International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-2 Number-4 Issue-6 December-2012

Data and Cost handling Techniques for Software Quality Prediction Through Clustering

Saifi Bawahir[1], Mohsin Sheikh[2]
M.E. (IT) M.I.T.M. Indore[1], Assistant Professor M.I.T.M. Indore[2]

Abstract

Analysis of Data quality is an important issue which has been addressed as data warehousing, data mining and information systems. It has been agreed that poor data quality will impact the quality of results of analyses and that it will therefore impact on decisions made on the basis of these results. An attempt to improve classification accuracy by pre-clustering did not succeed. However, error rates within clusters from training sets were strongly correlated with error rates within the same clusters on the test sets. This phenomenon could perhaps be used to develop confidence levels for predictions. The main and the common problem that the software industry has to face is the maintenance cost of industrial software systems. One of the main reasons for the high cost of maintenance is the inherent difficulty of understanding software systems that are large, complex, inconsistent and integrated. The main reason behind the above phenomena is because of different size and level of arrangements. Decomposing a software system into smaller, more manageable subsystems can aid the process of understanding it significantly. Different algorithms construct different decompositions. Therefore, it is important to have methods that evaluate the quality of such automatic decompositions. In our paper we present a brief survey on software quality prediction through clustering.

Keywords

Software quality, Clustering, Decomposition, Cost handling.

1. Introduction

To finding the fault or fault prediction is the biggest challenge in today's scenario. There is several research orientations in this direction. Despite the amount of effort spent in the design and application of fault prediction models, software fault prediction research area still poses great challenges. Unfortunately, none of the techniques which are created in few years ago satisfies the applicability in the software industry due to several reasons including the lack of software tools to automate this prediction process, the unwillingness to collect the fault data, and the other practical problems. The traditional way which is used from the beginning is to estimate software quality by using software metrics and fault data collected from previous system releases or similar projects to construct a quality-prediction or quality-classification model. Then engineers use this model to predict the fault proneness of software components in development. Previous research [1] has shown that software quality models based on software metrics can yield predictions with useful accuracy. Such models can be used to predict the response variable that can either be the class of a component or a quality factor for a component. The former is usually referred to as classification models [2] while the latter is usually referred to as prediction models [3]. The focus of this paper is on the former, i.e., classification models. Quite often, predicting the number of faults is not necessary.

The data is the crucial part for the software engineering and in the same sense it is used for predicts and discovers new strategies. They are also used to indicate that new strategies are working, or what impact new techniques have. It is interesting to see then, that data quality in empirical software engineering appears to be somewhat neglected in publications and even in data analyses. It is all the more astonishing since, as De Vaux and Hand [4] Stated, 60-95% of the effort of data analysis is making use for the cleaning. The area of research like information systems and data mining the impact of poor data has been recognized as an issue which needs to be addressed by database designers and data users alike. Redman [5] for instance stated that poor data quality is an issue which impacts on all segments of the economy: companies, governments, and academia and their customers", and Wand and Wang [6] warned of the severe impact of poor data quality on the effectiveness of an organization. The remaining of this paper is organized as follows. We discuss Clustering in Section 2. In Section 3 we discuss about problem domain. Literature Survey in section 4.In section 5 we discuss about software clustering. The conclusions and future directions are given in Section 6. Finally references are given.

2. Clustering

According to Shi Zhong [7] clustering is an Unsupervised learning methods such as clustering techniques are a natural choice for analyzing software quality in the absence of fault proneness labels. Clustering algorithms can group the

software modules according to the values of their software metrics. The underlying software-engineering assumption is that fault-prone software modules will have similar software measurements and so will likely form clusters. Clustering is a mechanism where it allows us to run applications on several parallel servers. The distribution load is distributed across different servers, and even if any of the servers fails, the application is still accessible via other cluster nodes. It is crucial for scalable enterprise applications, as you can improve performance by simply adding more nodes to the cluster by which it is recognized by several well established connections protocol. Basically cluster is a collection of network club together to form an interface. It is partition and the partitions contain several nodes which are interconnected which are shown in figure 1.

