Design and Implementation of Neural Processor for Parsing Manufacturing Query Language

Mr. Girish R. Naik

Associate Professor
Production Department
KIT's College of Engineering Kolhapur,India
girishnaik2025@gmail.com

Dr. V.A.Raikar

Director
Sanjay Ghodawat Group of Institutions, Atigre,
Kolhapur, India
var312@yahoo.com

Dr. Poornima G. Naik

Professor
Department of Computer Studies
Chh Shahu Institute of Business Education and Research,
Kolhapur, India
luckysankalp@yahoo.co.in

Abstract

Practically, all the approaches employed for parsing with natural languages use some or other type of neural network architecture and some typical statistical function for obtaining a parsing decision. In parsing with neural networks an incremental tree is usually obtained by using a set of rules for connecting a possible parse tree to the previously obtained incremental tree. In the current work, linguistic data is mapped to corresponding part-of-speech tags, which are then converted into a set of binary input vectors for each sentence. The tags have relationships with their neighbours which are modeled by the neural processor. When input is given to the neural processor, these relationships are analyzed and the string with the correct placement of parts-of-speech tag is output as syntactically correct else is declared as syntactically incorrect. A single layer network with back propagation is employed which utilizes a method based on minimization of error between the desired and actual activation of output nodes. A model is developed for dynamically accepting a query in natural language in the presentation tier of multi layered architecture which is processed and sent to the middle tier interfaced with R Software and MatLab for training the neural network and testing the query input by the user. The requisite Excel file in CSV format are generated and processed in the data layer. The entire approach is rendered generic and can be applied to similar cases containing the training data in the requisite format in Excel file. The confusion matrices generated by both the softwares are compared for judging the accuracy of classification.

Keywords- Classification, MatLab, Natural Language, Pattern Matching, R Software, Supervised Neural Network, Tokens

I. INTRODUCTION

Neural networks aim at development of a computational device operating at a faster rate than the traditional system. Artificial neural networks (ANNs) have gained tremendous importance in performing various tasks such as pattern matching and classification, optimization functions, vector quantizations and data clustering which prove to be difficult and time consuming in traditional systems. ANNs possess large number of highly interconnected processing elements called nodes which usually operate in parallel and have an internal state of their own which is referred to as an activation level of neuron. Each neuron is connected with the other by a connection link. Each connection link is associated with weights which contain information about input signal. The information is utilized by a neural net for solving a particular problem. ANNs collective behaviour is characterized not only by their ability to learn but also to recall and generalize training patterns. The main property of ANN is its capability to learn. Training a neural network is a process by means of which a neural network adapts itself to a stimulus by making proper parameter adjustments resulting in a desired output. EASE OF USE

Supervised Learning

In a supervised learning each input vector requires a corresponding target vector which represents the desired output. The input vector along with the target vector is referred to as a training pair. The network here is informed precisely about what should be emitted as output. The pattern matching capabilities of neural networks can be exploited for parsing a natural language. A typical neural network can capture the information implicit in training data which can then be used to model a relationship between input and output data. The declarative sentences can be decomposed into pre-subject, subject and predicate and each of the constituents can be decomposed further into basic grammatical elements which can then be encoded to constitute valid patterns used for training a neural network. The neural processor maps the grammar onto each sentence and computes the syntactical validity of a sentence. In a sentence, some constituents may be empty which can be signified with a numeric 0.

The neural networks outweigh other parallel stochastic approaches where they bear the ability to judge the correctness of sentence that has never occurred in the training set. Neural networks can model both probabilities and improbabilities. Another area where neural networks score high as compared to other similar approaches is the reduction in the computation time involved in problem solving. Suppose the runtime of the algorithm with the conventional approach is R, then for n test runs, the total computation time required is nR. In contrast to this, if the training time for the neural network is T and t is the processing time for an given set of input vector, then for n test runs, the total computation time is given by,

$$T + nt$$

where, $nR >> T+nt$,

as the training is done in advance, the runtime is tremendously reduced.

In the current work, linguistic data is captured from the user and is automatically mapped to corresponding part-of-speech tags, which are then converted into a set of binary input vectors for each sentence employing numeric mapping of part-of-speech tokens. The tags have relationships with their neighbours which are modeled by the neural processor. When input is given to the neural processor, these relationships are analyzed and the string with the correct placement of parts-of-speech tag is output as syntactically correct else is declared as syntactically incorrect. A single layer supervised neural network with back propagation is employed which utilizes a method based on minimization of error between the desired and actual activation of output nodes. Since the network has no hidden layer, the incorrectness in output can be attributed to weights and weight updates can be triggered from the error terms, which is referred to as an external error measure in contrast to internal error measure occurring due to one or more hidden layers.

Tools and Techniques.

R Software

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

R is an integrated suite of software applications employed for data manipulation, calculation and graphical display.

R has

- an effective data handling and storage facility,
- a large, coherent, integrated collection of intermediate tools for data analysis
- graphical facilities for data analysis and display and
- a well developed, simple and effective programming language (called ‘S’) which includes conditionals, loops, user defined recursive functions and input and output facilities. Most of the system supplied functions are themselves written in the S language.

The R functions employed in the current work for training a neural network are depicted in Table I.

