International Journal of Research Studies in Science, Engineering and Technology
Volume 2, Issue 6, June 2015, PP 14-23
ISSN 2349-4751 (Print) & ISSN 2349-476X (Online)

Reverse Dictionary Using Artificial Neural Networks

Natasha Mehta
Computer Engineering
Thadomal Shahani Engineering College
Mumbai, India

Dhaval Karani
Computer Engineering, K.J. Somaiya Institute of Engineering and Information Technology
Mumbai, India

Abstract: *This paper proposes a neural network to treat programming languages. Most of the conventional neural networks can only process sentences consisting of a few words, and their applications are very simple such as metaphor understanding. The proposed network can process many complicated sentences and can be used as an associative memory and a question answering system. The online dictionary could be used as a guide to programming languages such as HTML, Java, MATLAB etc., where, if the user wants to know about any in-built function/component of which the prime functionality is known to him/her, the output would be the name of the required element.*

Keywords: *Neural Networks, Back Propagation Algorithm, Network Construction, Network Learning.*

1. INTRODUCTION

In computer science and related fields artificial neural networks are computational models inspired by animals' central nervous systems (in particular the brain) that are capable of machine learning and pattern recognition. They are usually presented as systems of interconnected "neurons" that can compute values from inputs by feeding information through the network.

For example, in a neural network for handwriting recognition, a set of input neurons may be activated by the pixels of an input image representing a letter or digit. The activations of these neurons are then passed on, weighted and transformed by some function determined by the network's designer, to other neurons, etc., until finally an output neuron is activated that determines which character was read. Like other machine learning methods, neural networks have been used to solve a wide variety of tasks that are hard to solve using ordinary rule-based programming, including computer vision and speech recognition. The adaptive weights are conceptually connection strengths between neurons, which are activated during training and prediction. Neural networks are also similar to biological neural networks in performing functions collectively and in parallel by the units, rather than there being a clear delineation of subtasks to which various units are assigned. The term "neural network" usually refers to models employed in statistics, cognitive psychology and artificial intelligence. Neural network models which emulate the central nervous system are part of theoretical neuroscience and computational neuroscience. In modern software implementations of artificial neural networks, the approach inspired by biology has been largely abandoned for a more practical approach based on statistics and signal processing. In some of these systems, neural networks or parts of neural networks (like artificial neurons) form components in larger systems that combine both adaptive and non-adaptive elements. While the more general approach of such systems is more suitable for real-world problem solving, it has little to do with the traditional artificial intelligence connectionist models. What they do have in common, however, is the principle of non-linear, distributed, parallel and local processing and adaptation. Historically, the use of neural networks models marked a paradigm shift in the late eighties from high-level (symbolic) artificial intelligence, characterized by expert systems with knowledge embodied in if-then rules, to low-level (sub-symbolic) machine learning, characterized by knowledge embodied in the parameters of a dynamical system.

Neural network models in artificial intelligence are usually referred to as artificial neural networks (ANNs); these are essentially simple mathematical models defining a function or a distribution over or both and , but sometimes models are also intimately associated with a particular learning algorithm or learning rule. A common use of the phrase ANN model really means the definition of a class of such functions (where members of the class are obtained by varying parameters, connection weights, or specifics of the architecture such as the number of neurons or their connectivity).

2. LEARNING OF THE PROPOSED NEURAL NETWORK

The proposed network can process many complicated sentences and can be used to decipher the input of reverse dictionary to give the favorable output. The flow of learning of the neural network of the proposed system is:

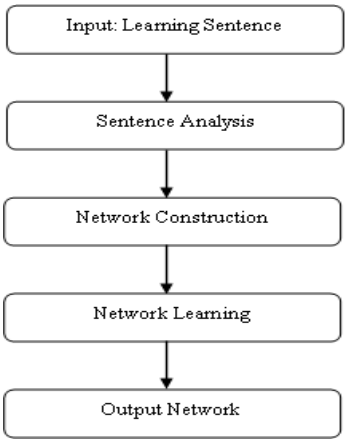


Fig1. *Flow of Learning.*

2.1. Sentence Analysis

When a sentence is input, it is analysed and divided into knowledge units. After that, deep case estimation is carried out which means expressing relation between verb and noun. Each knowledge unit and deep cases in a sentence are treated in different layers.