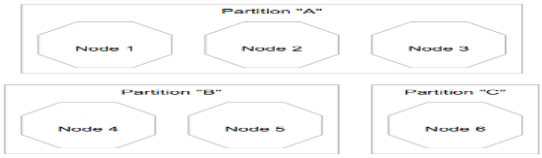

Figure 1: Cluster and Server Nodes

The stub object is generated by the server and it implements the business interface of the service. The client then makes local method calls against the stub object. The call is automatically routed across the network and invoked against service objects managed in the server. In a clustering environment, the server-generated stub object is also called intermediate by which the basic call is generated through the object stub for the process call which is shown in figure 2.

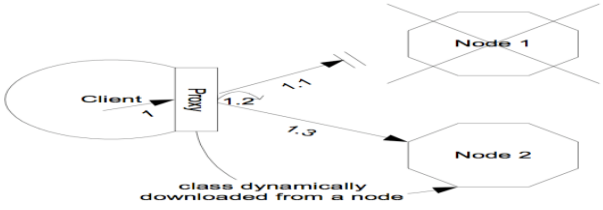

Figure 2: Class dynamically downloaded from the node

The clients use any web browser and send information to the server and send a request by using certain communication protocols. In this case, a load balancer is required to process all requests and dispatch them to server nodes in the cluster.

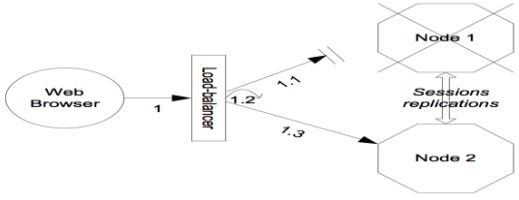

Figure 3: Load Balancing Architecture

3. Problem Statement

According to Mark Shtern [8] there are several issues which is facing by the researchers. Most of the researchers perform their work on a small set of data or a software system. The applicability is always a question mark when we apply the same process for the big database. Therefore, it is not possible to generalize the evaluation results to other software systems. The source code is no longer supported and it is not feasible for all other resources. Compatibility is the greater issue. Fault detection is also an issue. Data Quality defines as "fitness for purpose" [9][10]. Since this purpose is subjective and important to consider, data quality's characteristics or dimensions are subjective and cannot be assessed independent of the people who use the data .This means that the domain the data are used in has to be an important consideration. Redman [11] lists more than 10 dimensions. Redman [11] made the trenchant observation that his list cannot be comprehensive since as indicated above data quality dimension depend on the user's view of the data, pointing towards a reason for this lack of consensus about data quality dimensions. Redman categorized his dimensions into four groups:

(i) Dimensions related to the data model,
(ii) Dimensions related to the data values,
(iii) Dimensions related to data presentation and
(iv) Dimensions related to information technology.

Manago and Kodrato [12] stated that noise is present when a knowledge base does not truly reflect the environment we want to learn from". They are indicating that the causes of noise lead analysts to build inaccurate models. According to their definition, noise is wrong information, lack of information or unreliable information. The term unreliable information is interesting, since the information is not incorrect, but unreliable. So we also concentrate on the issue of noise.

4. Literature Survey

In 2004, Shi Zhong et al. [13] describe an exploratory analysis method that addresses two challenges and that is built with clustering and the help of a software engineering expert. It is an unsupervised method since labeled training data are not required to predict the fault-proneness of software modules. They present two real-world case studies to verify the effectiveness of the clustering- and expert-based approach in predicting both the fault-proneness of software modules and potential noisy for e.g., mislabeled modules.

In 2009, Mark Shtern[8] discuss about several software clustering algorithms Most of these algorithms have been applied to particular software systems with considerable success. However, the question of how to select a software clustering algorithm that is best suited for a specific software system remains unanswered. They introduce a method for the selection of a software clustering algorithm for specific needs. The proposed method is based on a newly introduced formal description template for software clustering algorithms. Using the same template, we also introduce a method for software clustering algorithm improvement.

In 2007, Mark Shtern[14] introduce UpMoJo, a novel comparison method for software decompositions that can be applied to both nested and flat decompositions. The benefits of utilizing this method are presented in both analytical and experimental fashion. We also compare UpMoJo to the END framework, the only other existing method for nested decomposition comparison.