TABLE I. R FUNCTIONS EMPOYED FOR TRAINING A NEURAL NETWORK

Function Name	Description
read.csv()	Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.
table()	table uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels.
sample()	sample takes a sample of the specified size from the elements of x using either with or without replacement.
setdiff()	Performs set union, intersection, (asymmetric!) difference, equality and membership on two vectors.
library()	loads and attaches add-on packages.
nnet()	Fit single-hidden-layer neural network, possibly with skip-layer connections.

MatLab

MatLab incorporates neural network and neural network pattern recognition tools for constructing, training and simulating a neural network. Neural network pattern recognition tool has a built-in feature for generating confusion matrix for displaying correct classifications and misclassifications visually. Confusion value indicates the fraction of values misclassified.

The MatLab functions employed in the current work for training a neural network are depicted in Table II.

TABLE II. MATLAB FUNCTIONS EMPOYED FOR TRAINING A NEURAL NETWORK

Function Name	Description
view()	Launches a window that shows neural network as a graphical diagram.
plotroc()	Plots the receiving operating characteristics for each output class. More the curve touches the top and left edges of the plot, the better is the classification.
plotconfusion()	Displays the classification confusion grid.
plotperform()	Plots the training, validation and test performances given the training record TR returned by the function train.
train()	Trains the neural network according to the input and target data and training function.
sim()	Simulates neural networks.

II. LITERATURE REVIEW

In recent years, many neural network models have been proposed for pattern classification, function approximation and regression problems. In literature there exist numerous papers for parsing natural language using variety of techniques which focus on techniques such as Deterministic Finite Automata, stochastic models and neural networks [1-5]. However, data driven, neural methods overweigh the benefits offered by various stochastic approaches. Neural nets can capture more of the implicit information in the training data since they can model negative as well as positive relationships [6-10].

Almost all current dependency parsers classify based on millions of sparse indicator features. Not only do these features generalize poorly, but the cost of feature computation restricts parsing speed significantly. In their work, Chen et.al [11] have proposed a novel way of learning a neural network classifier for use in a greedy, transition-based dependency parser. Because their classifier learns and uses just a small number of dense features, the authors claim that it can work very fast, while achieving about 2% improvement in unlabeled and labeled attachment scores on both English and Chinese datasets.

ANNs are practically “black boxes”, due to the complexity of the networks. Network pruning offers another approach for dynamically determining an appropriate network topology. Pruning techniques [12] begin by training a larger than necessary network and then eliminate weights and neurons that are deemed redundant. Typically, methods for removing weights involve adding a penalty term to the error function [13]. In their work authors of paper [14] have made an attempt to open up these black boxes by reducing the complexity of the network. The factor makes this possible is the pruning algorithm. By eliminating redundant weights, redundant input and hidden units are identified and removed from the network. Using the pruning algorithm, the authors been able to prune networks such that only a few input units, hidden units and connections left yield a simplified network. Experimental results on several benchmarks problems in neural networks show the effectiveness of the proposed approach with good generalization ability.

III. SUPERVISED LEARNING MODEL FOR PARSING MQL SENTENCES.

The authors have designed Manufacturing Query Language (MQL) to aid a manufacturing organization a quick selection of manufacturing method based on single/multi objective and/or single/multi/function [15-17]. In the current work, the authors have designed and implemented a neural processor for parsing MQL sentences. The training data consists of the MQL sentences depicted in Table III. along with the different combinations of all invalid statements.

TABLE III. SAMPLE MQL SENTENCES

List All Objectives
List All Methods
List All Functions
List All Methods in ClassS | ClassT | ClassP | ClassM | ClassX
List All Methods Meeting Objective1|Objective2|. . . . |Objective16
List All Methods Meeting Objective1|Objective2|. . . . |Objective16 And Function1.1|. . . . . |Function4.6
List All Methods Meeting Objective1|Objective2|. . . . |Objective16 And Function1.1|. . . . . |Function4.6 in ClassS | ClassT | ClassP | ClassT | ClassX
The language contains the following tokens.
{ List, All, Objectives, Methods, Functions, ClassS, ClassT, ClassP, ClassM, ClassX, Meeting, Objective1, . . ., Objective16, Function1.1, . . . ,Function4.6 }

The shortest sentence is broken into parts of speech tokens as shown in Figure 1.

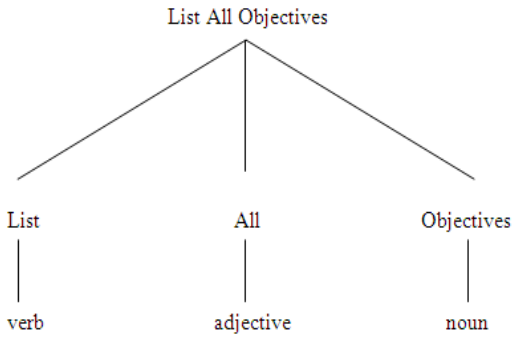


Figure 1. Decomposition of Shortest MQL Sentence into Parts of Speech Tags.

Similarly, a longest sentence is broken into different parts of speech tokens as shown in Figure 2.