2.1.1. Division of Sentence to Knowledge Units

Division from a sentence to knowledge units is carried out based on a syntax analyzer explained in section 4.

2.1.2. Deep Case Estimation

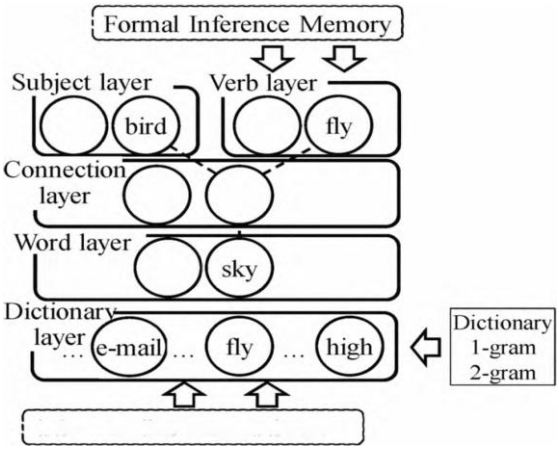


Fig2. *Structure of the Proposed Neural Network.*

Deep case estimation is done by dividing the knowledge units into subject, verb layer and a connection layer consisting of a neuron connecting the subject and the verb.

2.2. Network Construction

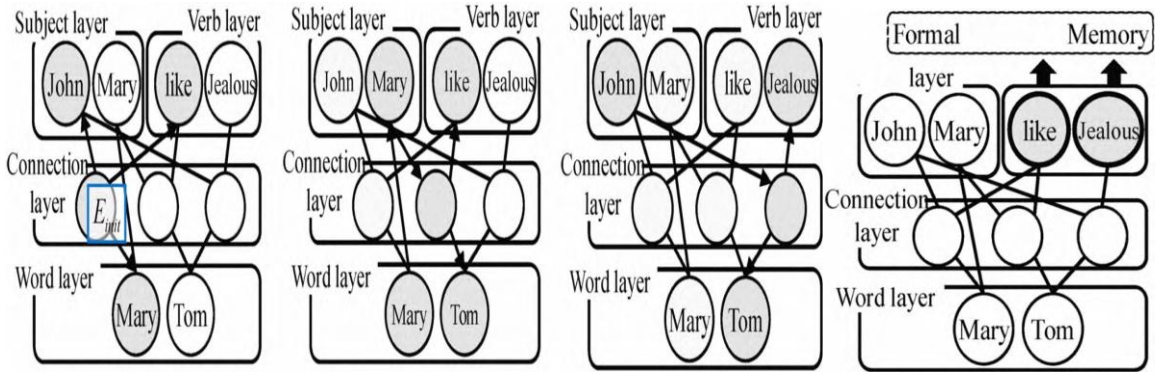
After the analysis of the input sentence the network construction is carried out. In this the input is divided into different layers as follows:

For Example: Bird flies in the sky.

Here, the bird is a neuron in the subject layer, flies is a neuron in the verb layer, sky is in the word layer. A Dictionary layer is the one which consists of similar words related to the sentence to increase the activation of the neuron.