In 2010, Ramandeep S. Sidhu [15] uses subtractive clustering based fuzzy inference system approach which is used for early detection of faults in the function oriented software systems. This approach has been tested with real time defect datasets of NASA software projects named as PC1 and CM1. Both the code based model and joined model of the datasets are used for training and testing of the proposed approach. The performance of the models is recorded in terms of Accuracy, MAE and RMSE values. The performance of the proposed approach is better in case of Joined Model. As evidenced from the results obtained it can be concluded that Clustering and fuzzy logic together provide a simple yet powerful means to model the earlier detection of faults in the function oriented software systems.

In 2010, Mark Shtern [16] introduces and quantifies the notion of clustering algorithm comparability. It is based on the concept that algorithms with different objectives should not be directly compared. Not surprisingly, we find that several of the published algorithms in the literature are not comparable to each other.

In 2012, Deepak Gupta et al. [17] discusses about Clustering which is the unsupervised classification of patterns into groups. A clustering algorithm partitions a data set into several groups such that similarity within a group is larger than among groups The clustering problem has been addressed in many contexts and by researchers in many disciplines; this reflects its broad appeal and usefulness as one of the steps in exploratory data analysis.. There is need to develop some methods to build the software fault prediction model based on unsupervised learning which can help to predict the fault –proneness of a program modules when fault labels for modules are not present. One of the methods is use of clustering techniques. They present a case study of different clustering techniques and analyze their performance.

5. Software Clustering

Cluster analysis is a group of multivariate techniques whose primary purpose is to group entities based on their attributes. They are classified according to the criteria which is predefined. The objective of any clustering algorithm is to sort entities into groups, so that the variation between clusters is maximized relative to variation within clusters.
The stages of cluster analysis techniques are

1. Fact Extraction
2. Filtering
3. Similarity Computation
4. Cluster Creation
5. Results Visualization
6. User Feedback Collection

Before applying clustering to a software system, the set of entities to cluster needs to be identified. Entity selection depends on the objective of the method. An attribute is a set of values. An attribute is usually a software artefact, such as a package, a file, a function, a line of code, a database query, a piece of documentation, or a test case. Attributes may also be high level concepts that encompass software artefacts, such as a design pattern.

Selecting an appropriate set of attributes for a given clustering task is crucial for its success Source Code Source code is the most popular input for fact extraction.

There are two conceptual approaches to extracting facts from source code: syntactic and semantic. The syntactic which is structure-based approaches focus on the static relationships among entities. The exported facts include variable and class references, procedure calls, use of packages, association and inheritance relationships among classes etc.

Some approaches work with the information available in binary modules. Depending on compilation and linkage parameters, the binary code may contain information, such as a symbol table that allows efficient fact extraction. This approach has three advantages:

1. It is language independent
2. Binary modules are the most accurate and reliable information, source code may have been lost or mismatched to a product version of binary modules. Source mismatch situations occur because of human mistakes, patches, intermediate/unreleased versions that are working in the production environment etc.
3. Module dependency information is easy to extract from binary modules Linkage information contains module dependency relations.

The main drawbacks of this approach are that binary meta-data information depends on building parameters, and that the implementation of the approach is compiler/hardware dependent. Also, binary code analysis cannot always discover all relationships.

Static information is often insufficient for recovering lost knowledge since it only provides limited insight into the runtime nature of the analyzed software; to understand Behavioural system properties, dynamic information is more relevant.
During the run-time of a software system, dynamic information is collected. The collected information may include:

1. Object construction and destruction
2. Exceptions/errors
3. Method entry and exit
4. Component interface invocation
5. Dynamic type information
6. Dynamic component names

Performance Counters and Statistics

(a) Number of threads
(b) Size of buffers
(c) Number of Network Connections
(d) CPU and Memory Usage
(e) Number of Component Instances
(f) Average, Maximum and Minimum Response Time

There are various ways of collecting dynamic information, such as instrumentation methods or third party tools (debuggers, performance monitors etc). Instrumentation techniques are based on introducing new pieces of code in many places to detect and log all collected events. Such techniques are language dependent, and not trivial to apply. After the extraction process is finished, a altering step may take place to ensure that irrelevant facts are removed, and the gathered facts are prepared for the clustering algorithm. According to shi Zhong [4]there are three different approaches to effective noise handling in data analysis exist designing robust algorithms that are insensitive to noise, filtering out noise, and correcting noise. Most robust algorithms have a complexity control mechanism so that the resulting models don't over fit training data and generalize well to future unseen data. Cross-validation, minimum description length, and structural risk minimization are some commonly used model selection principles.