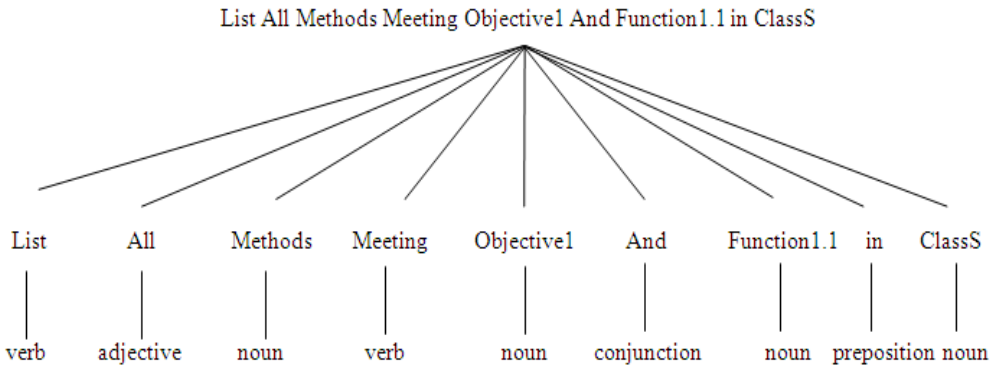


Figure 2. Decomposition of Longest MQL Sentence into Parts of Speech Tags.

Hence the sentences shown in Table III can be grouped into one of the six patterns shown in Table IV.

TABLE IV. POSSIBLE DECOMPOSITION OF MQL SENTENCES

verb-adjective-noun
verb-adjective-noun-preposition-noun
verb-adjective-noun-verb-noun
verb-adjective-noun-verb-noun-preposition-noun
verb-adjective-noun-verb-noun-conjunction-noun
verb-adjective-noun-verb-noun-conjunction-noun-prepotition-noun

Numeric encoding is employed for constructing an input vector to a neural network using the mapping shown in Figure 3.

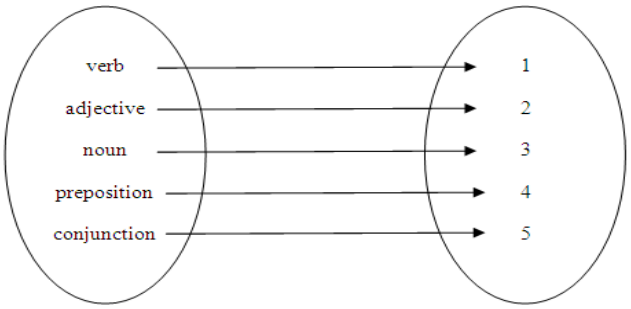


Figure 3. Numeric Mapping of Parts of Speech Tokens**.**

The above encoding mechanism yields the set of valid patterns shown in Table V. Different sentences differ in size of tokens as some of parts of speech elements may be missing. The maximum size of the input vector is 9. To yield input vectors of uniform size equal to the maximum size 0 padding semantics is employed as shown in Table 5.

TABLE V. GENERATION OF INPUT VECTORS FOR NEURAL NETWORK

123000000
123430000
123130000
123134300
123135300
123135343

Application Architecture

Figure 4. depicts the layered architecture for the interaction between different layers comprising the application where the VB application situated in presentation tier interacts with R Software and retrieves and stores data from/to corresponding Excel files in CSV format. Figure 5. shows capturing, processing and generating a response for user input entered in a natural language.

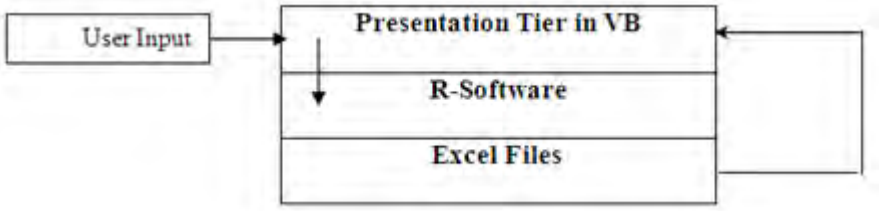


Figure 4. Layered Application Architecture.

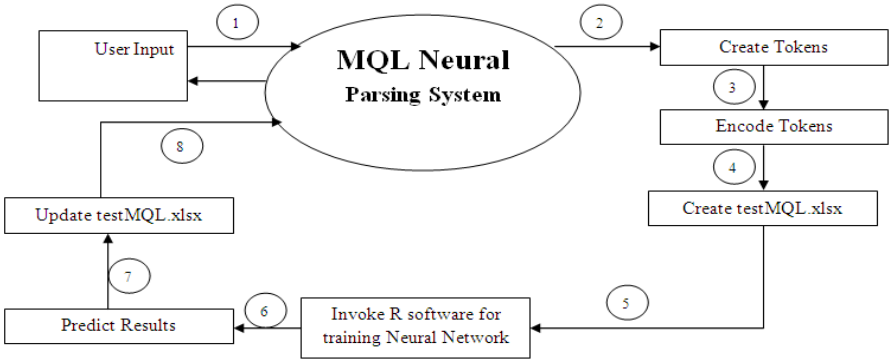


Figure 5. Processing of User Input by Supervised Neural Network.

Control Flow Diagram

The control flow diagram for reading and parsing MQL sentence is shown in Figure 6.

