Identification of Object Oriented Reusable Components Using Multilayer Perceptron Based Approach

Shamsher Singh, Pushpinder Singh, and Neeraj Mohan

Abstract—Software reuse, is the use of existing software assets partial, modified or complete, to create new software. An approach puts this idea central is called reuse-based software engineering. For Software professionals, reusability is powerful means to potentially overcome the situation known as software crisis and they promises significant software productivity, reliability and quality. In this way a software component is developed only once, and can save out development effort multiple times. In this research paper our objective is the identification of object oriented reusable components using Multilayer Perceptron based Approach.

Keywords—Reusability, Artificial Neural networks, Multilayer Perceptron.

I. INTRODUCTION

SOFTWARE reuse[10,24], the use of existing software artifacts or knowledge either partial, modified or complete, to create new software, is a key method for significantly improving software quality, reliability and productivity or in other words, it is the process of implementing or updating software systems using existing software assets. Software assets or components include anything that is produced from a software development effort. In this way a software component is developed only once, and can save out development effort multiple times. Many software developing organization uses CBSE as their software development standard because it reduces the development cost. To achieve this, CBSE is relying on reusability of software assets. So in directly it is clear that reusability is a key factor for reducing the development cost and an approach puts this idea central is called reuse-based software engineering.

Software professionals have recognized reuse as a powerful mean to potentially overcome the situation called, software crisis. According to Gomes, this idea is appeared in 1968, which opens new horizons for the software design and development. Reusable software components have been promoted in recent years. The software development community is gradually drifting toward the promise of widespread software reuse, in which any new software system can be derived virtually from the existing systems. As a result, an increasing number of organizations are using software not just as all-inclusive applications, but also as component parts of larger applications. In this new role, acquired software must integrate with other software functionality.

II. INTELLIGENT SYSTEMS

Large No Intelligent systems possess one or more of these characteristics
1. Capability to extract and store knowledge
2. Human like reasoning process
3. Learning from experience (or training)
4. Dealing with imprecise expressions of facts
5. Finding solutions through processes similar to natural evolution

Most intelligent systems in use today are based either on the rule based methodology from the field of Artificial Intelligence(AI) known popularly as expert systems, or on one or more of the methodologies belonging to the more recently emerging field known as Soft Computing (SC).

So, in this paper we have experimented multilayer Perceptron based Approach for identification of Object oriented Reusable Component.

III. ARTIFICIAL NEURAL NETWORKS (ANN)

An Artificial Neural Network (ANN) [1, 6, and 13] is an information-processing paradigm that is inspired from the working of biological nervous system in human brain. Large number of neurons present in the human brain forms the key element of the neural network paradigm and act as elementary processing elements. These neurons are highly interconnected and work in unison to solve complex problems. Similarly, an Artificial Neural Network can be configured to solve a number of difficult and complex problems.

ANNs find a wide variety of applications in diverse areas including functional approximation, nonlinear system identification and control, pattern recognition and pattern classification, optimization, English text pronunciation, protein secondary structure prediction and speech recognition.

Shamsher Singh is with the CSE, PTU Jalandhar (e-mail: Shamsher_singh_86@yahoo.co.in ).
Er.Pushpinder Singh is Assistant Professor with the Department of CSE, RBIEBT, Kharar.
Er. Neeraj Mohan is Head of Department of CSE, RBIEBT, Kharar (e-mail: erneerajmohan@gmail.com).
A. Salient Features Of ANN

The biological neurons have many useful and significant characteristics and properties. These are also emulated by neurons in artificial neural networks. Some of these features of artificial neural networks are outlined as under:

- **Input-output Mapping:** In this the network learns from these I/O training samples by emulating the input-output mapping for the problem considered for that network.
- **Learning with Experience:** It is like, an ANN trained to operate in a specific environment can easily adapt to minor changes in the operating environmental conditions.
- **Nonlinearity:** In ANN, the nonlinearity is distributed throughout the network amongst the different neurons in different layers.
- **Model Free Environment:** ANN model does not require derivation of the cumbersome mathematical model of the process being considered for a particular application.
- **Hardware Implementation:** The ANNs can be implemented in parallel and also realized in VLSI chips. Hardware implementation gives additional speed to ANN.
- **Parallel Distributed Processing:** ANNs have a highly interconnected and parallel structure, which lends parallel implementation and results fast processing.
- **Multivariable Systems:** The structure of a neuron is designed to accept many inputs and outputs. Therefore, ANN’s are readily applicable to multivariable systems.
- **Fault Tolerance:** An artificial neural network, when implemented in hardware, has the capability to be inherently fault tolerant.
- **Data Fusion:** ANNs have an extra-ordinary ability to process simultaneously both quantitative and qualitative data.

