Cross-Project Software Defect Prediction Using Feature-Based Transfer Learning

He Qing, Li Biwen, Shen Beijun*
School of Software
Shanghai Jiao Tong University
Shanghai 200240, China
{heqing1234567, bjshen}@sjtu.edu.cn

Yong Xia
IBM Client Innovation Center China,
Shanghai 200433, China
xiayong@cn.ibm.com

ABSTRACT
Cross-project defect prediction is used as an effective means of predicting software defects when data shortage exists in the early phase of software development. Unfortunately, the precision of cross-project defect prediction is usually poor, mainly because of the difference between source and target projects. This paper proposes a new cross-project defect prediction approach (TrCPDP) using feature-based transfer learning to solve issues caused by project differences. The core insight of TrCPDP is to (1) filter and transfer highly-correlated data based on data samples of target projects, and (2) evaluate and choose learning schemas for transferring data sets. Models are then built for predicting defects in target projects. We have also conducted an evaluation of the proposed approach on PROMISE datasets. The evaluation results show that, with our proposed approach for cross-project defect prediction, F-measure of 81.8% of projects and AUC of 54.5% projects are improved. It also achieves similar f-measure and AUC as some within-project defect prediction approaches.

Categories and Subject Descriptors
D.2.8 **[Software Engineering]**: Metrics—*Performance measures, Process metrics, Product metrics*. D.2.9 **[Software Engineering]**: Management –*Software quality assurance (SQA)*

General Terms
Management, Measurement, Reliability, Experimentation

Keywords
cross-project defect prediction; transfer learning; feature-based transfer.

1. INTRODUCTION
Software Defect Prediction is one of the most important software quality assurance techniques. It utilizes basic software features (e.g. average method complexity, cohesion amongst classes, etc.) and previously discovered defects to predict potential defects. The complexity of source code is one of the most important prominent indicators for such models. Besides, code churn information, change history, and structure of software development organizations are also taken into consideration. For example, Akiyama[1] proposes that the number of software defects in the early software development phase has a relation with the lines of code. The equation D = 4.86 + 0.018L holds, showing that there are approximately 22.86 defects per thousand lines of code.[1]

So far, many software defect prediction approaches have been proposed and most are effective when applied to one single project, which is also called *within-version defect prediction*. Due to its high prediction accuracy, within-version defect prediction is largely adopted in industry. Current defect prediction models are all built using historical data from projects, and their defects can be predicted based on these labeled data samples. Here, the features extracted from project dataset are taken as input of prediction model. Then potential defects remained in this project are predicted as output. Because of similar characteristics of different modules in a single project, within-version prediction model usually has good performance. Another defect prediction setting is, analyzing software defects in previous versions to predict the quality of its subsequent version. We call this *within-project defect prediction*. Within-project defect prediction also has good performance.

Defect prediction works well if models are trained with a sufficiently large amount of data samples. However, projects may lack the data needed to build such predictors early in the life cycle. Prior work assumed that relevant training data was sufficient. In practice, training data is often scarce, either because a project is too small or it is in its first release, for which no past data exists[2] In these cases, it is impossible to make automated predictions. A practical approach can then be: leveraging a model from other projects to predict defects in target project. For new projects or projects with limited training data, it is feasible to train a prediction model by using sufficient training data from existing source projects, and then apply the model to some target projects. We call this *cross-project defect prediction*.

Cross-project defect prediction has following main advantage:

Many projects lack data samples in the very beginning, which causes prediction model cannot be built. But cross-project defect prediction will not be affected by this problem, because other project dataset can be used to build defect predictor for target project.

However, compared with within-version defect prediction and within-project defect prediction, accuracy of cross-project prediction is very low. The main reason is that there are differences in features and datasets among various projects, which introduce irrelevant or

Conference'15, Month 1–2, 2015, City, State, Country.

*corresponding author

redundant information relative to defect prediction. The specific questions that we address here are:

(1) How to reduce the divergence in data distribution between source dataset and target dataset?

Performance of cross-project defect prediction is generally poor, mainly because of divergence in data distribution between the source and target data samples. Some researchers have studied one cross-project defect prediction model on a large scale. For 12 real-world applications, they ran 622 cross-project predictions. The results indicated that cross-project prediction is a serious challenge, i.e., simply using models from projects in the same domain or with the same process does not lead to accurate predictions [3].Out of these 622 non-trivial cross-project combinations, the model achieved relatively good results for only 21 combinations. What's more, some characteristic of cross-project defect prediction is especially interesting. Taking IE and Firefox as an example, they share similar features and components. However, they are implemented with different development process and tools. The results showed that Firefox was an effective defect predictor for IE, but IE does not predict Firefox. Cross-project defect prediction relies on feature dataset. Many factors, incl. coding styles from different developers, code structure and code complexity, can bring difficulty to cross-project defect prediction[4] .

(2) How to reduce the irrelevant data to lower down the false alarm rate?

When large amount of project data samples are used to predict defects, only a part of them make sense. The irrelevant data samples have bad effects on the prediction outcome [5] .

In this paper, we introduce transfer learning technique to cross-project defect prediction, and propose a feature-based transfer learning approach to cross-project defect prediction (TrCPDP). Transfer learning refers to using the knowledge learned from one domain to help to learn the knowledge in another unknown domain[6] . The core insight of TrCPDP is to filter highly correlated data samples based on target project dataset, transfer common knowledge, and then evaluate and choose the best learning schema for target dataset. Models are then built for predicting defects in target projects.

Our main contributions are threefold:

(1) Improving prediction performance by adopting feature-based transfer learning technique on cross-project defect prediction.

Feature-based transfer helps to train a model to predict the target project defects. Experiments show that cross-project defect prediction model using our approach obtains prediction performance comparable to the within-project models and yields similar results even when the training data is not sufficient.

