International Journal of Engineering, Science and Mathematics
Vol. 9 Issue 06, June 2020
ISSN: 2320-0294 Impact Factor: 6.765
Journal Homepage: http://www.ijesm.co.in, Email: editorijmie@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gage as well as in Cabell's Directories of Publishing Opportunities, U.S.A

NOVEL APPROACH USING MACHINE LEARNING FOR IMPLEMENTATION OF SOFTWARE REUSEABILITY

Priya Tiwari[1], [2]Dr. Lalji Prasad

[1]RGPV University, Sagar Institute of Research and Technology, Indore

[2]Sagar Institute of Research and Technology, RGPV University, Indore

ABSTRACT: With advancement in software field, development is increasing day by day and to reduce the cost and time involved in development by developers the demand of Reusability is increasing. we are doing work on source code reusability. Source code reusability utilized to grow quality and profitability of programming.Itso improves in general nature of programming in least vitality and time. So, we have studied about some attributes and also studied about various metrics to measure and quantify these attributes and justify them. Major goal is to describe the role of Reusability and its scenario and justify the result.

KEYWORDS:-Software Reuse,PCA, Machine Learning.

1. INTRODUCTION

Reuse of software components is becoming more and more important in a variety of aspects of software engineering. Recognition of the fact that many software systems contain many similar or even identical components that are developed from scratch over and over again has led to efforts to reuse existing components. Structuring a system into largely independent components has several advantages. It is easy to distribute the components among various engineers to allow parallel development. Maintenance is easier when clean interfaces have been designed for the components, because changes can be made locally without having unknown effects on the whole system. And, if components' interrelationsare clearly documented and kept to a minimum, it becomes easier to exchange components and incorporate new ones into a system.

Software reuse and software components have a major influence on the structure of software systems as well as on the way we build them. Yet, manyquestions are still unanswered. What are software components? What are their properties that support reuse and adaptability? What are the requirements for building evolving systems? What are the implications for the software life cycle? What are the legal, economic and organizational consequences? In this book we will provide answers to these important questions. In the rest of this chapter we briefly give an introduction to what we mean by software component provides an overview of the structure.

COMPONENT-BASEDSOFTWARE ENGINEERING

Component-based Software Engineering (CBSE) become known in late 1990s. Motivation behind the development of this approach is, software developer are not satisfied by reusability approach through object oriented analysis. It is the process of defining, selection, implementation, retrieving, integrating loosely coupled software components into applications.
In CBSE, software applications are build from pre-developed software components, it enhances the degree of software reusability, and the software applications are developed with reusable software components through integration mechanism like interface, pipe, glue code. Figure 1 depict the principal activities of CBSE.

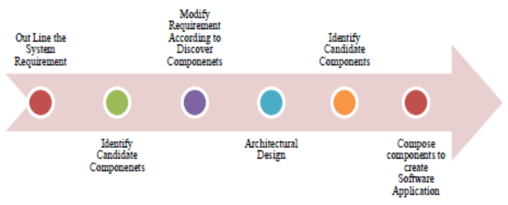

Out Line the System Requirement
Modify Requirement According to Discover Componenets
Identify Candidate Components
Identify Candidate Componenets
Architectural Design
Compose components to create Software Application

Figure 1 : Component-based Software Engineering Process

2. LITERATURE REVIEW

Shiva, S. & Shala, Lubna. (2007). Since the concept of systematic software reuse was proposed in 1968, several approaches have been suggested to achieve the promised potential of software reuse. Three of the major approaches are componentbased software reuse, software architecture and design reuse, and domain engineering and software product lines.

Tawfig et al (2014) The idea of reuse is nothing but the skill to merge the software concepts to form a larger unit of software. It improves the quality and productivity of software production process. This paper in brief gives the review of the current research status in the field of software reuse and the research contributions. Here several upcoming trends for research in software reuse are examine.

Devi, T. R., & Rama, B. (2017) Reuse of programme product results from the use of the already available modules in the development of new applications. In order to increase efficiency and the consistency of software programmes, reused components should be used in the most important and efficient way. In order to reuse repository elements, we must concentrate on choosing the required retrieval process. The option of choosing the right or best suited component and the excellent possible collection of the component we have received is more complex than easy component retrieval.