B. Fundamentals of ANN Model

A neuron is a small processing unit and performs a simple computation that is fundamental to the operation of an ANN. Figure 1 shows the model of a neuron containing the basic elements like inputs, synaptic weights and bias, summing junction and activation function. The elements work together in unison to perform the computation.

![Figure 1 Model of a Neuron for Execution of Output](image)

1) **Interconnection Weights**

A set of inter-connecting links, characterized by a synaptic weight forms the weight matrix. Inputs to the input neurons are given directly. The output of these neurons is fed to the neurons of the next layer of the network by multiplying its amplitude with interconnection synaptic weight value as shown in Equation 1. Unlike a synapse in the brain, the synaptic weight of an artificial neuron may lie in a range that includes negative as well as positive values.

2) **Summing Function**

Summing junction termed as an adder sums up weighted input signal to any neuron. Thus, net input received by any neuron from the neurons of its previous layer becomes:

\[ X_n = \sum_{m=1}^{j} (W_{mn} \cdot X_m) \quad (1) \]

Where \( m = 1, 2, 3, 4 \ldots \) \( j \), the number of neurons in the previous layer and \( n = 1, 2, 3, 4 \ldots \) \( k \), the number of neurons in the next layer. The neuron model of figure 4.1 also includes bias input ‘X0’ which is +1 by default that is applied externally and bias value is denoted by \( b_n \). Therefore, net bias \( b_n \cdot X0 \) has the effect of addition or subtraction to the net input of the activation function, depending on whether net bias is positive or negative. Mathematically, the net input to neuron ‘Xn’ is described with the help of the following equation

\[ X_n = \sum_{m=1}^{j} (W_{mn} \cdot X_m) + b_n X_0 \quad (2) \]

3) **Activation Function**

An activation function is used for limiting the amplitude of the output of a neuron. The activation function is also called ‘squashing function’ as it squashes or limits the amplitude of the output signal to some finite limits. Many types of activation functions are available in MATLAB neural network toolbox and are described as under:

- ‘purelin’, i.e. purely linear function \( Y = mX + C \), here \( m = 1 \) and \( C = 0 \) by default.
- ‘logsig’, i.e. a sigmoid function \( Y = \frac{1}{1 + e^{-x}} \) that resembles a squashed ‘’ whose output ranges between 0 and 1 as net input ‘’ to the neuron varies from -∞ to +∞.

There are many other activation functions available. The sigmoid function is shown in figure 2.
Mathematically, the net output \( Y_n \) of a neuron is described with the help of the following pair of equations:

\[
X_n = \sum_{m=1}^{j} w_{mn} x_m + b_n x_0 \quad \text{and} \quad Y_n = \frac{1}{1 + e^{-X_n}}, \quad (3)
\]

The neuron net input \( X_n \) is squashed using sigmoid activation function and \( Y_n \) is the output of the neuron.

C. ANN’s Architectures

Number ANN’s can be divided into two major categories based on their connection topology:

1. Feed forward neural networks
2. Feedback neural networks

- Feed-forward neural networks allow the signal to flow in the forward direction only. The signals from any neuron do not flow to any other neuron in the preceding layer. Feed-forward networks, based on their architectures, are further classified into different categories i.e. Multilayer Perceptrons (MLPs), Counter-propagation Networks (CPN), Cerebellar Model Articulation Controller (CMAC) and Radial Basis Function Networks (RBF Nets).

- Feedback neural networks do not impose any such constraint on the flow of the signal in the network. The signal from a neuron in a layer can flow to any other neuron whether it be preceding or succeeding layers. The interconnections between neurons in the same layer are called lateral connections. Such neural nets are very much suitable for systems where output of the system under consideration is a function of not only the inputs but past outputs and inputs also. However, the drawback with these networks is that these are complex and are difficult to implement.

