	PF	G	PD	PF	G	PD	PF	G	PD	PF	G
1	ant	**0.823**	0.313	**0.749**	0.811	0.370	0.709	0.819	0.524	0.602	0.371	**0.270**	0.492	0.399	0.275	0.099
2	arc	0.409	**0.106**	0.561	0.605	0.272	**0.661**	**0.745**	0.413	0.655	0.745	0.629	0.495	0.807	0.648	0.488
3	camel	0.417	**0.066**	0.576	0.487	0.300	0.574	0.564	0.290	**0.629**	**0.784**	0.709	0.424	0.784	0.710	0.423
4	elearn	0.75	0.283	0.733	0.675	0.243	0.713	**1.000**	0.393	0.756	0.900	0.383	0.724	0.900	0.383	0.724
5	jedit	0.646	0.181	**0.722**	0.568	0.237	0.651	0.475	**0.168**	0.605	**0.932**	0.616	0.544	**0.932**	0.628	0.532
6	log4j	0.576	0.247	0.652	0.611	0.234	0.679	0.635	**0.134**	**0.733**	**0.936**	0.706	0.444	**0.936**	0.709	0.444
7	lucene	0.722	0.474	0.608	0.581	0.382	0.598	0.580	**0.221**	**0.665**	**0.762**	0.554	0.563	0.750	0.548	0.564
8	poi	0.551	0.241	**0.638**	0.632	0.415	0.607	0.416	**0.228**	0.540	**0.910**	0.678	0.475	**0.910**	0.684	0.469
9	prop-6	0.655	**0.299**	**0.678**	0.670	0.331	0.669	0.529	0.336	0.589	**0.860**	0.635	0.512	**0.860**	0.628	0.519
10	redactor	0.591	**0.336**	**0.625**	0.616	0.679	0.422	0.634	0.513	0.550	**1.000**	0.899	0.184	**1.000**	0.891	0.196
11	synapse	0.786	**0.285**	**0.748**	0.871	0.490	0.643	0.775	0.422	0.662	**0.935**	0.777	0.360	**0.935**	0.781	0.355
12	system	0.75	0.490	0.607	0.717	0.340	0.687	0.563	**0.260**	0.640	**0.817**	0.341	**0.730**	**0.817**	0.341	**0.730**
13	tomcat	0.418	**0.163**	0.558	0.712	0.396	**0.653**	**0.914**	0.592	0.564	0.632	0.380	0.614	0.690	0.359	0.660
14	xalan	0.458	**0.175**	0.589	0.654	0.400	**0.625**	0.604	0.356	0.623	**0.961**	0.679	0.481	**0.961**	0.685	0.474
15	xerces	**0.586**	0.392	**0.595**	0.370	0.274	0.490	0.319	**0.268**	0.444	0.437	0.631	0.400	0.437	0.631	0.400
	Average	0.609	**0.270**	**0.643**	0.639	0.358	0.625	0.638	0.341	0.617	0.799	0.592	0.496	**0.808**	0.593	0.472

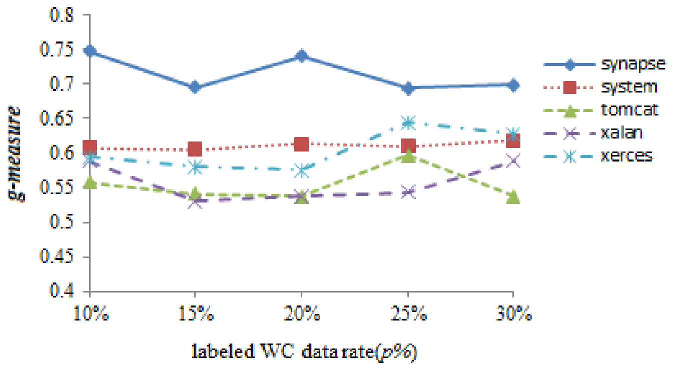
D. Results for Q2

In order to confirm how much labeled WC data is enough to help the prediction model achieve better performance by using our proposed MSTrA approach, we randomly select *p%* (*10%,15%,20%,25%,30%*) WC data as training data with CC data, and the remainder data is taken as test data. In every experiment, one dataset is selected as WC data and the rest are regarded as CC data. We repeated our proposed MSTrA approach 20 times in every experiment to avoid sample bias. Then the mean values of *g-measure* for MSTrA are recorded in Figure 1.

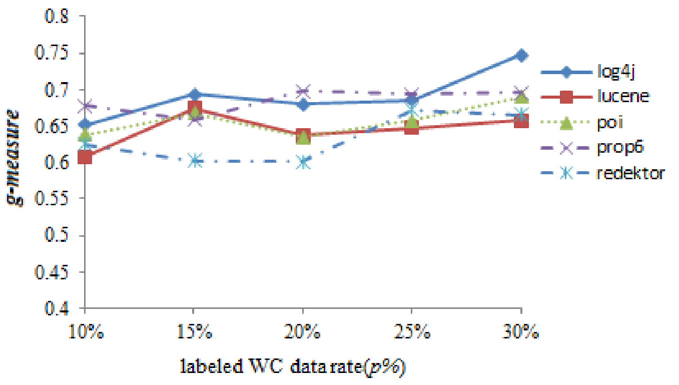
As shown in Figure 1, using only 10% labeled WC data with CC data is enough to achieve good performance by using our MSTrA approach. In particular, better performance is achieved on ant, elearn, jedit and xlalan datasets when using only 10% labeled WC data. There is no significant improvement on arc, synapse, system, xerces, tomcat, lucene, poi and prop6 datasets when incrementally adding labeled WC data. In total, we only need limited amount of labeled WC data (i.e. 10% is enough) to achieve good performance of CCDP by using our proposed MSTrA approach. Therefore, a new company can exploit our proposed MSTrA approach to perform CCDP at the early stages of development activities if there is limited historical data.