2.3. Network Learning

After network construction the network learning is carried out using the back propagation algorithm. Here let's first determine our training data. Let's look at this example:



(a)Activation Propagation from Connection layer (b) Activation propagation from Word layer (c) Activation propagation from Subject layer (d) Memorize fired neurons

Fig3. *Flow of Network Learning.*

The figure above shows an example of a formal analogy. The network learned three sentences; "John likes Mary. Mary likes Tom. John is jealous of Tom." We explain the process when "like" and "jealous" are learned as a pair of predicates. At first, as shown in Fig.3 (a), activation Einit is given to the neuron in the Connection layer. The neuron is connected to "John" and "like" neurons. The given activation transmits through connections. This propagation corresponds to knowledge "John likes Mary ".Next as shown in Fig.3 (b), activation transmits from "Mary" neuron in the Word layer and "Mary", "Tom" and "like" neurons fire. At the same time, activation propagation as shown in Fig.3 (c) is carried out. In this case, because of activation propagation from "John" in the Subject layer, "Tom" and "jealous" neurons are fired. As a result "like" and "jealous" neurons in the Verb layer are fired and memorized as a pair as shown in Fig.3 (d).

In analogy to this, our training data could be

E.g.: Input: Function that can print a statement

Output: System.out.print ()

Here, we train the input in such a way, that it should give the output as System.out.println ().

Similar examples are taken and training is performed.

Example: Input: Function which can display a statement and so on. So an analogy is formed between System.out.println, print and display in the word layer.

Let's now see how the data is tested Testing of data takes place in three cases:

- Simple inference

At the time of the simple reasoning, the neuron in the Subject layer corresponding to the subject and that in the Verb layer corresponding to the predicate in the question sentence are given activation Einit. The activation is spread through connections. At first, the downward spread is carried out from the Subject layer and the Verb layer to the Dictionary layer. After that, upward spread is carried out from the Dictionary layer to the Subject layer and the Verb layer. Finally, the fired neurons are compared with the words in the question sentence. If all the neurons corresponding to the words in the question sentence fire, it is considered that the inference is possible. And the output is given.

- Formal Analogy

At first, simple inference is carried out. However, if the neuron is not activated and correct inference cannot be carried out. Then, the formal inference is carried out. At first, the downward spread of activation is carried out from neurons to lower layers. Then upward spread of activation is carried out as shown in Fig.3.4. As a result, neuron in the Word layer and neuron in the Verb layer fire. In the Formal Inference Memory, the word which is easy to be associated with the word given is searched. Then the candidate is discovered and activation is given to the neuron. Because all the neurons of the words contained in the question sentence fire, we can get the correct output.

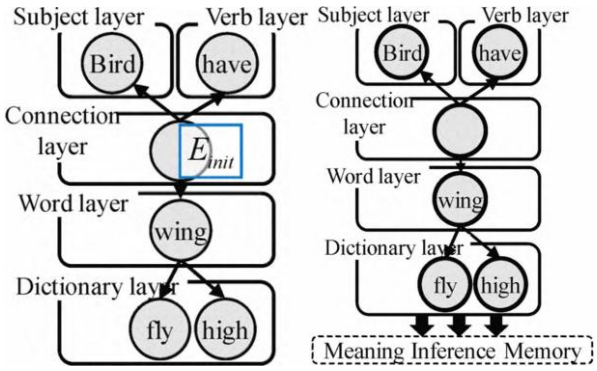
- Semantic Inference

When the knowledge of the question sentence cannot be recalled even by formal inference, the following inference by semantic inference is performed. Let's take an example:

The figure below shows an example of learning and memorization of the knowledge "Bird has a wing". At first, the neuron in the Connection layer connected to "bird' and "have" neurons is given activation Einit. The Activation propagates and a group of neurons, "bird," "have," the neuron in the connection layer, "wing," "fly," and "high," are memorized in the Meaning Inference Memory. The condition for this kind of memorization is that one neuron fires both in the Subject layer and in the Verb layer. This is because let the group of neurons to be memorized correspond to one sentence. Hence by analogy if:

Input- Something to display

Output- System.out.println ()



(a) Activation propagation (b) Memorize Fired Neurons

Fig4. *Example of Meaning Analogical Inference.*

3. NEURAL NETWORK

The network is constructed using the Back Propagation training. The Back propagation neural network is a multilayered, feedforward neural network and is by far the most extensively used. Back

propagation works by approximating the nonlinear relationship between the input and the output by adjusting the weight values internally. It can further be generalized for the input that is not included in the training patterns (predictive abilities).Generally, the Back propagation network has two stages, training and testing. During the training phase, the network is "shown" sample inputs and the correct classifications. For example, the input might be an encoded picture of a face, and the output could be represented by a code that corresponds to the name of the person.