D. Multilayer Perceptron

In ANNs, neurons are arranged into groups called layers. The term Multilayer Perceptron [1] is self-explanatory, multi means more than one. Thus, Multilayer Perceptron has more than one layer of neurons in its network as shown in the figure 3. The description of each layer is given as under:

1) Input layer

It consists of a set of neurons equal to no. of input variables that receive inputs form the external environment. In this layer, there is no activation function; no processing of input variables and output of each neuron is the same as its input.

2) Hidden Layer

In any Multilayer Perceptron neural network, there can be a number of hidden layers between the input and output layers. It is mathematically proven fact that a single hidden layer is sufficient to approximate any function to any degree of accuracy provided there is enough number of neurons in the hidden layers. The number of neurons in a hidden layer is chosen very carefully to limit the complexity and at the same time to achieve better functional approximation accuracy. The function approximation accuracy improves with increase in the neurons in the hidden layer, but, the network complexity increases and the response time of network deteriorates. Both the factors are important and act contrary to one another. In order to maintain balance between the two opposing constraints, the judicious choice of the number of hidden layer neurons becomes very significant.

3) Output Layer

The output layer consists of neurons that communicate the output of the system to the user or the external environment. The number of neurons in the output layer is equal to the number of output variables. The neurons in this layer receive their input from the preceding layer or last hidden layer in the network.

E. ANN model for Modeling of Reusability of OO software System

Neural network can be used to evaluate the reusability of OO-based component using its structural attributes as inputs. An algorithm has been proposed in which the inputs can be given to Neural network in form of tuned WMC, DIT, NOC, CBO, LCOM values of the OO software component and output can be obtained in terms of reusability.
Hence in this study, we will experiment with Multi Level Perceptron based classification approach for the reusability prediction of object oriented systems.

IV. METHODOLOGY

Reusability evaluation System for Object Oriented Software Components can be framed using following steps:

A. Selection and refinement of metrics targeting the reusability of Object Oriented software system and perform parsing of the software system to generate the Meta information related to that Software.

The proposed five metrics for Object-Oriented Paradigm are as follows [15,17]:

1) Weighted Methods per Class (WMC)
2) Depth of Inheritance Tree (DIT)
3) Number of Children (NOC)
4) Coupling Between Object Classes (CBO)
5) Lack of Cohesion in Methods (LCOM)

B. Calculate the metric values of the sampled software components.

C. Use MLP (Multi Level Perceptron) based system for the Reusability Prediction:

D. The performance criterion taken is the classification Accuracy%. It is the percentage of the predicted values that match with the expected values of the reusability for the given data. The best system is that having the high Accuracy value.

The number of folds is fixed to 10, as long as the number of instances in the training set is not smaller than 10. If this is the case the number of folds is set equal to the number of instances. Deduce the results on the 10 fold cross validation accuracy, precision and recall values.

In case of the two-level output value based problem, the confusion matrix has four categories: True positives (TP) are modules correctly classified as Reusable modules. False positives (FP) refer to non-reusable modules incorrectly classified as non-reusable modules as shown in table 1.

<table>
<thead>
<tr>
<th>Predicted Value</th>
<th>Real Data Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reusable</td>
<td>TP</td>
</tr>
<tr>
<td>Non-Reusable</td>
<td>FP</td>
</tr>
<tr>
<td>Reusable</td>
<td>FN</td>
</tr>
<tr>
<td>Non-Reusable</td>
<td>TN</td>
</tr>
</tbody>
</table>

With help of the confusion matrix values the precision and recall values are calculated described below:

- Precision: Precision = TP / (TP + FP)
- Recall: Recall = TP / (TP + FN) [7]
- F-measure: F-measure is the product of precision and recall for a particular class. The higher value of F-measure is desirable for a good classifier system.
- Accuracy: The percentage of the predicted values that match with the expected values of the reusability for the given data.

The best system is that having the high Accuracy, High Precision and High Recall value.

V. RESULTS AND DISCUSSION

The object oriented dataset considered have the output attribute as Reusability value. The Reusability in the dataset is expressed in terms of six numeric labels i.e. 1 to 6. The label 1 represents Non-Reusable and the label 6 represents the “Excellent Reusable” Label. The statistics of the count of the number of examples of certain reusability label is shown in the Table 2.

