Proceedings of the International Conference on
Pattern Recognition, Informatics and Medical Engineering, March 21-23, 2012

Classifying the Bugs Using Multi-Class Semi Supervised Support Vector Machine

Ayan Nigam
Quality & Process Lead
Ideavate Solutions
Indore, India
ayan.nigam@gmail.com

Bhawna Nigam
Dept. of Information Tech.
IET- DAVV
Indore, India
bhawnanigam@gmail.com

Chayan Bhaisare
Dept. of Computer Engg
IET- DAVV
Indore, India
chynbsre@gmail.com

Neeraj Arya
Dept. of Computer Engg
IET- DAVV
Indore, India
neerajaryagate2010@gmail.com

Abstract- **It is always important in the Software Industry to know about what types of bugs are getting reported into the applications developed or maintained by them. Categorizing bugs based on their characteristics helps Software Development team to take appropriate actions in order to reduce similar defects that might get reported in future releases. Defects or Bugs can be classified into many classes, for which a training set is required, known as the Class Label Data Set. If Classification is performed manually then it will consume more time and efforts. Also, human resource having expert testing skills & domain knowledge will be required for labelling the data. Therefore Semi Supervised Techniques are been used to reduce the work of labelling dataset, which takes some labeled with unlabeled dataset to train the classifier. In this paper Self Training Algorithm is used for Semi Supervised Learning and Winner-Takes-All strategy is applied to perform Multi Class Classification. This model provides Classification accuracy up to 93%.**

Keywords: Semi Supervised Learning, Multi-Class Classification, Support Vector Machine (SVM), Bugs, Self Training.

I. INTRODUCTION

A defect or a flaw in the software is called as Bug. In Software Development, Bug indicates the unexpected behaviour [1]. Mostly, bug arises from mistakes made by the development team in gathering requirements, designing or in writing source code and a few are caused by compilers. A buggy program [2] is the one that contains large number of bugs, which affects the functionality of the software.

During software testing, the unexpected behaviour identified by Software Testers or Quality Engineers is marked as a Bug. Common types of bugs are – Functional Bug, Logical Bug, Performance Bug, UI Bug, Requirement Bug and Design Bug. Reports detailing bugs in a program are commonly known as Bug Reports, Fault Reports, Problem Reports, Trouble Reports, Change Requests and so forth.

The problem with traditional Supervised Machine Learning Methods [3] is that these methods require large amount of labeled data for training the classifier. Since labeled data is difficult to obtain and expensive to process, therefore a new machine learning method named Semi Supervised Learning [4][5] is used. Semi Supervised Learning uses small amount of labeled data together with large amount of unlabeled data to train the classifier. Thus the Semi Supervised SVM provides good generalization performance using small amount of labeled data unlike Supervised SVM which uses huge amount of labeled data to train. The Self Training Algorithm [6] is used for Semi Supervised Learning, which is an iterative algorithm.

This paper proposes a Bug Classification [7] Model using Multi-Class Semi Supervised SVM [5,8] using different Kernel functions. The Semi Supervised SVM is trained with the combination of the labeled and unlabeled data. It is then used to detect the bug by classifying the data based on hyper plane. To perform Multi-Class Classification [9,10] of data from the traditional SVM, Winner-Takes-All strategy is used.

II. ARCHITECTURE OF BUG CLASSIFICATION SYSTEM

The architecture of the Semi Supervised SVM Bug Classification Model is shown in Fig. 1. First step involves Pre-processing of dataset, in which the stop words are removed from the data and the data is converted into numeric format so as to make it suitable for performing calculations. Pre-processing of the dataset is important to improve the accuracy of the SVM. The processed data is spitted into two parts- Training data and testing data. Training data is used to train the model and testing data is used to determine the accuracy of model.