4. STEPS FOR IMPLEMENTING THE REVERSE DICTIONARY NEURAL NETWORK

4.1. Analyzer

The system consists of five important databases like the input, output, verbs, keywords, predefined words. The input database consists of all possible input type for HTML language. The output database consists of the corresponding output HTML tags. The verbs database consists of all possible verbs that can appear in the input type with the respective synonyms. The keyword database consists of the keywords that appear in the input type. The analyzer takes the sentence as the input divides the sentence as verb, keyword and predefined words by matching each words from the input type to each of the three databases. Each of these words and input type have been provided a unique value using the following formulae:

For verbs:

NE_verb = NE_verb + (NE_v (j)*(10^ ((count*2)))) (1)

Where NE_verb is the current value stored at that variable, initially initialized to zero.

NE_v (j) is the value of the particular word found in the sentence.

Count is the counter variable.

Similarly, values for keywords, predefined words found in the input sentence are calculated and the total of these values is assigned to the input sentence. In this way the value for each input sentence is calculated. Thus, the role of the analyzer here is to identify the words of more importance i.e. verbs, keywords, predefined words which further help in the construction of the neural network.

4.2. Back Propagation Algorithm

The network is constructed using the Back Propagation training. The Back propagation neural network is a multilayered, feedforward neural network and is by far the most extensively used. Back propagation works by approximating the nonlinear relationship between the input and the output by adjusting the weight values internally. It can further be generalized for the input that is not included in the training patterns (predictive abilities).

Generally, the back propagation network has two stages, training and testing. During the training phase, the network is "shown" sample inputs and the correct classifications. For example, the input might be an encoded picture of a face, and the output could be represented by a code that corresponds to the name of the person.

The operations of the Back propagation neural networks can be divided into two steps: feedforward and Back propagation. In the feedforward step, an input pattern is applied to the input layer and its effect propagates, layer by layer, through the network until an output is produced. The network's actual output value is then compared to the expected output, and an error signal is computed for each of the output nodes. Since all the hidden nodes have, to some degree, contributed to the errors evident in the output layer, the output error signals are transmitted backwards from the output layer to each node in the hidden layer that immediately contributed to the output layer. This process is then repeated, layer by layer, until each node in the network has received an error signal that describes its relative contribution to the overall error.

Pseudo code: Initialize all weights with small random numbers, typically between -1 and 1

Repeat

For every pattern in the training set

Present the pattern to the network

// propagate the input forward through the network:

For each layer in the network for every node in the layer

1. Calculate the weight sum of the inputs to the node

2. Add the threshold to the sum

3. Calculate the activation for the node, end

// propagate the errors backward through the network

For every node in the output layer calculate the error signal, end

For all hidden layers for every node in the layer

1. Calculate the node's signal error

2. Update each node's weight in the network, end

// Calculate Global Error

Calculate the Error Function

End while ((maximum number of iterations < than specified) AND (Error Function is > than specified))