He, X., Xue, C., & Zhou, Q. (2015). In comparison, there is a lack of system engineering method in process science reusing information. This paper offers a device architecture method for reusable applications, integrating a multiple iterative model , in order to systematically construct reusing technique and methodology, and efficiently create reusable software items.

3. PROPOSED APPROACH

The non-linear approach is used mainly for data analysis and high-dimensional data visualisation. T-SNE gives you a feeling or intuition about how the data is organised in a high-dimensional space, in simpler words. In 2008, Laurens van der Maatens and Geoffrey Hinton created it.

t-SNE vs PCA

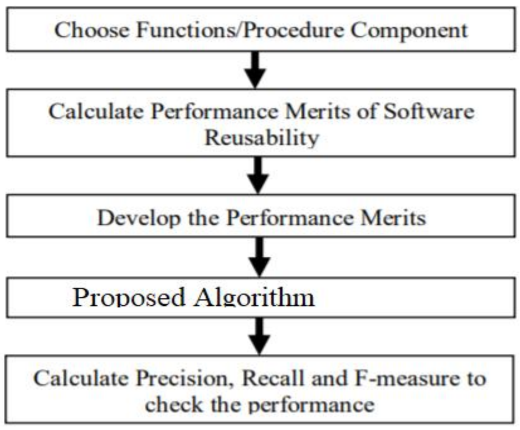
If you know Principal Components Analysis (PCA), so you're probably curious about the difference between PCA and t-SNE, just like me. The first thing to remember is that, while t-SNE was developed in 2008, PCA was developed in 1933. Since 1933, something has improved in the world of computer science, especially in the area of computing and data scale. Second, PCA is a technique of linear dimension reduction that aims to optimise variance and preserves large distances in pairs. In other words, things that are distinct end up far away. This can lead to poor visualisation, especially when working with multiple constructs that are not linear. Think of any geometric form as a multiple structure: cube, ball, circle, etc.

Bayes Algorithm

Naïve Bayes is a subset of Bayesian decision theory. It's called naive because the formulation makes some naïve assumptions. Python's text-processing abilities which split up a document into a vector are used. This can be used to classify text. Classifies may put into human-readable form. It is a popular classification method in addition to conditional independence, overfitting, and Bayesian methods. In the face of the simplicity of Naive Bayes, it can classify documents surprisingly well. Instinctively a potential justification for the conditional independence assumption is that if the document is about politics, this is a good evidence of the kinds of other words found in the document. Naive Bayes is a reasonable classifier in this sense and has minimal storage and fast training, it is applied to time-storage critical applications, such as automatically classifying web pages into types and spam filtering

Logistic Regression

NB is a basic Bayesian supervised classifier that assumes that, given the sense of the class, all attributes are independent of each other:


Choose Functions/Procedure Component
Calculate Performance Merits of Software Reusability
Develop the Performance Merits
Proposed Algorithm
Calculate Precision, Recall and F-measure to check the performance

In certain real-world cases, this is called the NB assumption, which is rarely valid. Classification is sometimes performed by NB. The parameters for each property can be modified independently due to this assumption, and this significantly simplifies learning, particularly when the number of properties is enormous. In view of the NB presumption, as a function of the likelihood of consistency values for all word characteristics, the likelihood of a study dependent on its class can be calculated.

- To show how to use linear model stochastic gradient descent on multi-class classification/discrimination
- import class sklearn.linear_model.SGDClassifier
- Gradient descent is an iterative optimization algorithm to find the minimum of a function. Here that function is our Loss Function.
- Imagine a valley and a person with no sense of direction who wants to get to the bottom of the valley. He goes down the slope and takes large steps when the slope is steep and small steps when the slope is less steep. He decides his next position based on his current position and stops when he gets to the bottom of the valley which was his goal. Let's try applying gradient descent to **m** and **c** and approach it step by step:
- Initially let m = 0 and c = 0. Let L be our learning rate. This controls how much the value of **m** changes with each step. L could be a small value like 0.0001 for good accuracy.
- Calculate the partial derivative of the loss function with respect to m, and plug in the current values of x, y, m and c in it to obtain the derivative value **D**.

