Chapter 5

Artificial Neural Network Based Approach

5.1 Introduction

Artificial Neural Network (ANN) are the models designed to work like a human brain. As the brain consists of neuron network, the neural network also has a network of neurons. The complexity of real neurons is highly abstracted when modeling artificial neurons. These basically consist of

i) A set of processing units called neurons

ii) An activation state for each unit, which is equivalent to the output of the unit

iii) Connections between the units. Generally each connection is defined by a weight $w_{jk}$ that determines the effect that the signal of unit $j$ has on unit $k$

iv) A propagation rule which determines the effective input of the unit from its external inputs

v) An activation function which determines the new level of activation based on the effective input and the current activation;

vi) An external input (bias, offset) for each unit

vii) A method for information gathering (learning rule)

viii) An environment within which the system can operate, provide input signals and, if necessary, error signals.

A processing unit also called a neuron or node performs a relatively simple job; it receives inputs from neighbors or external sources and uses them to compute an output signal that is propagated to other units.
Within the neural systems there are three types of units:

i) Input units which receive data from outside of the network;
ii) Output units which send data out of the network;
iii) Hidden units whose input and output signals remain within the network.

Each unit $j$ can have one or more inputs $x_0, x_1, x_2, \ldots x_n$, but only one output $z_j$. An input to a unit is either the data from outside of the network, or the output of another unit, or its own output.

Each non-input unit in a neural network combines values that are fed into it via synaptic connections from other units, producing a single value called the net input. The function that combines values is called the combination function, which is defined by a certain propagation rule. In most neural networks it is assumed that each unit provides an additive contribution to the input of the unit with which it is connected. The total input to unit $j$ is simply the weighted sum of the separate outputs from the connected units plus a threshold or bias term $\theta_j$:

$$ a_j = \sum_{i=1}^{n} w_{ji} x_i + \theta_j \quad ---(5.1) $$

The contribution for positive $w_{ji}$ is considered as an excitation and an inhibition for negative $w_{ji}$. The units with the above propagation rule are called sigma units. Lots of combination functions usually use a
"bias" or "threshold" term in computing the net input to the unit. For a linear output unit, a bias term is equivalent to an intercept in a regression model. It is needed in much the same way as the constant polynomial ‘1’ is required for approximation by polynomials.

Most units in the neural network transform their net inputs by using a scalar-to-scalar function called an activation function, yielding a value called the unit’s activation. Except possibly for output units, the activation value is fed to one or more units. Activation functions with a bounded range are often called squashing functions. Some of the most commonly used activation functions are identity function, binary step function, sigmoid function, etc. Sigmoid function is especially advantageous for use in neural networks trained by back-propagation, because it is easy to differentiate, and thus can dramatically reduce the computation burden for training. It applies to applications whose desired output values are between 0 and 1. It is defined by equation 5.2.

\[
g(x) = \frac{1}{(1 + e^{-x})} \quad -(5.2)
\]

The topology of a network is defined by the number of layers, the number of units per layer, and the interconnection patterns between layers. Feed-forward networks is a topology where the data flow from input units to output units is strictly feed-forward. The data processing can extend over multiple layers of units, but no feedback connections are present. That is, connections extending from outputs of units to inputs of units in the same layer or previous layers are not permitted. The functionality of a neural network is determined by the combination of the topology (number of layers, number of units per layer, and the interconnection pattern between the layers) and the weights of the connections within the network. The topology is usually fixed, and the weights are determined by a certain training
algorithm. The process of adjusting the weights to make the network learn the relationship between the inputs and targets is called learning, or training. Many learning algorithms have been invented to help find an optimum set of weights those results in the solution of the problems. One of the learning algorithms is supervised learning. The network is trained by providing it with inputs and desired outputs (target values). These input-output pairs are provided by an external teacher, or by the system containing the network. The difference between the real outputs and the desired outputs is used by the algorithm to adapt the weights in the network. It is often posed as a function approximation problem. Given training data consisting of pairs of input patterns \( x \), and corresponding target \( t \), the goal is to find a function \( f(x) \) that matches the desired response for each training input. The model is shown in Figure 5.2.

![Figure 5.2. Supervised Learning Scenario](Rumelhart et.al. 1986)

To train a network and measure how well it performs, an objective function (or cost function) must be defined to provide an unambiguous numerical rating of system performance. Selection of an objective function is very important because the function represents the design goals and decides what training algorithm can be taken. To develop an objective function that measures exactly
what is wanted is not an easy task. A few basic functions are very commonly used. One of them is the sum of squares error function,

\[ E = \frac{1}{NP} \sum_{p=1}^{P} \sum_{i=1}^{N} (t_{pi} - y_{pi})^2 \]  

where \( p \) indexes the patterns in the training set, \( i \) indexes the output nodes, and \( t_{pi} \) and \( y_{pi} \) are, respectively, the target and actual network output for the \( i^{th} \) output unit on the \( p^{th} \) pattern. [Zhanshau Yu 2010].