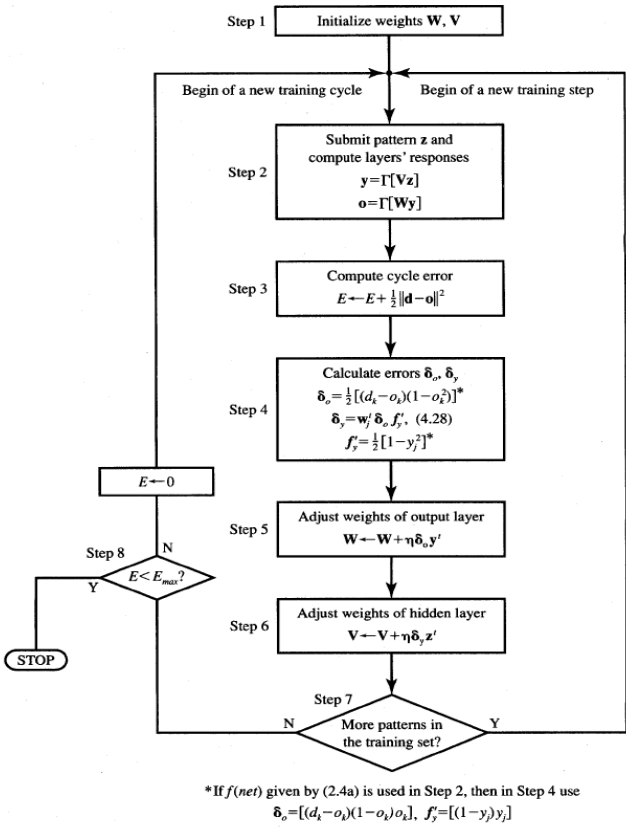


Fig5. *Back Propagation Algorithm.*

4.2.1. Implementation of Tangent Hyperbolic Function

To implement the neuron, various nonlinear activation functions, such as threshold, sigmoid, and hyperbolic tangent can be used. Hyperbolic tangent and sigmoid are mostly used because their differentiable nature makes them compatible with back propagation algorithm.

The hyperbolic tangent function is given by equation:

$$\tan(x) = \frac{\tan(x) - \exp(-x)}{\tan(x) + \exp(-x)} \quad (2)$$

Hyperbolic tangent is an odd function

$$\tanh(-x) = -\tanh(x) \quad (3)$$

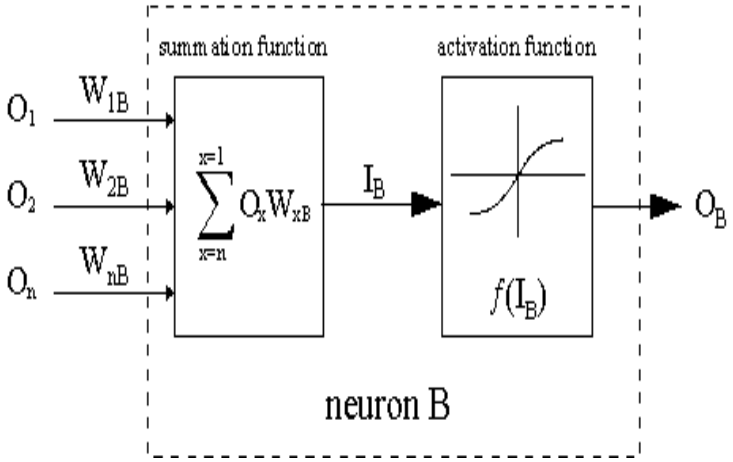


Fig6. *Tangent Hyperbolic Function.*

The dashed line represents a neuron *B*, which can be either a hidden or the output neuron. The outputs of *n* neurons (*O 1 ...O n)* in the preceding layer provide the inputs to neuron *B*. If neuron *B* is in the hidden layer then this is simply the input vector.

These outputs are multiplied by the respective weights (*W1B...WnB*), where *WnB* is the weight connecting neuron *n* to neuron *B*. The summation function adds together all these products to provide the input, *IB*, that is processed by the activation function *f* (**.**) of neuron *B*.

f (*IB*) is the output, *OB*, of neuron *B*.

For the purpose of this illustration, let neuron 1 be called neuron *A* and then consider the weight *WAB* connecting the two neurons. The approximation used for the weight change is given by the delta rule:

$$W_{ABnew} = W_{ABold} - \eta \frac{\delta E^2}{\delta W_{AB}} \quad (4)$$

Where **η** is the learning rate parameter, which determines the rate of learning, and

$$\frac{\delta E^2}{\delta W_{AB}} \quad (5)$$

Is the sensitivity of the error, *E*2, to the weight *WAB* and determines the direction of search in weight space for the new weight *WAB (new)* as illustrated in the figure below.