<table>
<thead>
<tr>
<th>No.</th>
<th>Label</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>11</td>
</tr>
</tbody>
</table>

The graphical display of the above statistics is as follows:
The statistics shows that in the dataset, there are 48 examples of label 1 and 39 examples of label 2. The input attribute-wise statistical details of the count of the examples of the labels are shown in table 3, 4, 5, 6, 7. The input attributes are expressed in the three linguistic labels i.e. 1, 2, and 3. The label 1 corresponds to the Low value, label 2 corresponds to the Medium value and label 3 corresponds to the high value.

The given data is with five Input Attributes i.e. TWMC, LTNOC, LTNOC, LCBO, TLCOM, and one Output attributes named as Reusability Level of the Software Component. Then MLP based algorithm is run in WEKA environment. The following are the parameters used in the MLP algorithm implementation in the WEKA as shown in figure 6:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut</td>
<td>False</td>
</tr>
<tr>
<td>autobuild</td>
<td>True</td>
</tr>
<tr>
<td>debug</td>
<td>False</td>
</tr>
<tr>
<td>decay</td>
<td>False</td>
</tr>
<tr>
<td>HidenLayers</td>
<td>0</td>
</tr>
<tr>
<td>learningRate</td>
<td>0.3</td>
</tr>
<tr>
<td>momentum</td>
<td>0.2</td>
</tr>
<tr>
<td>normalizeBinaryFilter</td>
<td>True</td>
</tr>
<tr>
<td>normalizeAttributes</td>
<td>True</td>
</tr>
<tr>
<td>normalizeNumClass</td>
<td>True</td>
</tr>
<tr>
<td>seed</td>
<td>0</td>
</tr>
<tr>
<td>trainingTime</td>
<td>500</td>
</tr>
<tr>
<td>validationSize</td>
<td>0</td>
</tr>
<tr>
<td>validationThreshold</td>
<td>20</td>
</tr>
</tbody>
</table>

Fig. 6 Parameters used for the MLP

After the 10 fold cross validation the Correctly Classified Instances 62 i.e.71.2644 %. It means Incorrectly Classified Instances are 25 i.e. 28.7356 %. The Mean absolute error and Root mean squared error calculated are 0.109 and 0.2772 respectively.

On the basis of the confusion matrix the precision and recall is calculated for the different classes as shown in Fig. 7.

<table>
<thead>
<tr>
<th>Confusion Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>a 1 0 0 0</td>
</tr>
<tr>
<td>3 1 2 1 0 0</td>
</tr>
<tr>
<td>3 2 10 0 1 0</td>
</tr>
<tr>
<td>0 1 1</td>
</tr>
<tr>
<td>0 0 2 2</td>
</tr>
<tr>
<td>0 0 1 9</td>
</tr>
</tbody>
</table>

Table 8. Confusion Matrix

Fig. 7 Structure of MLP used
Fig. 7 Performance Results

VI. CONCLUSION

In this study Multi Level Perceptron (MLP) based classification approach is evaluated for Reusability Prediction of Object based Software systems. Here, the metric based approach is used for prediction. Reusability value is expressed in the Six linguistic values. Five Input metrics are used as Input and the training of the proposed system is performed, thereafter performance of the system is recorded for prediction of reusability of the software modules. After the 10 fold cross validation the Correctly Classified Instances is 62 i.e. 71.2644%. It means Incorrectly Classified Instances are 25 means 28.7356 %. The Mean absolute error and Root mean squared error calculated are 0.109 and 0.2772 respectively. The Recall value of “Reusability level 1” is the maximum among other classes and overall the F-measure value of all classes is satisfactorily high. It means that the Multi Level Perceptron (MLP) algorithm can satisfactorily predict reusability level of object oriented software components and this can further help in improving the value of the reusability of software components.

VII. FUTURE SCOPE

The proposed approach is applied on the C++ based software modules/components and it can further be extended to the Artificial Intelligence (AI) based software components e.g. Prolog Language based software components. It can also be tried to calculate the fault-tolerance of the software components with help of the proposed metric framework. The research work can be extended in the following directions:

- Intelligent Component Mining or Extraction algorithms can be developed
- Early prediction of the quality of component based system
- Characterization of Software Components for easy retrieval

REFERENCES