A. Pre-Processing

The Preprocessing [11] of dataset consists of removal of stop words and conversion of the data into numerical format**.**

1. Removal of Stop Words: This step involves removal of punctuation marks and stop words.

In order to improve the accuracy of the model stop

978-1-4673-1039-0/12/$31.00 ©2012 IEEE

words must be removed from the data [12]. Here, stop-words dictionary is used to filter useless words. For e.g. sample data before Preprocessing is-

Application captures the text and error message of captcha verification code on registration form from contact us form, when a user inputs wrong captcha on contact us form, FunctionalBug

After Preprocessing, the above lines will be-

Application captures text error message captcha verification code registration contact form user inputs wrong captcha contact form FunctionalBug

Training algorithm is applied which is simple and effective Semi Supervised Learning Algorithm.

In Self Training algorithm [6], the classifier is first trained with the small amount of labeled data and then it is used to classify the unlabeled data. This newly obtained labeled data is added to the previous labeled data. Now the classifier is trained with the increased labeled data. This procedure is repeated until the classifier is sufficiently trained. This algorithm is also called as Self-Teaching. The algorithm works as follows:

Suppose that there is a training dataset d_t containing n samples $\{x_i, i = 1 \ldots n\}$ with given label $\{y_i, i = 1 \ldots n\}$ and a test dataset d_s containing m

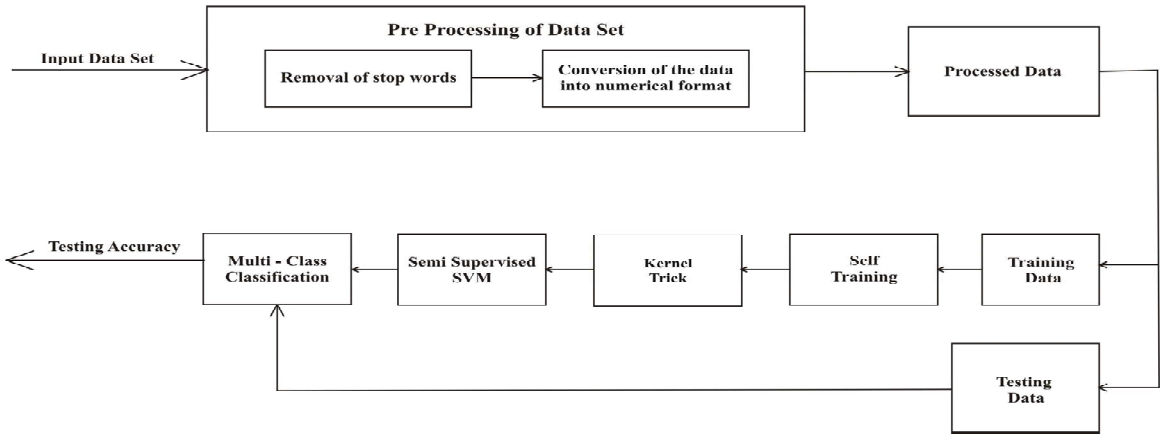

Fig. 1 Architecture of Bug Classification System

2. Conversion of the data into numerical format:- After the Preprocessing of bug's description, whole dataset is converted into numerical format to perform necessary computation for the training of SVM. For converting the data in numeric format, the word is assigned with a number which indicates the number of times that word occurs in its domain. Numerical equivalent of the above description will be-

18 1 3 7 4 2 1 1 4 2 3 10 1 2 2 2 3 1

The numerical dataset obtained is further divided into Training, Testing and Unlabeled Dataset. Every third example is added to the Testing Set, every fourth example is added to the Unlabeled Set and the rest of the examples are added to the Training Set.

B. Semi Supervised Learning

There are many algorithms available for Semi Supervised Learning [14,15]. The commonly used algorithms [4] are Self Training, Co-Training, Tri-Training, Generative Mixture Model, Transductive SVM and Graph Based Method. In this work Self

samples $\{x_i, i = 1 \ldots m\}$ but its labels are known. The following steps are followed-

1. Train the classifier with the training dataset d_t and then classify the test samples.
2. Perform step 2.1 – 2.2 k times (k=2...)
   2.1 Create a new training set $D_{new} = d_t + d_s$. d_s is the new labelled data obtained in the first step.
   2.2 Train the classifier with the dataset D_{new} and again classify the unlabeled data in d_s
3. After kth (i.e. k=11) iteration the classifier is sufficiently trained for performing the Classification. Classifier shows good accuracy when k is set to 11.