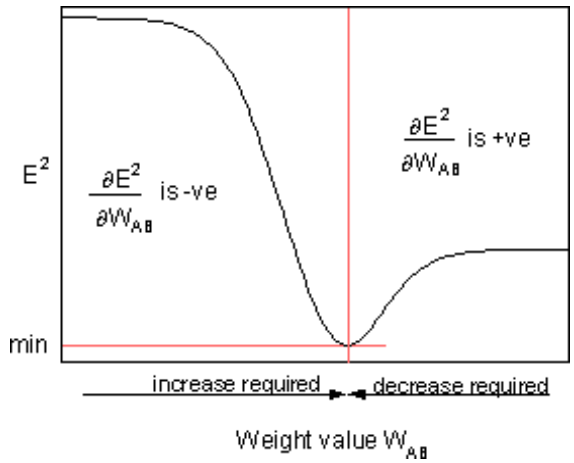


Fig7. *Weights Direction.*

Example:

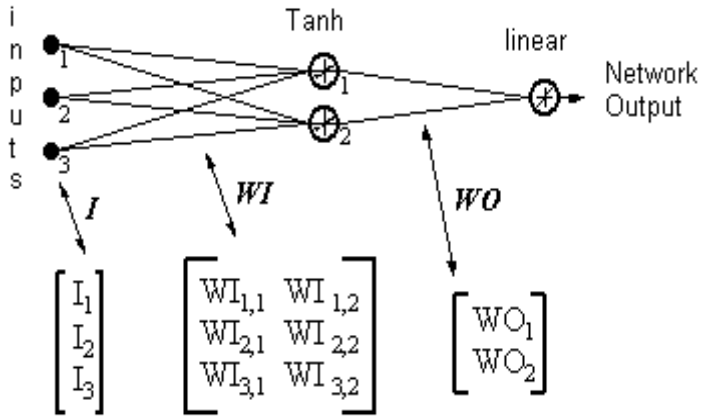


Fig8. *Tangent Hyperbolic Function Example*

Network Output: = $[\tanh (I^T.WI)]^T$- the output of the hidden neurons.

Let HID = $[\text{Tanh } (I^T.WI)]^T$ - output of the hidden neurons.

LR = Learning Rate.

The weight updates become linear output neuron,

$$WO = WO - (LR \times ERROR \times HID) \quad (6)$$

Tanh hidden neuron

$$WI = WI - \{LR * [ERROR * WO * (1 - HID^2)]. I^T\}^T \quad (7)$$

Equations (6) and (7) show that the weight change is an input signal multiplied by a local gradient. This gives a direction that also has magnitude dependent on the magnitude of the error. If the direction is taken with no magnitude then all changes will be of equal size which will depend on the learning rate. The algorithm above is a simplified version in that there is only one output neuron. In the original algorithm more than one output is allowed and the gradient descent minimizes the total squared error of all the outputs. With only one output this reduces to minimizing the error.

5. TESTING IMPLEMENTATION

The thirty percent of the data reserved is now tested using the trained neural network to check the system for giving the accurate results. The results of the testing implementation are as follows:

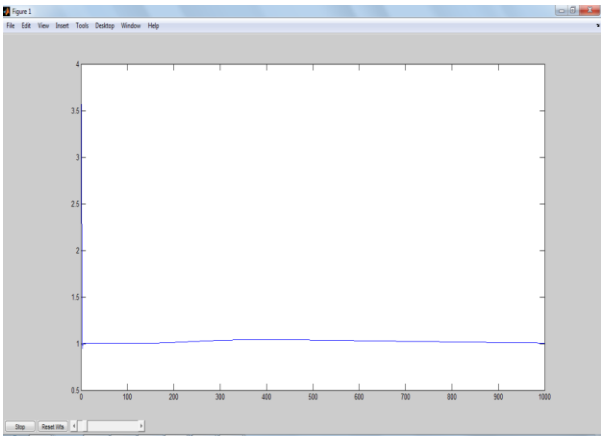


Fig9. *Testing Graph 1*

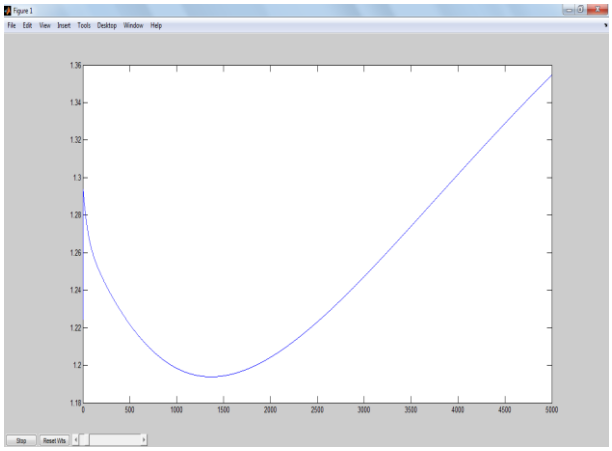


Fig10. *Testing Graph 2*

The Tangent Hyperbolic function implementation gives an efficiency of approximately 86%.

6. NETWORK APPLICATION EXAMPLE

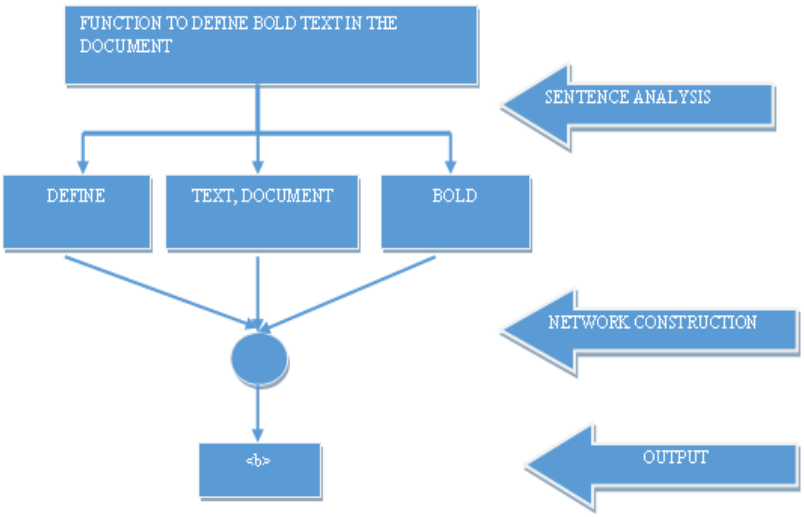


Fig11. *Application Network*

7. CONCLUSION

The system developed is a simple and user – friendly guide to progamming languages for beginners. This application can be scalable over different lanaguages and a variety of inputs. Futthermore, adavnced methods of neural networks can be used and the efficiency of the system can be improved to 100%.

REFERENCES

[1] Masahiro Saito and Masafumi Hagiwara: "Natural Language Processing Neural Network for Analogical Inference".

[2] Tsukasa Sagara and Masafumi Hagiwara: "Natural Language Neural Network and its Application to Question-Answering System", WCCI2012 IEEE World Congress on Computational Intelligence June, 10-15, 2012 - Brisbane, Australia.

[3] Ryan Shaw, Anindya Datta, Debra VanderMeer, Kaushik Dutta:" Building a Scalable Database-Driven Reverse Dictionary", IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 25, NO. 3, MARCH 2013.

[4] R.C Chakraborty, "Back Propagation Network.

[5] Artificial Neural Networks. [Online] Available: http://en.wikipedia.org/wiki/Artificial_neural_network.

AUTHORS' BIOGRAPHY

Natasha Mehta received the B.E degree in Computer Science from Thadomal Shahani Engineering College in Mumbai.

Dhaval Karani received the B.E degree in Computer Science from K.J. Somaiya Institute of Engineering and Information Technology in Mumbai.