6. Conclusion

Traditional software engineering has neglected the issue of data quality to some extent. This fact poses the question of how researchers in empirical software engineering can trust their results without addressing the quality of the analyzed data. In this paper we present a discussion as well as some insight on the problem faces in software quality prediction which will be achieved by clustering.

References

[1] Khoshgoftaar, T. M., Allen, E. B., Jones, W. D., and Hudepohl, J. P. Accuracy of software quality models over multiple releases, in Annals of Software Engineering 9(1–4): 103–116.
[2] Khoshgoftaar, T. M., Allen, E. B., and Deng, J. Controlling overfitting in software quality models: experiments withregression trees and classification, in Proceedings: 7th International Software Metrics Symposium. London UK, 190–198. 2001.
[3] Gokhale, S. S., and Lyu, M. R. Regression tree modeling for the prediction of software quality, in H. Pham (ed.): Proceedings: 3rd International Conference on Reliability and Quality in Design. Anaheim, California, USA, 31–36.1997.
[4] Richard D. De Veaux and David J. Hand. How to lie with bad data. Statistical Science, 20(3):231{238, 2005.
[5] Lesley Pickard, Barbara Kitchenham, and Stephen G. Linkman. Using simulated data sets to compare data analysis techniques used for software cost modelling. IEE Proceedings - Software, 148(6):165{174, 2001.
[6] Andres Folleco, Taghi Khoshgoftaar, Jason Van Hulse, and Lofton A. Bullard. Software quality modeling: The impact of class noise on the random forest classifier. In Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2008, June 1-6, 2008, Hong Kong, China, pages 3853{3859. IEEE, 2008.
[7] Shi Zhong, Taghi M. Khoshgoftaar, and Naeem Seliya, "Analyzing Software Measurement Data with Clustering Techniques" , IEEE 2004.
[8] Mark Shtern and Vassilios Tzerpos," Methods for Selecting and Improving Software Clustering Algorithms", IEEE 2009.
[9] Michael Gertz, M. Tamer Ozsu, Gunter

Saake, and Kai-Uwe Sattler. Report on the Dagstuhl seminar: Data quality on the web". SIGMOD Record, 33(1):127{132, 2004.

[10] Rossane Prince and Graeme G. Shanks. A semiotic information quality framework. In Proceedings of IFIP International Conference on Decision Support Systems (DSS2004): Decision Support in an Uncertain and Complex World, 2004.

[11] Thomas C. Redman. Data Quality for the Information Age. Artech House, Inc., Norwood, MA, USA, 1996. ISBN 0890068836. Foreword By-A. Blanton Godfrey.

[12] Michel Manago and Yves Kodrato Noise and knowledge acquisition. In Proceedings of the 10th International Joint Conference on Artificial Intelligence, pages 348{354, 1987.

[13] Shi Zhong, Taghi M. Khoshgoftaar, and Naeem Seliya, "Expert-Based Software Measurement Data Analysis with Clustering Techniques", Accepted to IEEE Intelligent Systems, Special Issue on Data and Information Cleaning and Preprocessing. 2004.

[14] Mark Shtern and Vassilios Tzerpos , "Lossless Comparison of Nested Software Decompositions", Working Conference on Reverse Engineering, Vancouver, BC, October 2007, pp. 249-258.

[15] Ramandeep S. Sidhu, Sunil Khullar, Parvinder S. Sandhu, R. P. S. Bedi, Kiranbir Kaur, "A Subtractive Clustering Based Approach for Early Prediction of Fault Proneness in Software Modules", World Academy of Science, Engineering and Technology,2010.

[16] Mark Shtern and Vassilios Tzerpos "On the Comparability of Software Clustering Algorithms" Proceedings of the 18th IEEE International Conference on Program Comprehension, Braga, Minho, June-July 2010, pp. 64-67.