C. Kernel Trick

The procedure for calculating the values of parameters, which are needed in the learning task of the Semi Supervised SVM, is lengthy and complex. It requires in transformed space $\Phi(x_i).\Phi(x_j)$, the

calculations of dot product between pairs of vectors. These calculations are very time consuming and complex and may suffer from the dimensionality problem. To reduce these calculations, Kernel functions [13] are used. These are also called as Kernel Trick.

The Kernel Function is a kind of mathematical trick through which one can classify the data in three dimensional spaces, when the data was originally defined in two dimensional spaces. The Kernel functions originally maps the data from the lower dimensional space in which the data is not linearly separable to the higher dimensional space where the data is linearly separable. A Kernel function K can be represented as

$$K(x_i, x_j) = \Phi(x_i).\Phi(x_j)$$

Where u and v are n-dimensional vectors.
The following Kernel functions are used:

- Sigmoid Kernel Function –
$$k(x, y) = \tanh(\alpha x^T y - c)$$
- Multiquadric Kernel Function –
$$k(x, y) = \sqrt{\|x - y\|^2 + c^2}$$
- Radial Basis Function (RBF) –
$$k(x, y) = \exp\left(-\gamma \|x - y\|^2\right)$$
- Inverse Multiquadric Kernel Function –
$$k(x, y) = \frac{1}{\sqrt{\|x - y\|^2 - c^2}}$$
- Power Kernel Function–
$$k(x, y) = -\|x - y\|^d$$

D. Semi Supervised Support Vector Machine

The Semi Supervised SVM, which uses small quantity of labeled data, provides better accuracy than the Supervised SVM. The Semi Supervised SVM uses the concept of Maximal Margin Hyperplane as shown in Fig 2. There are many hyperplanes available but the hyperplane having maximal margin is preferred because it contains less generalization errors. The Semi Supervised SVM finds the Maximal Margin Hyperplane and thus the Semi Supervised SVM is often called the Maximal Margin Classifier. The learning task of the Semi Supervised SVM is given as:

$$y_i(w.\Phi(x_i) + b) \geq 1 \text{ where } i = 1,2,3 \ldots n$$

Where y_i represents the output labels of the training data sets and x_i represents the training samples.

The dual Lagrangian for the constrained optimization problem is given as:

$$\alpha_i = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

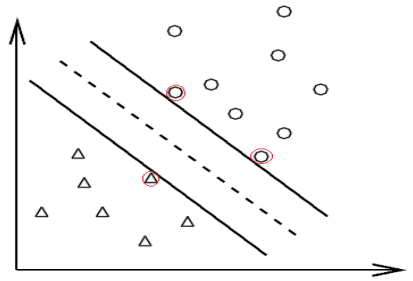

Fig 2. Maximal Margin Hyperplane

Where $K(x_i, x_j)$ denotes the Kernel Functions whose value is equivalent to

$$K(x_i, x_j) = \Phi(x_i).\Phi(x_j)$$

Once the value of α_i's are found using the quadratic programming techniques, the parameters w and b are calculated with the help of the following equations:

$$W = \sum_i \alpha_i y_i \Phi(x_i)$$

$$\alpha_i \left\{ y_i \left(\sum_j \alpha_i \, y_j \Phi(x_i).\Phi(x_j) + b \right) - 1 \right\} = 0$$

Finally a test sample or instance 's' is classified using the following decision function equation:

$$f(s) = sign(w.\Phi(s) + b) = sign\left(\sum_{i=1}^{n} \alpha_i \, y_i \, K(x_i, x_j) + b\right)$$

The value of the decision function $f(s)$ will decide the label or class of the test samples. If the value of the decision functions $f(s) = 1$, then the test sample will be classified as a positive class. If the value of the decision function is 0 i.e. $f(s) = 0$ then the test samples will be classified as the negative class or given negative label. Thus, the sign of the decision function $f(s)$ is checked for the labelling of the unlabeled samples or test samples.

E. Multi- Class Classification

By combining several two-class SVMs Multiclass [8] SVMs are can be implemented. The One-Versus-

All Method using Winner-Takes-All strategy and the One-Versus-One method implemented by Max-Wins Voting are popularly used for this purpose. In this work Winner-Takes-All strategy is used for the performing Multi-Class Classification.