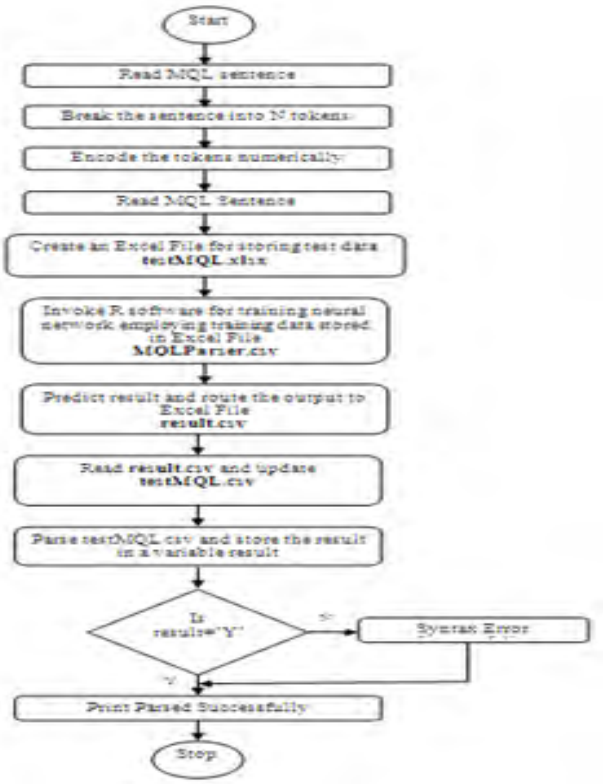


Figure 6. Control Flow Diagram for Parsing of MQL Sentence by Neural Network

Proposed Algorithm

```
/* Algorithm in C-Style */
/*
```

Every high-level language has built-in string manipulation functions present in a string library. The following functions assume the existence of the following string manipulation functions.

instr() – Accepts two string arguments and returns the position of the second string within a first string, if the string is not found returns -1.

right() – Accepts two arguments of type string and int, respectively and returns a substring of a string passed as the first argument containing rightmost n characters passed as the second argument to a function.

Purpose of some functions listed below are only described and their implementation is language dependent.

save_pattern("mqltest.csv") - Function is used for storing the patter in Excel CSV file format which contains 10 columns containing 9 words and accept column which is left blank.

create_file("temp.R") - Function which dynamically generates R file containing a formula and nnet function for training neural network.

create_and_execute("run.bat") - Function which dynamically generates a batch file for environment variable settings and execution of R Script.

save_results("mqlout.csv) - Function for storing the results of the test data in mqlout.csv file.

update_file("mqltest.csv","mqlout.csv") - Function for updating the mqltest.csv file based on the contents of mqlout.csv file which fills out accept column left blank in save_pattern() function.

result=read_from_file("mqltst.csv") - Function which reads a value in accept column of mqltst.csv file and assigns the value to result variable.

```
*/
struct Token
{
   char name[50];
   char type[20];
   int code;
}
char words[10][10];
int cntWords;
char syntax[10];
char query[50];
Token t[50];
/* Main function for parsing the given sentence using Neural Network */
function parse()
{
          initialize_tokens();
       read sentence;
       cntWords=count_words(sentence);
          split_words(sentence);
       for(i=1;i<cntWords ;i++)
            {
       for( ii = 1 ;ii< 18;ii++)
           {
found = false;
  if (UCase(Word(i)) = UCase(t(ii).name))
 {
  found = True
  pattern = pattern & CStr(t(ii).code)
    break;
 }
 }

 if (found = =false)
 {
  pattern = pattern & "0"
 }
   }
save_pattern("mqltest.csv");
```

```
create_file("temp.R");
create_and_execute("run.bat");
save_results("mqlout.csv);
update_file("mqltest.csv","mql_out.csv");
result=read_from_file("mqltst.csv");
        if (result=="y")
             print "Parsed successfully...";
           else
          print "Syntax Error!"
}
/* Function for initializing standard tokens in one of the cateories and encoding numerically */
function initialize_tokens()
{
t(0).name = "List"
t(0).type = "VERB"
t(0).code = 1
.
t(17).name = "Meeting"
t(17).type = "VERB"
t(17).code = 1
}
/* Function for returning No. of words in a given sentence */
function int count_words(char sentence[10])
{
        int count=0;
        int pos;
        pos=instr(sentence," ");
           while (pos != -1)
        {
                    count++;
                    sentence=right(sentence,pos+1);
                    pos=instr(sentence," ");
        }
            return count;
}

/* Function for splitting the words in a given sentence and storing them in words array */
function split_words(char sentence[10])
{
    words=sentence.split(" ");
}
```

Figure 7. shows dynamically generated R File. The VB code for dynamic generation of R file is shown in Appendix A.

```
#ANN
mql<- read.csv("D:/R systems/mqlparser.csv")
table(mql$accept)
mqlTrain = sample(1:22,20)
mqlVal = setdiff(1:22,mqlTrain)
mql[mqlTrain,]
mql[mqlVal,]
table(mql[mqlTrain,]$accept)
table(mql[mqlVal,]$accept)
library(nnet)
formula <- accept ~ word1+word2+word3+word4+word5+word6+word7+word8+word9
studANN = nnet(formula, data=mql, subset=mqlTrain, size = 10, rang = 0.2,decay = 5e-4, maxit = 200)
table(mql$grade[mqlTrain], predict(mqlANN, mql[mqlTrain,], type = "class"))
table(student$grade[stVal], predict(studANN, mql[mqlVal,], type = "class"))
predict(mqlANN, mql[mqlTrain,], type="class")
predict(mqlANN, mql[mqlVal,], type="class")
```

Figure 7. Dynamically Generated R File

IV. RESULTS AND DISCUSSIONS.

The model proposed above is implemented in VB which interfaces with R software and MatLab for constructing and training a neural network. Figure 8-10. show the Graphical User Interface (GUI) residing in presentation tier for supplying training and test data to R software and accepting MQL query in human language for parsing. The Excel file in CSV format storing test data which is generated by VB application is shown in Figure 11. The content of mqlout.csv file generated on execution of R script is shown in Figure 12. which in turn is used for updating the contents of mqltest.csv file.