(2) Reducing data distribution divergence by filtering data samples and selecting common features to transfer.

To obtain highly correlated data samples for target project dataset, we adopt k-nearest algorithm and kNN algorithm to filter source data samples. K-nearest algorithm is used to find number of groups that data samples can be divided into, and kNN algorithm is used to find data samples that are correlated to a specific target data sample. To reduce divergence in data distribution between source dataset and target dataset, we design an algorithm to select common features to transfer.

(3) Evaluating different combinations of feature selection algorithm and classification algorithm.

Different target project defect prediction may call for different feature selection and classification algorithm. In this paper, we design an evaluation and selection process to find out the best fit prediction schema for corresponding target project.

The rest of this paper is structured as follows: In section 2 we describe related work about cross-project defect prediction. A feature-based transfer learning approach to cross-project defect prediction – TrCPDP is proposed in section 3. We then discuss experiment and cross-project predictability in section 4, and finally conclude the paper in section 5.

2. RELATED WORK

A variety of defect prediction technologies have emerged in recent years. Wang Qing et al.[7] categorize these technologies, such as metric-based defect prediction technology, defect distribution prediction technology, dynamic defect prediction technology, and so on. However, the performance of these prediction models is limited by the amount of data samples in the corresponding project. Except the COQUALMO model and the Bayesian method take the influence of different projects and environment factors into consideration, other methods and technologies are all limited by the history data samples. Obviously, these prediction models cannot be widely used.

In recent years, some researchers have explored the cross-project defect prediction in the following ways: reducing the divergence of data distribution, identifying groups of software projects with similar characteristic, and building adaptive prediction model. Some of them have achieved good performance.

2.1 Solve Divergence of Data Distribution

In order to reduce the divergence of data distribution, researchers take some measures such as reducing the data distribution space, selecting relevant data samples, etc.

Briand et al.[8] made a first attempt to solve the cross-project defect prediction. They used linear regression and MARS to build a prediction model. Results indicated that a model built on one system could be accurately used to rank classes within another system according to their fault proneness. However, because of project differences, the predicted fault probabilities were not representative.

Cruz et al.[9] found the distribution of design-complexity metrics varies from project to project, making the task of predicting across projects difficult to achieve. To solve the problem, they employed simple log transformations for making design-complexity measures more comparable among projects.

Nam et al.[10] found the performance of cross-project defect prediction is generally poor, largely because of feature distribution differences between source and target projects. They applied a state-of-art transfer learning approach, TCA[14], to make feature distributions in source and target projects similar.

Turhan et al.[2] proposed a practical defect prediction approach for companies that do not track defect related data. Specifically, they investigated the applicability of cross-company data for building localized defect predictors using static code features.

Jaechang Nam et al. [12] claimed that cross project defect prediction requires projects that have the same metric set, and proposed heterogeneous defect prediction to solve the limitation.

Xiaoyuan Jing et al. [13] proposed that for CPDP, the metrics used and the size of metric set are different in the data of two companies. They aimed to provide an effective solution for this problem by unifying metric representation.

Peters et al.[11] introduced the Peters filter, which was based on the conjecture that when local data is scarce, more information exists in other projects. Accordingly, the filter selected training data via the structure of other projects. To assess the performance of the Peters filter, they compared it with two other approaches for quality prediction. They found that within-project predictors are weak for small datasets, and the Peters filter+ cross-projects builds better predictors.

Tosun et al.[15] briefly explained their model, presented its payoff, and described how they have implemented the model in the company. Furthermore, they compared the performance of their model with that of another testing strategy applied in a pilot project that implemented a new process called team software process (TSP). Their results showed that defect predictors could predict 87 percent of code defects, decrease inspection efforts by 72 percent, and hence reduce post-release defects by 44 percent. They applied a data filtering process, used the dataset from NASA to predict other project defects successfully.

Peters and Zhang Hongyu et al.[16] aimed to enable effective defect prediction from shared data while preserving privacy. They proposed CLIFFed+MORPHed algorithms that maintain class boundaries in a dataset. Results showed that, for the OO defect data studied here, data could be privatized and shared without a significant degradation in utility.

2.2 Identify Groups of Similar Projects

Jureczko and Madeyski et al.[17] performed clustering on software projects in order to identify groups of software projects with similar characteristic. They believed one defect prediction model should work well for all projects that belong to such group.

Zhang et al.[18] proposed context-aware rank transformations for predictors. They clustered projects based on the similarity of the distribution of 26 predictors, and derived the rank transformations by using quantities of predictors for a cluster. Adding context factors to the universal model improves the predictive power. The universal model obtains prediction performance comparable to the within-project models and yields similar results when applied on five external projects.

2.3 Adaptive Prediction Model

Zhang Hongyu from Microsoft Research Asia and Zhou Zhihua from Nanjing University et al.[21] proposed a sample-based method for software defect prediction. They described three methods for selecting a sample: random sampling with conventional machine learners, random sampling with semi-supervised learner and active sampling with active semi-supervised learner. To facilitate the active sampling, they proposed a novel active semi-supervised learning method ACoForest that was able to sample the modules that were the most helpful for learning a good prediction model. Results showed that the proposed methods were effective.

Liu et al.[22] presented a novel search-based approach to software quality modeling with multiple software project repositories. They adopted a genetic-programming-based approach to select training dataset. They evaluated 17 different machine learning methods and ranked them, and then selected the best dataset.

2.4 Summary

We analyze the advantages and disadvantages of these technologies above in Table 1.