For a given Multi-Class problem, let K denotes the number of classes and ω_i, i = 1 . . . K represents the K classes. For Binary Classification two classes will be referred as positive and negative; a Binary Classifier will be assumed to produce an output function that gives relatively large values for examples from the positive class and relatively small values for examples belonging to the negative class. The Winner-Takes-All SVM constructs M Binary Classifiers. The ith classifier output function ρi is trained taking the examples from ωi as positive and the examples from all other classes as negative. For a new example x, Winner-Takes-All SVM strategy assigns it to the class with the largest value of ρi.

III. EXPERIMENTS

A .DataSets

Two datasets, as shown in Table 1, are considered for the experiment which contains the description of bugs. Machine generated bug description is used for training and testing of Classifier. The datasets contains the description of the bugs together with its type. Five kernel functions are used for the Classification namely Inverse Multiquadric, Radial Basis, Sigmoid, Multiquadric and Power. The experiment is performed on two separate datasets containing different bugs and on different Kernel parameters to compare the accuracy of classifier on the applied datasets. The dataset used for testing consists of records that are not the part of training dataset. Thus, the machine is not biased towards some particular type of records.

The information of the datasets used is:

	Bug1.txt	**Bug2.txt**
Total Data	400	800
Labeled Data	208	400
Unlabeled Data	64	136
Test Data	128	264

TABLE 1. Dataset Used for Training and Testing

B. Experimental Result

The Semi Supervised SVM is first trained with the collection of labeled and unlabeled data. Five kernels are used for the purpose of training and testing. Different kernel functions are applied over these datasets with different parameters such as gamma, degree and coefficients to analyze the performance of the classifier over the different dataset .Also, the accuracy of classifier have been analyzed for different parameters of the Kernel Function.

For calculating testing accuracy, the testing data is applied on the Classifier and then the accuracy of the formula:

Testing accuracy= (correct classification of instance) / (total number of instance in the testing dataset).

Kernel Function	Values of Parameter	Bug1.txt Accuracy	Bug2.txt Accuracy
Inverse Multiquadric Kernel	Coeff=1	75%	72%
	Coeff=2	87%	75%
	Coeff=3	87%	78%
Radial Basis	Gamma=1	87%	69%
	Gamma=2	56%	78%
	Gamma=5	62%	63%
Sigmoid	Coeff=2, Gamma=1	56%	57%
	Coeff=3, Gamma=1	75%	66%
	Coeff=5, Gamma=1	67%	63%
Multiquadric	Coeff=1	86%	75%
	Coeff=4	81%	76%
	Coeff=5	93%	75%
Power	Degree=1	81%	72%
	Degree=3	65%	61%
	Degree=4	62%	63%

TABLE 2. Classification Accuracy of the model over two datasets

It is observed from Table 2 that as the number of unlabeled data increases, the accuracy of the Classifier reduces. Among the different kernel functions used in the experiment, Inverse Multiquadric, Radial Basis and Multiquadric Kernel functions provide good accuracy for various Kernel function parameters.

Fig. 3 shows the type of bugs in the test data of Bug1.txt. From the total number of test data 400, 208

are labeled data, 64 are unlabeled data and 128 are test data tested using Multiquadric kernel function with coefficient=4 and accuracy obtained is 81%.

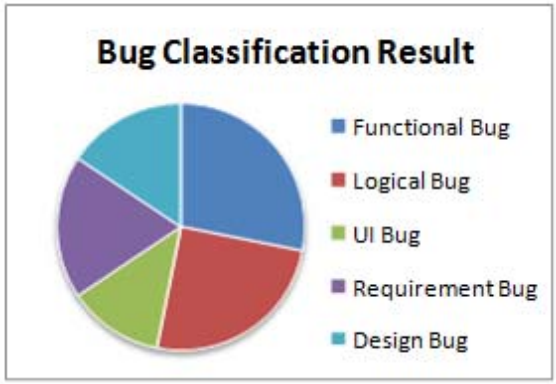

Fig.3. Graph of testing data of Bug1

IV. CONCLUSION

This paper proposes Classification of Bugs using Multi-Class Semi Supervised SVM. Most of the Kernel Functions gave high accuracy based on the parameter of function and the number of labeled and unlabeled data. We have successfully classified the data to one of the multiple classes with high accuracy. The method can be applied to classify Defects on criteria such as Severity, Priority, and Bug Injection Stage etc.