$$D_m = \frac{1}{n}\sum_{i=0}^{n} 2(y_i - (mx_i + c))(-x_i)$$

$$D_m = \frac{-2}{n}\sum_{i=0}^{n} x_i(y_i - \bar{y}_i)$$

- D□ is the value of the partial derivative with respect to **m**. Similarly lets find the partial derivative with respect to **c**, Dc

$$D_c = \frac{-2}{n}\sum_{i=0}^{n}(y_i - \bar{y}_i)$$

- Now we update the current value of **m** and **c** using the following equation:

$$m = m - L \times D_m$$

$$c = c - L \times D_c$$

- We repeat this process until our loss function is a very small value or ideally 0 (which means 0 error or 100% accuracy). The value of **m** and **c** that we are left with now will be the optimum values.

- Now going back to our analogy, **m** can be considered the current position of the person. **D** is equivalent to the steepness of the slope and **L** can be the speed with which he moves. Now the new value of **m** that we calculate using the above equation will be his next position, and **L×D** will be the size of the steps he will take. When the slope is more steep (**D** is more) he takes longer steps and when it is less steep (**D** is less), he takes smaller steps. Finally he arrives at the bottom of the valley which corresponds to our loss = 0. Now with the optimum value of **m** and **c** our model is ready to make predictions !
- Proposed algorithm used algorithms in machine learning, mainly because it can be applied to any function to optimize it

4. RESULTS ANALYSIS

Compute precision, recall, F-measure and support for each class

the precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

The F-beta score can be interpreted as a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0.

The F-beta score weights recall more than precision by a factor of beta. beta == 1.0 means recall and precision are equally important.

The support is the number of occurrences of each class in y_true.

- Naive Bayes using TF-IDF
- but split it first into train-set (80% of our data-set), and validation-set (20% of our data-set)

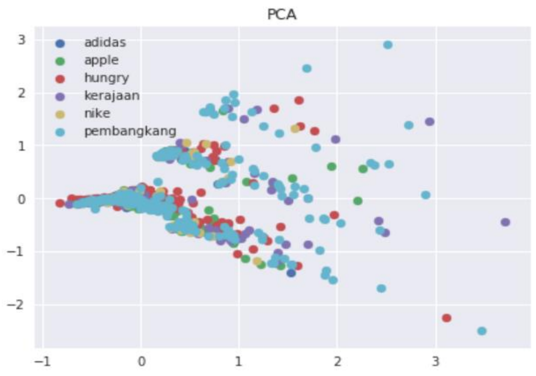


Figure 4. 1: Data Visualization Principal Components analysis

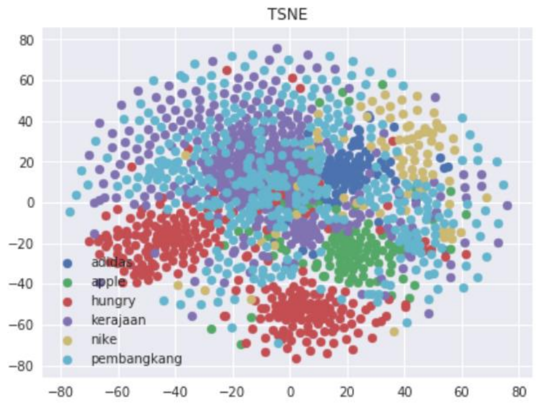


Figure 4.2 : Data Visualization using t-distributed stochastic neighbor embedding

International Journal of Engineering, Science and Mathematics
Vol. 9 Issue 06, June 2020
ISSN: 2320-0294 Impact Factor: 6.765
Journal Homepage: http://www.ijesm.co.in, Email: editorijmie@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gage as well as in Cabell's Directories of Publishing Opportunities, U.S.A