Table 1. Analysis of related work

Technology	Advantage	Disadvantage
Solve Difference of Various Project Data Samples	Based on data filtering, the ratio of relevant data samples can be increased, and the false alarm rate can be reduced.	The data after filtering still have some difference with target project. Potential relationships of features are ignored.
Identify Groups of Similar Projects	One defect prediction model is supposed to work well for all projects that belong to such group.	The technique of how to group projects is still not mature. And taking the project as granularity may ignore the possible existence of irrelevant data in project.
Adaptive Prediction Model	By adopting the genetic-programming-based approach, adaptive prediction model receive good performance.	The runtime of adaptive algorithm is long and its effectiveness is low. When there are large amount of other project dataset, the time complexity of prediction process will be incredible large.

Because of the disadvantage of current cross-project defect prediction approach, Y. Ma et al. [19] apply transfer learning [23] to defect prediction. In traditional classifiers, in order to ensure high precision and high reliability of the classification model, two prerequisites are hold:(1)the training data samples for learning and the target data samples for testing distribute independently and identically; (2)there are sufficient labeled data samples for training a model. However, in real software engineering practices, these two premises usually cannot be met. Transfer learning can effectively transfer knowledge and information between two similar domains. If adopting transfer learning, the two premises need not to be strictly met. The research of Nam validate that transfer learning can help to improve the accuracy of cross-project defect prediction. But no systematic processes and algorithms are proposed.

Our proposed TrCPDP approach overcomes weakness mentioned above and it consists of three parts: data distribution divergence reduction, feature selection, and prediction schema evaluation. Comparative experiments are designed to validate the effectiveness of our approach.

3. CROSS-PROJECT DEFECT PREDICTION

As shown in Figure 1, this approach takes a large amount of labeled other project data samples and a few labeled target project data samples as input. After data filter and transfer, and prediction schema evaluation, we can obtain highly powerful prediction model.

Our TrCPDP approach mainly accomplishes two tasks:

(1) Choose common components to transfer.

Simply using multi project dataset to train the model directly will lead to high false alarm rate. Data samples that are highly correlated to target project dataset need to be filtered out. Considering divergence in data distribution between source project dataset and target project dataset, some common features need to be selected to reduce the cost of transfer.

(2) Choose feature selection and classification algorithm.

Different features have different impacts on classification performance. There are many feature selection algorithms to select out the most representative features. But which one is best fit for cross-project defect prediction? This question remains to be answered.

Besides, using different machine learning classification algorithms may lead to different classification results. It is also necessary to evaluate which algorithm is the best for a specific target project defect prediction.

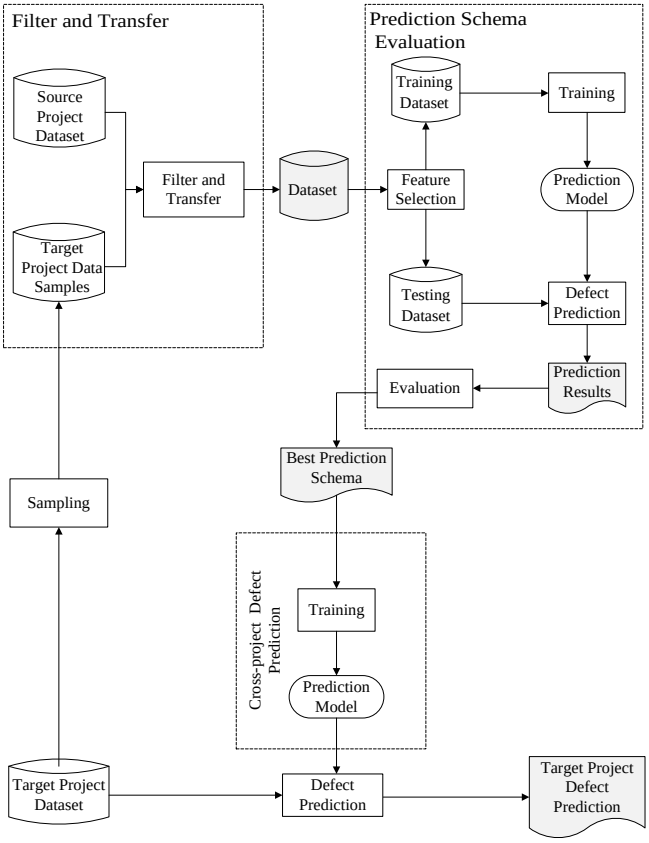

Figure 1. TrCPDP approach

Our TrCPDP approach consists of three phases:

(1) Filter and Transfer.

During this process, a subset of source data samples, which is highly correlated with target project dataset, is filtered out. Then common features between source dataset and target dataset are extracted out. Both source dataset and target dataset will be mapped into a new dimensional dataset. Based on the reduced dimensional dataset, we transfer knowledge from source projects to target project.

(2) Schema Evaluation.

New dimensional source dataset will be taken as training dataset, and new dimensional target dataset will be taken as testing dataset. Different feature selection algorithms and different classification algorithms will be evaluated. Prediction results will be recorded, and then a best performance prediction schema will be chosen.

(3) Cross-Project Defect Prediction.

Based on the feature selection algorithms and the prediction algorithms that are chosen from the previous steps, a cross-project defect prediction model can be built.

3.1 Data Filter and Transfer

The divergence in data distribution between source project dataset and target project dataset is a key cause of the bad performance of cross-project defect prediction. One solution is to preprocess source projects dataset, and then extract a subset that is highly correlated to the target project.

TrCPDP adopts two processes to achieve this objective: First, a data filtering algorithm is utilized to extract a subset of data samples. Second, a transfer learning algorithm is introduced to extract common features between source dataset and target dataset.

3.1.1 Data Filter

There are large amount of data samples in project. It is too expensive to train a model directly using all these data. Moreover, the false alarm rate will also be high. Peters et al.[28] discovered that selecting training dataset from initial dataset would help to decrease the false alarm rate, and eventually increase the recall (c.f. Section 3.2.2).