REFERENCES

[1] Michael Grottke, and Kishor S. Trivedi, "A Classification of Software Faults.", Sixteenth International IEEE Symposium on Software Reliability Engineering,pages 4.19-4.20, 2005.

[2] Olga Baysal, Michael W. Godfrey and Robin Cohen, "A Bug You Like: A Framework for Automated Assignment of Bugs", n Proceedings of the 17th IEEE Intl. Conference on Program Comprehension (ICPC-09), 17-19 May 2009.

[3] ABM Shaukat Ali, Sleh A.Wasimi, "Data Mining: Methods and Techniques", South Melbourne, Thomson Learning Australia, 2007.

[4] Xiaojin Zhu, "Semi-Supervised Learning Literature Survey", Computer Sciences TR 1530 University of Wisconsin – Madison, 2008.

[5] G.Fung,O. L. Mangasarian, "Semi-Supervised Support Vector Machines for Unlabeled Data Classification", Data Mining Institute Technical Report 99-05, October 1999. Optimization Methods and Software 15, 2001, 29-44.

[6] Yuanqing Li *, Cuntai Guan, Huiqi Li, Zhengyang Chin, "A self-training semi-supervised SVM algorithm and its application in an EEG-based brain computer interface speller system", Journal Pattern Recognition Letters archive Volume 29 Issue 9, July, 2008.

[7] S. Kim , E. James Whitehead, Jr. , Yi Zhang, Classifying Software Changes: Clean or Buggy?, IEEE Transactions on Software Engineering, March 2008.

[8] Kai-Bo Duan1 and S. Sathiya Keerthi – "Which Is the Best Multiclass SVM Method? An Empirical Study", In Proceedings of the Sixth International Workshop on Multiple Classifier Systems, 2005.

[9] Jun-Ki Min, Jin-Hyuk Hong, and Sung-Bae Cho, "Ensemble Approaches of Support Vector Machines for Multiclass Classification", PReMI'07 Proceedings of the 2nd international conference on Pattern recognition and machine intelligence, 2007.

[10] Arnulf B. A. Graf & Christian Wallraven - "Multi-class SVMs for Image Classification using FeatureTracking", Technical Report No. 099, 2002.

[11] Shivaji, S., J. E. James Whitehead, R. Akella, and S. Kim, "Reducing Features to Improve Bug Prediction.", 24th IEEE/ACM International Conference on Automated Software Engineering , Auckland, New Zealand, Nov. 16-20, 2009.

[12] Qinghua Zou, Wesley W. Chu, PhD, Craig Morioka, Gregory H. Leazer and Hooshang Kangarloo, "IndexFinder: A Method of Extracting Key Concepts from Clinical Texts for Indexing", Annual Symposium proceedings/ AMIA Symposium. AMIA Symposium (2003)

[13] Nello Cristiani, John Shawe-Taylor – "An introduction to support vector machine and other kernel-based learning", Cambridge University Press New York, NY, USA ©2000

[14] GholamReza Haffari and Anoop Sarkar, "Analysis of Semi-supervised Learning with the Yarowsky", 23rd Conference on Uncertainty in Artificial Intelligence, UAI 2007. Vancouver, BC. July 19-22, 2007.

[15] Lian Yu, Changzhu Kong, Lei Xu, Jingtao Zhao, and Hull-iui Zhang, "Mining Bug Classifier and Debug Strategy Association Rules for Web-Based Applications", Proceedings of The Fourth International Conference on Advanced Data Mining And Applications, 2008.

[16] Syed Nadeem Ahsan, Javed Ferzund and Franz Wotawa, "Automatic Software Bug Triage System (BTS) Based on Latent Semantic Indexing and Support Vector Machine". Proc. Fourth International Conference on Software Engineering Advances (ICSEA 2009), Porto, Portugal, 2009.