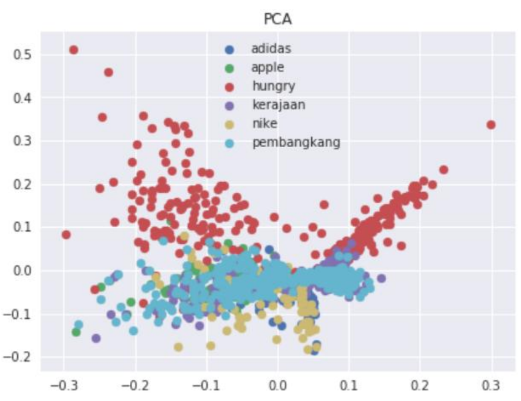


Figure:4.3 Data Visualization using Principal Components analysis

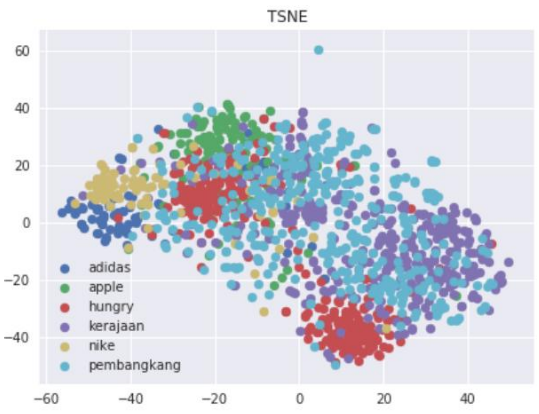


Figure 4.4 : Data Visualization using t-distributed stochastic neighbor embedding

```
accuracy validation set:  0.87131844238
             precision    recall  f1-score   support

     adidas       0.96      0.85      0.90       279
      apple       0.99      0.82      0.90       434
     hungry       0.99      0.89      0.94      1060
   kerajaan       0.87      0.85      0.86      1436
       nike       0.93      0.82      0.87       303
pembangkang       0.77      0.91      0.83      1547

avg / total       0.88      0.87      0.87      5059
```

The support is the number of occurrences of each class in y_true.

```
accuracy validation set:  0.867958094485
             precision    recall  f1-score   support

     adidas       0.93      0.83      0.88       306
      apple       0.99      0.78      0.87       451
     hungry       0.99      0.91      0.95      1043
   kerajaan       0.86      0.85      0.86      1406
       nike       0.97      0.77      0.86       321
pembangkang       0.76      0.91      0.83      1532

avg / total       0.88      0.87      0.87      5059
```

```
accuracy validation set:  0.849970349872
             precision    recall  f1-score   support

     adidas       0.95      0.85      0.90       313
      apple       0.98      0.77      0.87       478
     hungry       0.99      0.90      0.95      1046
   kerajaan       0.84      0.82      0.83      1377
       nike       0.98      0.77      0.87       310
pembangkang       0.73      0.88      0.80      1535

avg / total       0.87      0.85      0.85      5059
```

Feed Naive Bayes using hashing but split it first into train-set (80% of our data-set), and validation-set (20% of our data-set)

```
accuracy validation set:  0.793239770706
             precision    recall  f1-score   support

     adidas       0.95      0.58      0.72       286
      apple       1.00      0.48      0.65       461
     hungry       0.92      0.91      0.91      1077
   kerajaan       0.86      0.80      0.83      1366
       nike       0.98      0.54      0.70       345
pembangkang       0.64      0.89      0.75      1524

avg / total       0.83      0.79      0.79      5059
```

This segment addresses established research model and its implementation. The related prediction of reusability using numerous techniques has been investigated. Here, machine parameters are used as data, and the output is their respective significance for determining reusability of the individual classes and outputs for prediction. Inter-modal evaluation of effectiveness is carried out in order to find the highest performing model to be recommended for employability in real time. The following section briefs on policy or analytical application, accompanied by collected findings and their respective interpretations. In this article, given the relevance of reusability estimation for optimum programme architecture, numerous algorithms have been evaluated for their respective effectiveness against earlier reusability prediction, including numerous artificial intelligence techniques.

5. CONCLUSION

Here we shows that a survey of Reusability. We have studied about approximately twelve attributes and also studied about various metrics to measure and quantify these attributes and justify them.

As a result of this evaluation it is confirmed that the proposed study justify the fact of Reusability.

The reliability terminology increases the efficiency of the Software Reuse process. It also enhances the whole process of compatible issues with software reuse component because the component of a software should be supportive and compatible the other component of other software. Today, the scenario of software reuse is highly progressive. Software reuse terminology can be more extended in a progressive manner in future. Software reuse directly impacts on the factor of cost and time. The third factor reliability can be introduced in the software reuse terminology. In the article, we have tried to get the possible issues of reliability with the help of software reuse terminology. Finally, it is observed that the software reuse process can be beneficial for achieving the reliable and quality software products

6. FUTURE WORK

By applying these cohesion and coupling metrics on source code we will find how much code is reusable. So our main focus is to find how the source code will reusable. Higher stability, and lower defect density of reused parts clearly illustrate the industrial value of reuse. Both of these observations reflect explicit information based on their own data and are also critical for determining future reuse approaches. Results in potential information reuse experiments can also be considered as a benchmark for comparison.