We adopt k-means clustering algorithm and kNN classification algorithm to filter out highly correlated project data samples. First we use k-means algorithm to partition different project data samples into different groups. Then for each group, we use k-nearest neighbor algorithm to pick out data samples that are highly correlated with the target project data samples.

3.1.2 Feature-Based Data Transfer

Features shown in Table 2 are common metrics selected from initial software features.

Table 2. Software features of datasets

Features	Description
amc (average method complexity)	The average size of java byte code
avg_cc (average McCabe)	the average complexity of McCabe's cyclomatic in a class
ca (afferent couplings)	the number of classes in other packages that depend upon classes within the package
cam (cohesion amongst classes)	classify the parameters of each method
cbm (coupling between methods)	the number of methods in one class that call the methods or attributes in other classes
cbo (coupling between objects)	the number of classes that call the methods or attributes in other classes
ce (efferent couplings)	the number of data types a class knows about
dam (data access)	the percentage of private or protected attributes

dit (depth of inheritance tree)	the place of the class in the inheritance tree
ic (inheritance coupling)	the number of parent classes which is coupled with it
lcom (lack of cohesion in methods)	measure the cohesion of each class of system
lcom3 (another lack of cohesion measure)	another method to measure the cohesion of each class of system
loc (lines of code)	the number of total lines of code
max_cc (maximum McCabe)	the max value of complexity of McCabe' cyclomatic in a class
mfa (functional abstraction)	the number of methods that can be inherited or can be called by member functions
moa (aggregation)	the number of class attributes that user defined
noc (number of children)	the number of direct child classes
npm (number of public methods)	the number of public functions in a class
rfc (response for a class)	the number of functions that are called to response to a message
wmc (weighted methods per class)	the weighted sum of all the methods in a class
defects	the number of defects that have been known

TrCPDP utilizes transfer learning algorithm to make cross-project prediction. Unlike traditional supervised learning, semi-supervised learning and unsupervised learning, transfer learning extracts knowledge from source domain, and then applies the knowledge to target domain. Based on different situations between source and target domains and tasks, transfer learning can be categorized under three sub-settings, namely inductive transfer learning setting, transductive transfer learning setting, and unsupervised transfer learning setting. In our cross-project defect prediction, transductive transfer learning is most suitable, because

- source and target tasks are the same; and
- unlabeled data in target domain are available; and
- a lot of labeled data in source domain are available.

Feature-representation transfer approaches to transductive transfer learning setting are used in this paper. After analyzing correlation between source dataset and target dataset[11], common features, on which the knowledge transfer is based, are recognized. If two datasets have some kind of correlation, there must be some common features between them. More specifically, there may exist some features that cause divergence in data distribution, while others may not. After discovering common features among source projects and target project, dimension of feature representation space is reduced.

As shown in figure 2, different shapes, such as circle, triangle, and diamond, denote different project samples. They have different data distribution. TrCPDP recognizes common features between source and target dataset, then maps both source and target dataset into the new feature space.

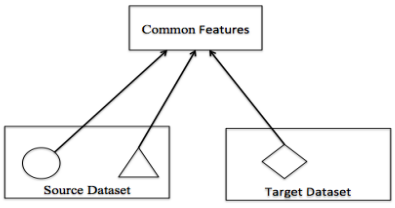

Figure 2. Feature-based transfer

In this paper, we use a method called *Transfer Component Analysis* (TCA) proposed in [14] to extract common features. The pseudo code is shown as follow:

Common Features Selection Algorithm

Input: Labeled source project dataset X_S.
Unlabeled target project dataset X_T.
Number of features in X_S, n_1.
Number of features in X_T, n_2.

Procedure:

1. Define a $(n_1 + n_2) \times (n_1 + n_2)$ kernel matrix $K = \begin{bmatrix} K_{S,S} & K_{S,T} \\ K_{T,S} & K_{T,T} \end{bmatrix}$, from witch $K_{S,S}$, $K_{T,T}$ and $K_{S,T}$ respectively are the kernel matrices on the data in the source project datasets X_S, target project dataset X_T, and cross-project datasets X_S+ X_T.
2. Define a $(n_1 + n_2) \times (n_1 + n_2)$ matrix $L = [L_{ij}] \succcurlyeq 0$ with $L_{ij} = \frac{1}{n_1^2}$ if $x_i, x_j \in X_S$; $L_{ij} = \frac{1}{n_2^2}$ if $x_i, x_j \in X_T$; otherwise, $-\frac{1}{n_1 n_2}$.
3. Use a $(n_1 + n_2) \times m$ matrix $\widetilde{W}$ to transform the corresponding feature vectors to a *m*-dimensional space. The resultant kernel matrix is $\widetilde{K} = (KK^{-\frac{1}{2}}\widetilde{W})(\widetilde{W}^{\intercal}K^{-\frac{1}{2}}K) = KWW^{\intercal}K$, where $W = K^{-1/2}\widetilde{W} \in \mathbb{R}^{(n_1+n_2)\times m}$.
4. Define centering matrix $H = I_{n_1+n_2} - \frac{1}{n_1+n_2}ll^{\intercal}$, where $l \in \mathbb{R}^{n_1+n_2}$ is the column vector with all ones, and $I_{n_1+n_2} \in \mathbb{R}^{(n_1+n_2)\times(n_1+n_2)}$ is the identity matrix.
5. Define a function $f = tr((W^{\intercal}(I + \mu KLK)W)^{-1}W^{\intercal}KHKW$, where identity matrix $I = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} \in \mathbb{R}^{m\times m}$, and μ is a trade-off parameter. Here μ is set to be 0.1.
6. Find a solution of W such that f has a maximum value.

Output: Matrix *W*.

In the sequel, $A \succ 0$ (resp. $A \succcurlyeq 0$) means that the matrix A is symmetric and positive definite (pd) (resp. positive semi-definite (psd)). Moreover, the transpose of vector or matrix is denoted by the superscript ⊺, and tr(A) denotes the trace of A.