(a) ant, arc, camel, elearn, jedit



(b) synapse, system, tomcat, xalan, xerces



(c) log4j,lucene,poi,prop6,redektor

Fig. 1. G-measure performances with different size of labeled WC data

V. CONCLUSION AND FUTURE WORK

In this paper, we address the issues of how to weaken the impact of irrelevant CC data and how to avoid negative transfer when leveraging multiple source companies data to improve the performance of CCDP. We introduce Multi-Source TrAdaBoost algorithm to improve the performance of CCDP. First of all, we use NN-filter and data gravitation for reweighting the whole distribution of CC data to fit WC data.

Then we train and combine a set of weak prediction models for building a stronger ensemble defect prediction model using not only reweighted CC data but also limited amount of labeled WC data. In each training round, we transfer knowledge from multiple sources to avoid negative transfer and reduce the weights of irrelevant instances in CC data to weaken the impact of irrelevant CC data continuously.

We conduct experiments on the 15 datasets to evaluate the performance of the proposed approach. The experimental results indicate that the proposed approach can effectively weaken the impact of irrelevant data and avoid negative transfer to improve the performance of CCDP. The proposed MSTrA approach is an effective approach for CCDP.

In the future, we would like to validate the generalization ability of our approach on more company data.

ACKNOWLEDGMENT

This work is partly supported by the grants of National high technology research and development program (863 Program 2012AA011204-01), National Natural Science Foundation of China (61070013, 61300042, U1135005, 71401128), the Fundamental Research Funds for the Central Universities (No. 2042014kf0272, No. 2014211020201) and Natural Science Foundation of HuBei (2011CDB072).

REFERENCES

[1] K. Elish and M. Elish, "Predicting defect-prone software modules using support vector machines," Journal of Systems and Software,2008, 81(5):649-660.

[2] J. Zheng,"Cost-sensitive boosting neural networks for software defect prediction," Expert Systems with Applications, 2010, 37(6):4537-4543.

[3] Z. B. Sun, Q. B. Song, and X. Y. Zhu, "Using coding based ensemble learning to improve software defect prediction," IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 2012,42(6):1806-1817.

[4] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," IEEE Transactions on Reliability,2013,62(2):434-443.

[5] M. Liu, L. Miao, and D. Zhang, "Two-stage cost-sensitive learning for software defect prediction," IEEE Transactions on Reliability,2014, 63(2):676-686.

[6] X. Y. Jing, S. Ying, Z. W. Zhang, S. S. Wu, and J. Liu, "Dictionary learning based software defect prediction," in: Proc. of the 36th International Conference on Software Engineering (ICSE), 2014,pp. 414-423.

[7] Xiaoyuan Jing et al, "Heterogeneous Cross-Company Defect Prediction by Unified Metric Representation and CCA-Based Transfer Learning," in: Proc. of the 10th Joint Meeting on Foundations of Software Engineering, 2015, pp 496-507.

[8] Briand L C, Melo W L, Wust J, "Assessing the applicability of fault-proneness models across object-oriented software projects," IEEE Transactions on Software Engineering, 2002, 28(7): 706-720.

[9] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, B. Murphy, "Cross-project defect prediction: a large scale experiment on data vs. domain vs. process," in: Proc. of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ACM, 2009, pp. 91–100.

[10] B. Turhan, T. Menzies, A.B. Bener, J. Di Stefano, "On the relative value of cross company and within-company data for defect prediction," Empirical Softw. Eng. 2009,14 (5): 540–578.

[11] Z. He, F. Shu, Y. Yang, M. Li, Q. Wang, "An investigation on the feasibility of cross-project defect prediction," Autom. Softw. Eng. 2012,19 (2) 167–199.

[12] Peters F, Menzies T, Marcus A, "Better cross company defect prediction," In: Proc. of the 10th International Workshop on Mining Software Repositories, San Francisco, CA, 2013, 409-418.

[13] Zhang F, Mockus A, Keivanloo I, et al, "Towards Building a Universal Defect Prediction Model," In: Proc. of the 11th Working Conference on Mining Software Repositories, 2014, 182-191.

[14] Y. Ma, G. Luo, X. Zeng, A. Chen, "Transfer learning for cross-company software defect prediction," Inform. Softw. Technol. 2012,54 (3): 248–256.

[15] Chen L, Fang B, Shang Z, et al. "Negative samples reduction in cross-company software defects prediction," Information and Software Technology, 2015, 62: 67-77.

[16] B. Turhan, A. Tosun Mısırlı, A. Bener, "Empirical evaluation of the effects of mixed project data on learning defect predictors," Inform. Softw. Technol. 2013, 55 (6):1101–1118.

[17] Yao Y, Doretto G, "Boosting for transfer learning with multiple sources," in: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),2010: 1855-1862.

[18] L. Peng, B. Yang, Y. Chen, A. Abraham, "Data gravitation based classification," Inform. Sci. 2009,179 (6): 809–819.

[19] D.D. Lewis, "Naive (Bayes) at forty: the independence assumption in information retrieval" Machine Learning: ECML-98, Springer, 1998, pp. 4–15.

[20] T. Hall, S. Beecham, D. Bowes, D. Gray, S. Counsell, "A systematic literature review on fault prediction performance in software engineering," IEEE Trans.Softw. Eng. 2012,38 (6): 1276–1304.

[21] G. Boetticher, T. Menzies, T. Ostrand, The PROMISE Repository of Empirical Software Engineering Data, 2007 <http://promisedata.org/repository>.

[22] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," J. Artif. Intell. Res. 2002,16 :321–357