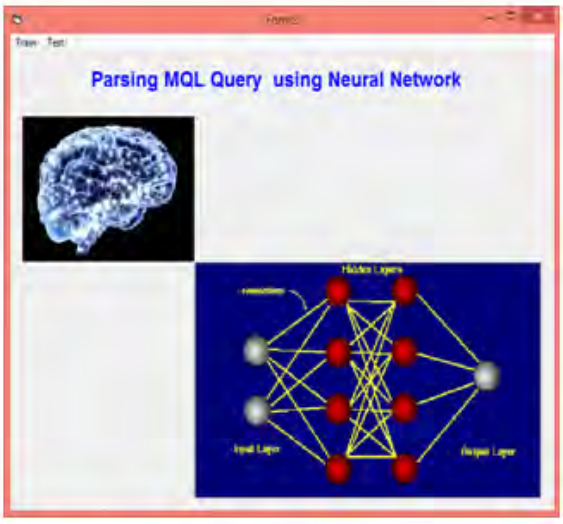


Figure 8. GUI for User Interaction with Neural Network.

Figure 9. GUI for Supplying Data for Training Neural Network.

Figure 10. GUI for Supplying MQL Query to Neural Network

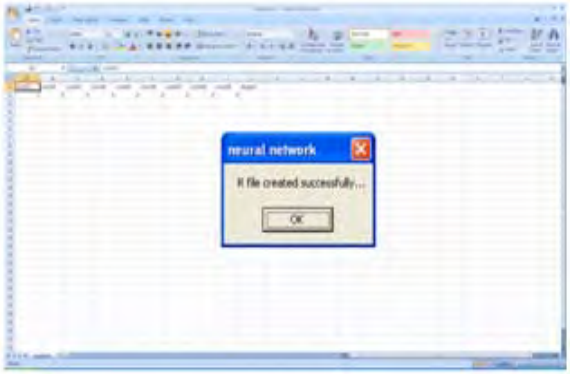


Figure 11. Dynamically Generated Excel File in CSV Format containing Test Data.

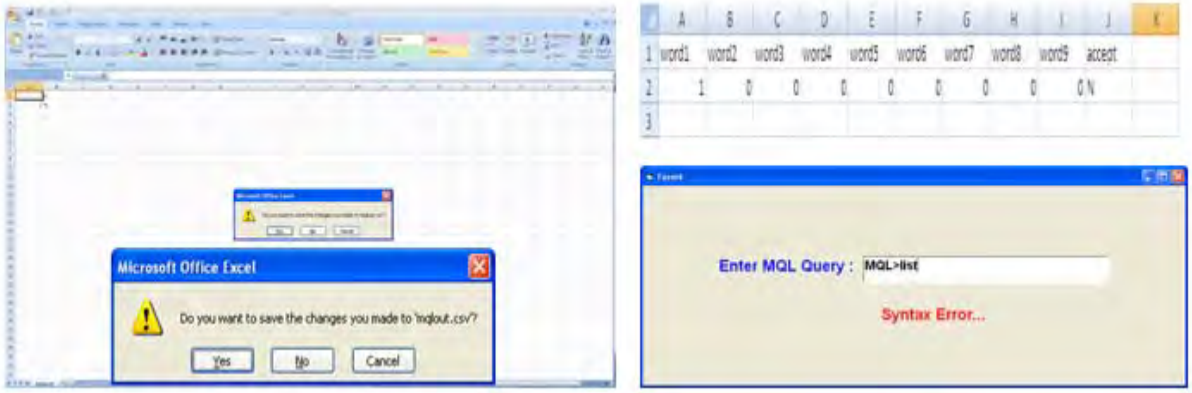
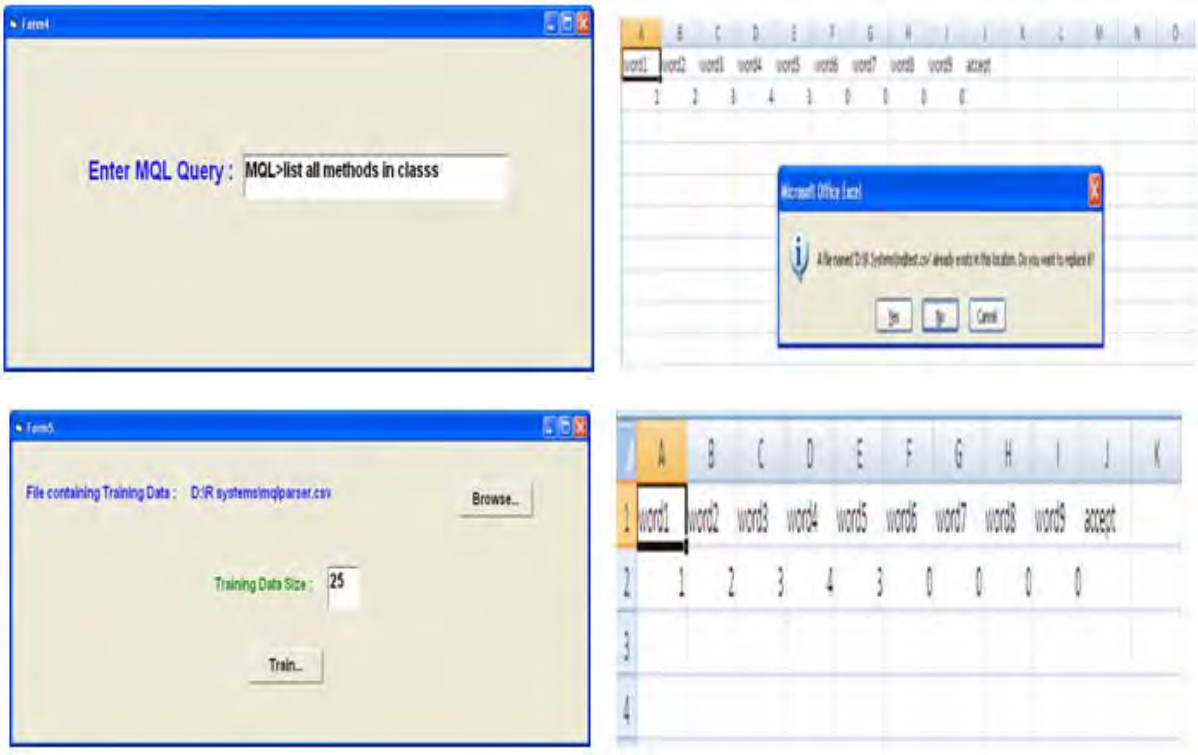