The output of W is the eigenvectors corresponding to the m leading eigenvectors of $(I + \mu KLK)^{-1}KHK$. According to these eigenvectors, the leading components that minimize the difference between source dataset and target dataset are found. These leading components are extracted as common features.

Due to non-linear relationship among different features, TrCPDP adopts transformation form of the Principal Component Analysis – Kernel Principal Component[29] to select high-impact common features. Then based on common feature dataset, source dataset and target dataset will be transferred to a high correlation and small distance dataset. This obtained dataset will be used as input of prediction schema evaluation (the second step of TrCPDP).

3.2 Evaluation of Prediction Schema

During this process, we will compare the performance of different prediction schemas, including feature selection algorithms and prediction algorithms.

3.2.1 Evaluation Process of Prediction Schema

During the evaluation process, various combinations of feature selection and classification algorithms will be performed. Four common used feature selection algorithms, Chi-Square

algorithm(CS), InfoGain method(IG), Forward Selector method(FS), and Backward Elimination method(BE), are performed in our experiment. And three classification algorithms, Naïve Bayes, decision tree J48 and oneR (they are all realized in WEKA[27]), are used in our paper. Then 4×3 combinations, namely 12 prediction schema are obtained, shown in Table 4.

Table 3. Schema for evaluation

Schema	NB	j48	oneR
CS	CS+ NB	CS+j48	CS+oneR
IG	IG+ NB	IG+j48	IG+oneR
FS	FS+ NB	FS+j48	FS+oneR
BE	BE+ NB	BE+j48	BE+oneR

The input is the dataset obtained from previous process. To avoid occasional data anomalies, each combination of a feature selection algorithm and a classification algorithm will be tested m times. Then the average value of the m results will be taken as the final evaluation value, which is denoted as AvgResult. These AvgResult will be ranked in descending order, and then the first one will be selected as the best schema.

The pseudo-code of the evaluation process is shown as follow:

Algorithm Evaluation

input: *source*–dataset obtained from previous process

target – target dataset, the project to be predicted

schema–combination of a feature selection algorithm and a classification model

M - maximum number of repetitions

N - maximum number of folds

procedure:

M=3; /*number of repetitions

repeat

test=*target*;

train=*source*;

[learner,bestAttrs]= Learning(train,*schema*);

test' =*select*bestAttrs*from* test;

Result=TestClassifier(test', learner);

/* Compute the performance measures of the learner based on test dataset */

until M *times*

$$\text{AvgResult} = \frac{1}{\text{M}}\sum Result$$

output: AvgResult

3.2.2 Evaluation Measurement

In this paper the standard F-measure and the AUC (Area Under the ROC Curve) are taken as predictive performance evaluation measurement. It depends on precision and recall value. They are defined as follow:

$$\text{Precision} = TP/(TP+FP) \quad (1)$$

$$\text{Recall} = TP/(TP+FN) \quad (2)$$

TP (true positive) is the percent of defects that are predicted accurately, while FP (false positive) is the percent of defects that are predicted by mistake. FN is the percent of defects that are not predicted effectively.

The F-measure is then calculated as:

$$\text{F-measure} = (2*\text{precision}*\text{recall}) / (\text{precision} + \text{recall}) \quad (3)$$

ROC curve takes the recall value as y-axis, precision value as x-axis. The value of AUC is the area under the ROC curve, ranging from 0 to 1.

The larger F-measure and AUC are, the more accurate the prediction schema is.

3.3 Cross-project Defect Prediction

After the evaluation process, the best schema (namely the best combination of a feature selection algorithm and a classification algorithm) corresponding to each target project is figured out. Based on the dataset obtained from process (1), this schema can be utilized to make cross-project defect prediction.

3.4 Summary

In this section, we propose our TrCPDP approach for cross-project defect prediction. At the first step, a data filter and transfer process is adopted. Through data filter, highly correlated data samples can be picked out from source dataset. To reduce divergence in data distribution between source dataset and target dataset, common features are chosen and transferred. At the second step, we perform different schema on cross-project defect prediction, and evaluate their performance. The best schema corresponding to a specific target project is figured out. By virtue of the source dataset gained from the fist step and the best schema gained from the second step, cross-project defect prediction can be carried out.

4. EXPERIMENTS

In this paper the PROMISE datasets are used (http://code.google.com/p/promisedata/) to carry out the experiment. The experiments aim to show that the data selection and transfer process helps to improve the precision of the result, and TrCPDP is effective for the cross-project defect prediction, and evaluate the prediction power of TrCPDP.

4.1 Experimental Dataset

We choose datasets from 11 projects of their current version and their previous version from PROMISE dataset to conduct comparative experiments. The datasets are illustrated in Table 4.

Table 4. Project datasets

Project Dataset		Previous Version	
Project Name	**Number of Instances**	**Project Name**	**Number of Instances**
pbeans2	51	pbeans1	26
vel16	229	vel14	214
syn12	256	syn11	214
luc24	340	luc22	247
ivy20	352	ivy14	241

jedit42	367	jedit41	312
poi30	442	poi25	385
xer14	558	xer13	453
ant17	745	ant16	351
xal27	909	xal26	885
cam16	965	cam14	872

In order to meet the premise of cross-project defect prediction, only a few data samples in target project are labeled when validating the TrCPDP method. The target datasets are randomly partitioned into k subsets, and one of them is selected out. Since the experiment dataset is small, we set k = 5.

In this paper, we use WEKA[27] to make defect prediction. Processed dataset is taken as input, and WEKA will output prediction results. Comparing the results with actual defects of projects, we can calculate the value of F-measure and AUC.