REFERENCES

[1]. Shiva, S. & Shala, Lubna. (2007). Software Reuse: Research and Practice. 603-609. 10.1109/ITNG.2007.182.

[2]. Frakes, William & Kang, Kyo. (2005). Software Reuse Research: Status and Future. Software Engineering, IEEE Transactions on. 31. 529 - 536. 10.1109/TSE.2005.85.

[3]. Tawfig.M.Abdelaz.Z, Yasmeen.N.Zada and Mohamed.A.Hagal(2014) A structural approach to improve software design reusability

[4]. Devi, T. R., & Rama, B. (2017). *Designing software reuse repository through intelligent classification for effective search and retrieval mechanism. 2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB).* doi:10.1109/aeeicb.2017.7972327

[5]. He, X., Xue, C., & Zhou, Q. (2015). *A system engineering approach for reusable software. 2015 First International Conference on Reliability Systems Engineering(ICRSE).* doi:10.1109/icrse.2015.7366426

[6]. Devesh Manjhi, Amrita Chaturvedi, (2017) Software Component Reusability Classification in Functional Paradigm, 978-1-5386-8158-9/19/$31.00©2019IEEE

[7]. Ahmed Mateen, Samina Kausar (2017) ,"A Software Reuse Approach and Its Effect On Software Quality, An Empirical Study for The Software Industry", International Journal of Management, IT & Engineering Vol. 7 Issue 2, February 2017, ISSN: 2249-0558 Impact Factor: 7.119 Journal Homepage: http://www.ijmra.us

[8]. M. Melik-Merkumians, M. Wenger, R. Hametner and A. Zoitl, "Increasing portability and reuseability of distributed control programs by I/O access abstraction," 2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010), Bilbao, 2010, pp. 1-4, doi: 10.1109/ETFA.2010.5641252.

[9]. H. Lounis, T. F. Gayed and M. Boukadoum, "Using Efficient Machine-Learning Models to Assess Two Important Quality Factors: Maintainability and Reusability," 2011 Joint Conference of the 21st International Workshop on Software Measurement and the 6th International Conference on Software Process and Product Measurement, Nara, 2011, pp. 170-177, doi: 10.1109/IWSM-MENSURA.2011.44.

[10]. P. V. Parande and M. K. Banga, "Evolutionary Computing Assisted Heterogenous Ensemble Model for Web-of-Service Software Reusability Prediction," 2019 4th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT), Mysuru, India, 2019, pp.

158-163, doi: 10.1109/ICEECCOT46775.2019.9114712.

[11]. M. Hashemi, "Reusability of the Output of Map-Matching Algorithms Across Space and Time Through Machine Learning," in IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 11, pp. 3017-3026, Nov. 2017, doi: 10.1109/TITS.2017.2669085.

[12]. M. Banga, A. Bansal and A. Singh, "Implementation of Machine Learning Techniques in Software Reliability: A framework," 2019 International Conference on Automation, Computational and Technology Management (ICACTM), London, United Kingdom, 2019, pp. 241-245, doi: 10.1109/ICACTM.2019.8776830.

[13]. P. Zehnder and D. Riemer, "Modeling self-service machine-learning agents for distributed stream processing," 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, 2017, pp. 2203-2212, doi: 10.1109/BigData.2017.8258170.

[14]. A. P. Singh and P. Tomar, "The analysis of software metrics for design complexity and its impact on reusability," 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, 2016, pp. 3808-3812.

[15]. J. L. Ribeiro, M. Figueredo, A. Araujo, N. Cacho and F. Lopes, "A Microservice Based Architecture Topology for Machine Learning Deployment," 2019 IEEE International Smart Cities Conference (ISC2), Casablanca, Morocco, 2019, pp. 426-431, doi: 10.1109/ISC246665.2019.9071708.