Figure 12. Excel File Generated on Execution of R Script.

Figures 13-19 show similar steps for parsing of another MQL sentence.



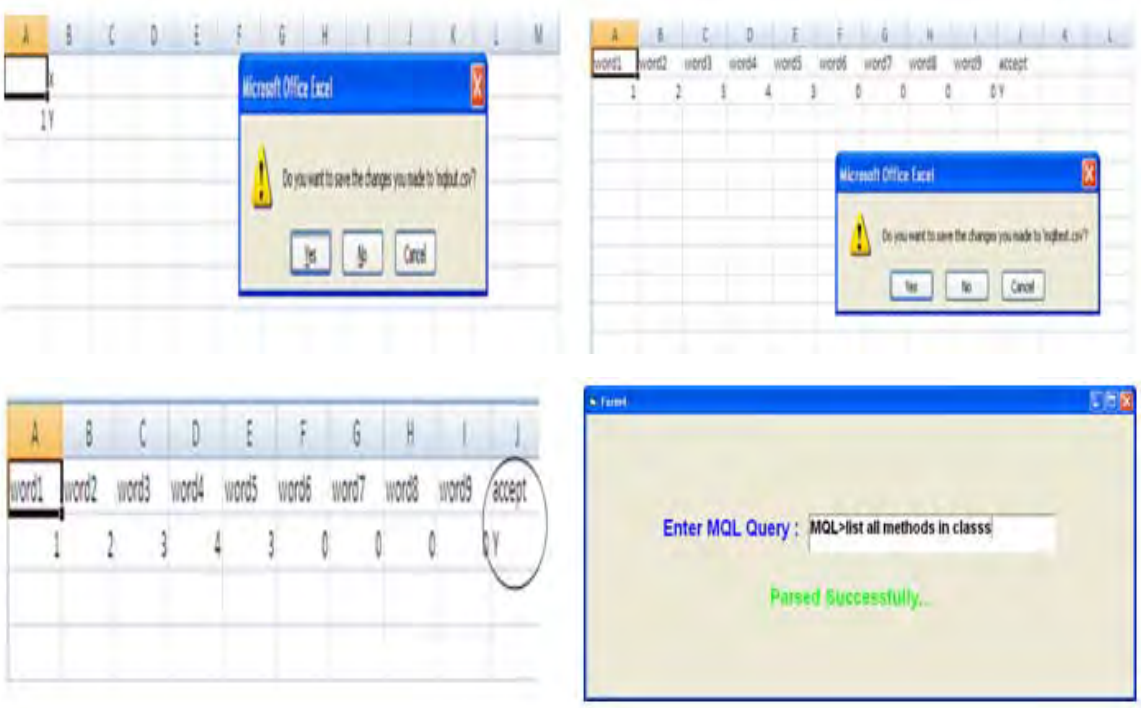


Figure 13-19. Parsing of MQL Sentence using Supervised Neural Network.

The execution of R script in R software is shown in Figure 20. where 2500 records of data set are employed for training purpose and 659 are employed for testing purpose. Figure 21 shows the corresponding confusion matrix generated by R Software. As seen in the output of confusion matrix there is 0% misclassification which can be attributed to large dataset employed for training purpose taking care of virtually all possibilities.