By adjusting the weights of an artificial neuron it is possible to obtain the output expected for specific inputs. But when an ANN consists of hundreds or thousands of neurons (arranged in the form of layers), it would be quite complicated to find by hand all the necessary weights. But there are algorithms which can adjust the weights of the ANN in order to obtain the desired output from the network. This process of adjusting the weights is called learning or training. Artificial neural nets play a vital role in pattern recognition. Since ontology building uses design patterns, use of neural net in these cases are of more benefit. Hence a feed forward two layer neural network is selected for the experiment. The training algorithm used is back propagation algorithm [Rumelhart et.al. 1986].

5.2 Proposed Method to Ontology Mapping

This method uses neural network based approach. Neural network is used to learn the different weights for the eighteen instances based and metadata based matchers during matching process. These matchers have been grouped into six groups, each group having an aggregate of the similarities computed in three measures using new aggregation method discussed in section 4.2.1. The details of grouping is shown in Table 5.1. The measures explained in chapter 3 are used for each group. For simulation, a two layer feed forward
neural network is adapted. The network has 6 inputs each corresponding to one type of similarity measure group and the network has six units in the input layer, ten units in the hidden layer and one output unit. A rough sketch of the neural network pattern used is shown in Figure 5.3. The input to the network is normalized vector $s$ representing a different type of similarity computed as explained earlier in Chapter 3. The output is 's' the similarity value between concept or entity $C_1$ and $C_2$ of $O_1$ and $O_2$. Initially six concept similarity matrices $M_i$ for $i=1$ to 6 is obtained. Then randomly a set of concepts or entities and an expected similarity value as per manual matching are prepared and this set is used to train the network with initially set random weights and biases. The order of the matrix depends upon the total number of entities in the ontologies being compared. In order to handle large ontologies, the order of the matrix considered per training and mapping extraction is kept as multiple of 500. Figure 5.3 shows the architecture of the net used.

Figure 5.3. Architecture of Neural Network
Table 5.1 Grouping of Matchers

<table>
<thead>
<tr>
<th>Groups</th>
<th>Matchers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group1(string based)</td>
<td>String, Hamming, Levenshtein</td>
</tr>
<tr>
<td>Group2(semantic)</td>
<td>Substring, Cosine, Synonymy</td>
</tr>
<tr>
<td>Group3(semantic)</td>
<td>Co-Synonymy, Path, Resnik</td>
</tr>
<tr>
<td>Group4(structure, semantic, instance)</td>
<td>Lin, Dice, Graph</td>
</tr>
<tr>
<td>Group5(instance, semantic, structure)</td>
<td>Instance, Distance, NAS</td>
</tr>
<tr>
<td>Group6(semantic, structure)</td>
<td>Lesk, EAS, DDS</td>
</tr>
</tbody>
</table>

5.3 Results of ANN Approach

The data sets of OAEI 2010 are used for rendering the results. The resulting precision and recall are shown in Table 4.7.

Table 5.2 Results of ANN based ontology mapping

<table>
<thead>
<tr>
<th>Datasets</th>
<th>precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benchmark</td>
<td>0.98</td>
<td>0.96</td>
<td>0.97</td>
</tr>
<tr>
<td>Anatomy</td>
<td>0.75</td>
<td>0.73</td>
<td>0.74</td>
</tr>
<tr>
<td>Conference</td>
<td>0.71</td>
<td>0.68</td>
<td>0.69</td>
</tr>
<tr>
<td>Small ontologies in other domains</td>
<td>1.00</td>
<td>0.90</td>
<td>0.94</td>
</tr>
</tbody>
</table>

It can be observed that the accuracy is good for anatomy and benchmark data sets. The Figures 5.4, 5.5 and 5.6 show the plot of training state values, regression and error histogram respectively, for two ontologies out of sixteen in conference data set described in chapter 1. The Figure 5.4 shows the gradient value at epoch 25. Lesser the gradient, better the training and testing result.
Figure 5.4 Training State Values

Figure 5.5 Regression Plot
It can be observed from the same figure that as number of epochs increase gradient value decreases. Error histogram in Figure 5.6 shows status of errors versus instances in the training and testing data set. Regression plot in Figure 5.5 shows regression of targets relative to outputs.

5.4. Summary

This chapter proposes a neural network approach to ontology mapping and states and discusses the results of this method. The method gives good accuracy for benchmark when compared to simple instance and metadata method. It is also applied to other data sets such as anatomy, conference and small ontologies set. It can be observed that the accuracy is 97% for benchmark, 74% for anatomy and 69% for conference and 94% for small ontologies data set. This is due to the supervised learning of the weightage to be given to various similarity groups and consideration of instance and metadata information present in ontologies.