4.2 Experiment on Data Filter and Transfer

A comparative experiment is conducted between **TrCPDP** and **NTrCPDP**. TrCPDP contains a data filter and transfer process, while NTrCPDP does not. The experiment aims to validate that the data filter and transfer process helps to improve the result of the cross-project defect prediction.

4.2.1 Experiment Design

In this paper, we implement NTrCPDP and TrCPDP on 11 various size projects and 12 prediction schemas. We make defect prediction on target project, and evaluate prediction performance of different prediction models. F-measure and AUC are recorded and analyzed. In each iteration, data samples of one project are taken as target project, and the other 10 project data samples are taken as training dataset.

4.2.2 Experimental Results

The experiment results of NTrCPDP and TrCPDP are shown in Table 5 and Table 6. We can observe that best prediction schema corresponding to different target project is different. Evaluation process of prediction schema is useful to select best classification algorithm and feature selection algorithm.

Table 5. Prediction evaluation of NTrCPDP

Target Dataset	F-measure	AUC	Best Prediction Schema
pbeans2	0.799	0.746	IG+NB
vel16	0.626	0.714	IG+NB
syn12	0.682	0.736	BE+j48
luc24	0.635	0.688	IG+NB
ivy20	0.793	0.732	BE+NB
jedit42	0.813	0.823	CS+NB
poi30	0.394	0.798	CS+NB
xer14	0.343	0.767	IG+NB
ant17	0.789	0.791	IG+NB
xal27	0.476	0.774	IG+j48
cam16	0.722	0.582	BE+NB

Table 6. Prediction evaluation of TrCPDP

Target Dataset	F-measure	AUC	Best Prediction Schema
pbeans2	0.697	0.531	BE+j48
vel16	0.757	0.732	FS+NB
syn12	0.715	0.707	IG+j48
luc24	0.744	0.736	FS+NB
ivy20	0.857	0.691	IG+j48
jedit42	0.84	0.823	CS+NB
poi30	0.825	0.813	IG+j48
xer14	0.928	0.908	IG+j48
ant17	0.785	0.753	IG+j48
xal27	0.832	0.87	IG+NB
cam16	0.776	0.606	IG+j48

Table 7. Comparison between NTrCPDP and TrCPDP

Target Dataset	F-measure			AUC		
	NTrCPDP	TrCPDP	Δf	NTrCPDP	TrCPDP	Δa
pbeans2	0.799	0.697	-0.102	0.746	0.531	-0.215
vel16	0.626	0.757	0.131	0.714	0.732	0.018
syn12	0.682	0.715	0.033	0.736	0.707	-0.029
luc24	0.635	0.744	0.109	0.688	0.736	0.048
ivy20	0.793	0.857	0.064	0.732	0.691	-0.041
jedit42	0.813	0.84	0.027	0.823	0.823	0
poi30	0.394	0.825	0.431	0.798	0.813	0.015
xer14	0.343	0.928	0.585	0.767	0.908	0.141
ant17	0.789	0.785	-0.004	0.791	0.753	-0.038
xal27	0.476	0.832	0.356	0.774	0.87	0.096
cam16	0.722	0.776	0.054	0.582	0.606	0.024
Average Value	0.643	0.796	**0.153**	0.741	0.743	**0.002**

4.2.3 Analysis of the Results

Prediction results of NTrCPDP and TrCPDP are shown in Table 7. Let △f denote difference of F-measures between the two methods, and △a denote difference of AUC between them. As we can see from table 7, data filter and transfer process improves precision of defect prediction effectively. F-measure of 9 projects among the total 11 increases 0.153 in average, and AUC of 6 projects increases

0.002 in average. Especially for xer14, its F-measure increases 0.585, and AUC increases 0.141.

4.3 Experiment on Cross Project Prediction

To evaluate prediction power of TrCPDP, we conduct a comparison experiment between cross-project defect prediction and within-project defect prediction. Performance of within-project defect prediction is generally considered to be better than that of cross-project defect prediction.

4.3.1 Experiment Design

Within-project defect prediction and cross-project prediction are carried out on 11 different projects and 12 prediction schemas.

(1) Within-project defect prediction. The four kinds of machine learning classification algorithms, NB, j48, and oneR, are used to make **IVDP** (within-version defect prediction) and **SVDP** (within-project prediction). Results of F-measure and AUC are recorded. SVDP takes previous version project dataset as training dataset, and then build prediction model to predict defects of subsequent version project. 10-fold cross validation is adopted in this experiment. In each iteration, we divide data samples into 10 subsets of equal size, and then select one subset as testing dataset and the other nine datasets as training dataset.

(2) Cross-project defect prediction. The experiments are carried out iteratively. We select one project dataset as target dataset, and the other projects as training dataset in each iteration. F-measure and AUC of best performance prediction model will be recorded.

4.3.2 Experimental Results

The experimental results are shown in Table 8 and Table 9.

Table 8. IVDP evaluation using different prediction algorithms

Evaluation of IVDP						
Target Dataset	**F-measure**			**AUC**		
	NB	J48	oneR	NB	J48	oneR
pbeans2	0.755	0.685	**0.799**	0.591	0.38	**0.626**
vel16	0.644	**0.702**	0.68	**0.733**	0.677	0.635
syn12	0.715	**0.784**	0.721	**0.733**	0.73	0.676
luc24	0.556	**0.645**	0.558	**0.723**	0.645	0.535
ivy20	0.844	0.837	**0.847**	**0.764**	0.536	0.533
jedit42	0.849	**0.87**	0.82	**0.83**	0.676	0.568
poi30	0.528	**0.787**	0.758	0.797	**0.812**	0.732
xer14	0.674	0.929	**0.932**	0.837	**0.904**	0.895
ant17	0.805	**0.808**	0.776	**0.808**	0.699	0.65
xal27	0.902	**0.994**	0.98	**0.851**	0.657	0.498
cam16	**0.767**	0.751	0.725	**0.676**	0.611	0.509