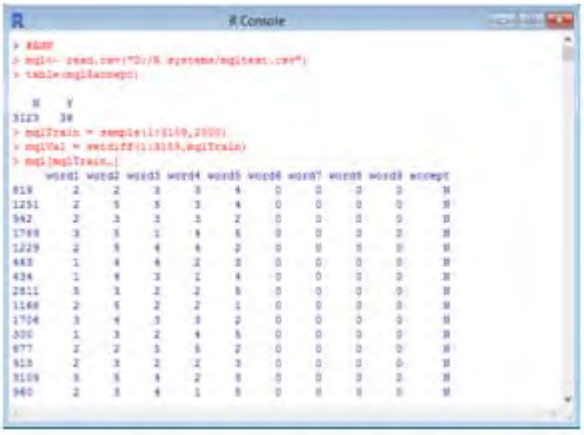


Figure 20. Execution of R Script in R Software.

Figure 21. Confusion Matrix Generated by R Software.

The model is also implemented in MatLab using nprtool toolkit. The network generated is shown in Figure 22.

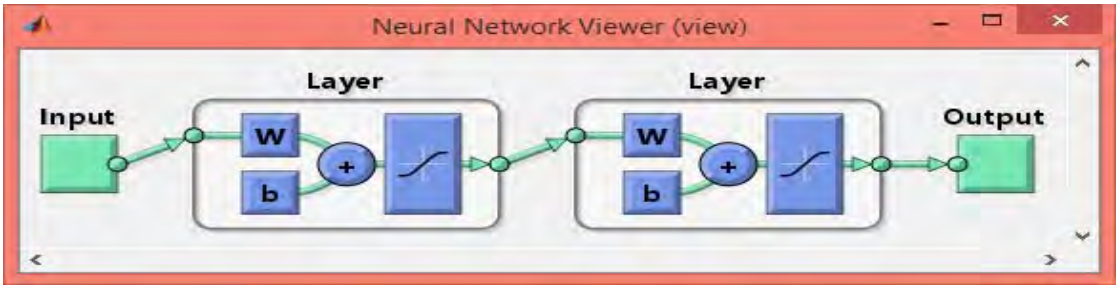


Figure 22. Neural Network Generated by MatLab.

As shown in Figure 23 , the best performance is achieved at epoch 53 with .01% Mean Square Error (MSE). The corresponding plots for Receiver Operating Characteristics (ROC) are shown in Figure 24.

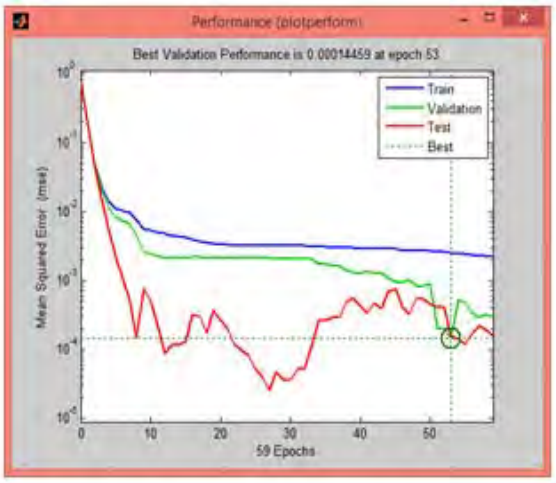


Figure 23. Neural Network Performance.

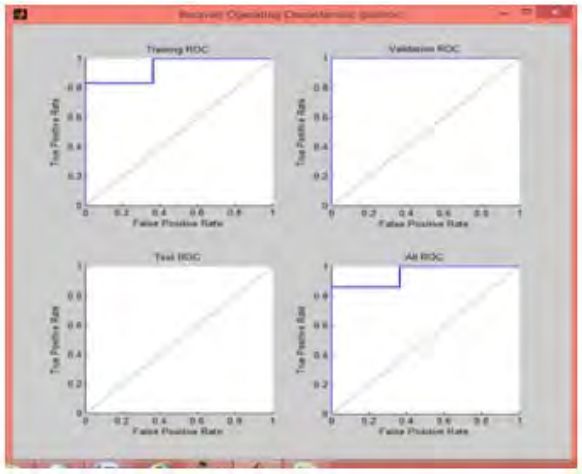


Figure 24. Receiver Operating Characteristics (ROC) of Neural Network.

The confusion matrix grid generated by nprtool and neural network are shown in Figure 25 and Figure 26, respectively.

Figure 25. Confusion Matrix Grid generated by nprtool.

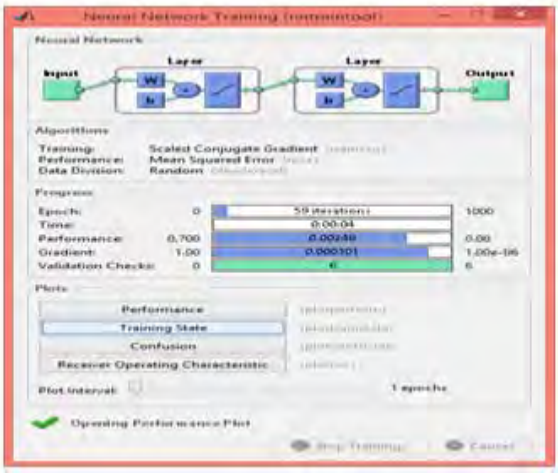


Figure 26. Neural Network Generated by nnttool of MatLab.