Table 9. SVDP evaluation using different prediction algorithms

Evaluation of SVDP						
Target Dataset	**F-measure**			**AUC**		
	NB	J48	oneR	NB	J48	oneR
pbeans1	**0.741**	0.714	0.503	**0.512**	0.345	0.601
vel14	0.631	**0.72**	0.625	**0.72**	0.693	0.57
syn11	0.73	**0.773**	0.659	**0.74**	0.745	0.604
luc22	0.566	0.673	**0.823**	**0.73**	0.677	0.481
ivy14	**0.85**	0.842	0.848	**0.749**	0.668	0.54
jedit41	**0.857**	0.823	0.811	**0.832**	0.568	0.553
poi25	0.506	**0.787**	0.73	**0.797**	0.797	0.699
xer13	0.68	**0.94**	0.932	0.842	**0.903**	0.895
ant16	0.809	**0.812**	0.762	**0.805**	0.689	0.629
xal26	0.901	**0.991**	0.98	**0.85**	0.564	0.498
cam14	**0.772**	0.751	0.724	**0.676**	0.616	0.509

Table 10. F-measure and AUC of TrCPDP compared to IVDP and SVDP

Target Dataset	f-measure			AUC		
	IVDP	SVDP	TrCPDP	IVDP	SVDP	TrCPDP
pbeans2	0.799	0.741	0.697	0.626	0.512	0.531
vel16	0.702	0.72	**0.757**	0.677	0.693	**0.732**
syn12	0.784	0.773	0.715	0.73	0.604	0.707
luc24	0.645	0.823	0.744	0.645	0.481	**0.736**
ivy20	0.847	0.85	**0.857**	0.533	0.668	**0.691**
jedit42	0.87	0.857	0.84	0.676	0.832	0.823
poi30	0.787	0.787	**0.825**	0.812	0.797	**0.813**
xer14	0.932	0.94	0.928	0.895	0.903	**0.908**
ant17	0.808	0.812	0.785	0.699	0.689	**0.753**
xal27	0.994	0.991	0.832	0.657	0.564	**0.87**
cam16	0.767	0.772	**0.776**	0.676	0.676	0.606
Mean Value	0.812	0.824	0.796	0.693	0.674	0.743

As we can see from Table 8 and Table 9, different prediction algorithm leads to different prediction results. So prediction algorithm should be chosen carefully to improve prediction accuracy. In other words, proper prediction schema can effectively improve the accuracy of defect prediction. This shows that the evaluation process of different prediction schemas is necessary.

4.3.3 Analysis of the Results

The F-measure results of IVDP, SVDP, and TrCPDP are shown in Figure 3, and the AUC results are shown in Figure 4. As is seen

from the figure, TrCPDP can get similar F-measure and AUC values to within-version and within-project defect prediction approach, sometimes even better.

(1) TrCPDP *vs* IVDP: Considering prediction accuracy, F-measure of TrCPDP is close to that of IVDP. Average difference between them is 0.016. What's more, the F-measures of projects vel16, luc24, ivy20, poi30, and cam16 are even higher than that of the within-project defect prediction. Considering the AUC, the results of 8 projects are higher than that of the traditional one, which increases 0.049 in average. Especially the result of xal27 increases 0.213, which is the highest.

(2) TrCPDP *vs* SVDP: Considering the prediction accuracy, F-measure of TrCPDP is close to that of SVDP. Average difference between them is 0.028. What's more, the F-measures of projects vel16, ivy20, poi30, and cam16 are even higher than that of the cross-version defect prediction. Considering the AUC, the results of 9 projects are higher than that of the traditional one, which increases 0.068 in average. Especially the result of xal27 increases 0.306, which is the highest.

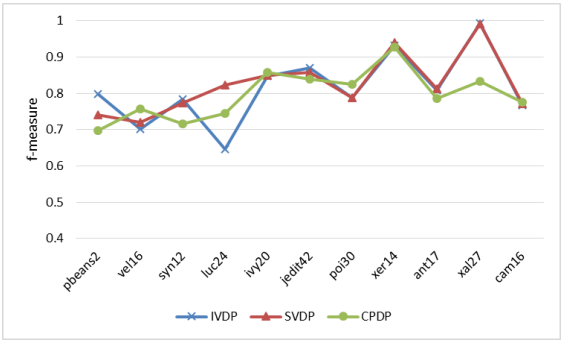

Figure 3. F-measure of TrCPDP compared to IVDP and SVDP

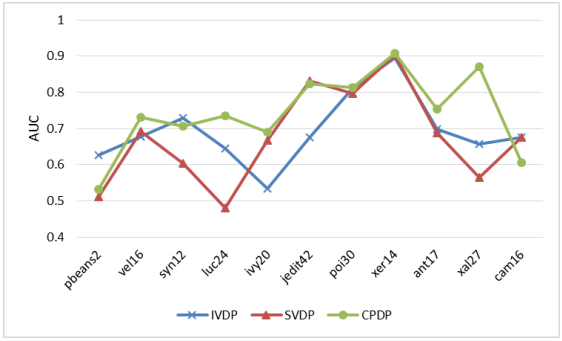

Figure 4. AUC of TrCPDP compared to IVDP and SVDP

5. CONCLUSION

In this paper we propose a cross-project defect prediction approach TrCPDP using feature-based transfer learning. By solving the difference among different project datasets, performance of cross-project defect prediction is improved. TrCPDP adopts a data filter and transfer process at first, and then evaluates various combinations of different feature selection algorithms and different classification algorithms. The false alarm rate is reduced largely, and prediction accuracy is improved effectively. Experiment result shows that, TrCPDP is better than traditional defect prediction approaches. Even when data samples of target project are not enough, it still has good performance.

Experiment results can be concluded as follow:

(1) Based on data filter and transfer, difference between source dataset and target dataset is reduced. It helps to improve the prediction accuracy. In our experiment, the precision increases 0.183 in average.

(2) By evaluating different combinations of feature selection algorithm and classification algorithm, the best prediction schemas corresponding to different target projects are different.

For future work we envision the following:

(1) Our experiment is conducted on PROMISE dataset. Projects in the dataset are limited. In future work, we plan to use more projects and extract more features to validate effectiveness of our approach.

(2) Feature selection algorithms and classification algorithms used in our TrCPDP approach are relatively simple. In the future, we will improve the algorithms and explore the effects of different combination of these new feature selection algorithm and classification algorithm.

(3) Our approach has been proved to be useful for projects of PROMISE dataset. But its effectiveness for projects in industry remains to be verified. In future, associated verification will be done with more sophisticated tools, which also need to be further developed for software defect prediction of large scale industry applications.

6. ACKNOWLEDGMENTS

This research is supported by 973 Program in China (Grant No. 2015CB352203) and National Natural Science Foundation of China (Grant No. 61472242).

7. REFERENCES

[1] Akiyama F. An example of software system debugging. In: Proc. of the Int'l Federation of Information Proc. Societies Congress. New York: Springer Science and Business Media, 1971. 353−359.

[2] Turhan B, Menzies T, Bener A, et al. On the relative value of cross-company and within-company data for defect prediction. Empirical Software Engineering, 2009, 14(5): 540-578.

[3] Zimmermann T, Nagappan N, Gall H, et al. 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, NY, USA, 2009, 91-100.

[4] Nagappan N, Ball T, Zeller, A. Mining metrics to predict component failures. In: Proc. of the 28th international conference on Software engineering, NY, USA, 2006, 452-461.

[5] Turhan B, Menzies T, Bener A B, et al. On the relative value of cross-company and within-company data for defect prediction. Empirical Software Engineering, 2009, 14(5): 540-578.

[6] Zhuang FZ, He Q, Shi ZZ. Survey on transfer learning research. Journal of Software,2015, 26(1): 26-39. (in Chinese with English abstract).

[7] Wang Q, Wu SJ, Li MS. Software Defect Prediction. Journal of Software, 2008, 19(7): 1565-1580. (in Chinese with English abstract).

[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] Cruz A, Ochimizu K. Towards logistic regression models for predicting fault-prone code across software projects. In: Proc. of Empirical Software Engineering and Measurement, Lake Buena Vista, FL, 2009, 460-463.

[10] Nam J, Pan S J, Kim S. Transfer defect learning. In: Proc. of International Conference on Software Engineering, San Francisco, CA, 2013, 382-391.

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

[12] Y. Ma, G. Luo, X. Zeng, and A. Chen. Transfer Learning for Cross-company Software Defect Prediction. Information and Software Technology, 2012, 54(3): 248-256.

[13] Xiaoyuan Jing et al. Heterogeneous Cross-Company Defect Prediction by Unified Metric Representation and CCA-Based Transfer Learning. ESEC/FSE 2015, 496-507.

[14] Pan S J, Tsang I W, Kwok J T, et al. Domain Adaptation via Transfer Component Analysis. IEEE Transactions on Neural Networks, 2010, 22(2): 199-210.

[15] Tosun A, Bener A B, Kale R. AI-Based Software Defect Predictors: Applications and Benefits in a Case Study. IAAI, 2011, 32(2): 57-68.

[16] Fayola Peters, Tim Menzies, Liang Gong, Hongyu Zhang. Balancing Privacy and Utility in Cross-Company Defect Prediction. IEEE Trans. Software Eng. 39(8): 1054-106.

[17] Marian Jureczko, Lech Madeyski. Towards identifying software project clusters with regard to defect prediction. In: Proc. of the 6th International Conference on Predictive Models in Software Engineering , 2010.

[18] 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.

[19] Y. Ma, G. Luo, X. Zeng, and A. Chen. Transfer Learning for Cross-Company Software Defect Prediction. Inf. Softw. Technol, 2012, 54(3):248-256.

[20] Nam J, Pan S J, Kim S. Transfer Defect Learning. ICSE'13, 382-391.

[21] Ming Li, Hongyu Zhang, Rongxin Wu, Zhi-Hua Zhou. Sample-based software defect prediction with active and semi-supervised learning. Automated Software Engineering, 2012, 19(2): 201-230.

[22] Liu Y, Khoshgoftaar T M, Seliya N. Evolutionary optimization of software quality modeling with multiple repositories. IEEE Transactions on Software Engineering, 2010, 36(6): 852-864.

[23] Wenyuan Dai, Qiang Yang, Gui-RongXue, Yong Yu. Boosting for transfer learning. In: Proc. of ICML 2007: 193-200

[24] Wenyuan Dai, Yuqiang Chen, Gui-RongXue, Qiang Yang, Yong Yu. Translated Learning: Transfer Learning across Different Feature Spaces. NIPS 2008: 353-360

[25] Wenyuan Dai, Ou Jin, Gui-RongXue, Qiang Yang, Yong Yu. EigenTransfer: a unified framework for transfer learning. ICML 2009: 193-200

[26] Dai WY. Instance-based and Feature-based Transfer Learning [MS. Thesis]. Shanghai Jiao Tong University,2008 (in Chinese with English abstract).

[27] http://www.cs.waikato.ac.nz/ml/weka/

[28] Peters F, Menzies T, Gong L, et al. Balancing privacy and utility in cross-company defect prediction. IEEE Transactions on Software Engineering, 2013, 33(9): 637-640.

[29] Scholkopf B, Smola A, Muller K R. Kernel Principal Component Analysis. Lecture Notes in Computer Science, 1997, 1327: 583-588.