Figure 27 shows regression plots.

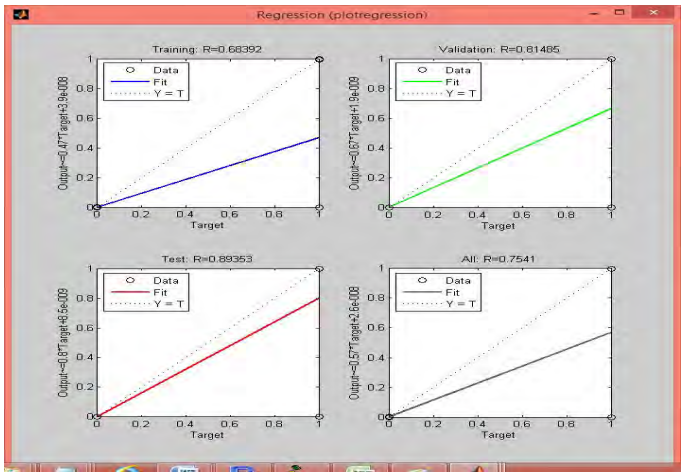


Figure 27. Regression Plots.

The M file generated dynamically at presentation tier is shown in Figure 28.

```
numHiddenNeurons = 20;
net = newpr(inputs,targets,numHiddenNeurons);
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train and Apply Network
[net,tr] = train(net,inputs,targets);
outputs = sim(net,inputs);

% Plot
plotperf(tr)
plotconfusion(targets,outputs)
```

Figure 28. Dynamically Generated M File.

Table VI and VII depict the R Software and MatLab results showing the accuracy of classification.

TABLE VI. RESULTS OF R SOFTWARE

Training Dataset			Validation Dataset			Entire Dataset		
Total	Misclassified	Accuracy	Total	Misclassified	Accuracy	Total	Misclassified	Accuracy
2500	0	100%	659	0	100%	3159	0	100%

TABLE VII. RESULTS OF MATLAB

Training Dataset			Validation Dataset			Testing Dataset			Entire Dataset		
Total	Misclassified	Accuracy	Total	Misclassified	Accuracy	Total	Misclassified	Accuracy	Total	Misclassified	Accuracy
2205	5	99.7%	473	0	100%	473	0	100%	3151	5	99.8%

Table VIII. depicts the relative comparison between classifications performed by R software and nprtool toolbox of MatLab.

TABLE VIII. COMPARISON OF CLASSIFICATION RESULTS BY R SOFTWARE AND MATLAB

	Size of Dataset	Misclassifications	% Accuracy
R software	3159	0	100%
MatLab	3151	5	99.8%

V. CONCLUSION AND SCOPE FOR FUTURE WORK

In this paper, the authors have proposed a generic model for parsing a sentence employing 3-tier architecture. Back propagation algorithm is employed for training a neural network. The model is implemented using R software and MatLab. The model is tested for parsing sentences of Manufacturing Query Language (MQL) designed by authors. Confusion matrix is generated for evaluating the accuracy of classification. The results obtained by both R software and MatLab are found to be comparable and close to the actual results. Both softwares generate results with more than 95% accuracy.

In the current work many of the strings that would be rejected by a neural processor which differ in a single parts-of-speech tag from their correct counter parts can be identified and neural processor can be further trained to detect one or more correct versions of such nearly correct sentences. For achieving this, probabilities of correctness can be associated with each sentence and cut-offs can be defined for classifying the sentence into one of three overlapping classes, correct, nearly correct and incorrect. A fuzzy neuro processor can be constructed wherein the output generated by a neural processor can be input to a fuzzy inference system which would classify the sentence into one of correct, nearly correct or incorrect linguistic variables.

REFERENCES

[1] Mr. Girish R. Naik, Dr. V.A.Raikar, Dr. Poornima G. Naik, Implementation of DFA Parser for Manufacturing Query Language Tokens, International Journal of Engineering Sciences and Research Technology, Vol 4, Issue 1, January 2015, p.no 370-382.

[2] J Shavlik, R Mooney, and G Towell. Symbolic and neural learning algorithms: An experimental comparison. Machine Learning, 1992.

[3] P J Wyard and C Nightingale. A single layer higher order neural net and its application to context free grammar recognition. Connection Science, 4, 1990.

[4] C Lyon. Guidelines for neural network design and their application to natural langage processing. Technical report, School of Information Sciences, University of Hertfordshire, September 1995.

[5] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Ma- chine Learning Research.

[6] C Lyon and R Dickerson. A fast partial parse of natural language sentences using a connectionist method. In 7th Conf. of European Chapter of Association of Computational Linguistics, 1995.

[7] H T Siegelmann, E D Sontag, and C Lee Giles. Complexity of language recognition by neural net- works. In 12th World Computer Congress on Algorithms,Software and Architecture. Elsevier, 1992.

[8] C Lyon. The representation of natural language to enable neural networks to detect syntactic struc- tures. PhD thesis, University of Hertfordshire, 1994.

[9] Ronan Collobert, Jason Weston, L´eon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. Journal of Machine Learning Research.

[10] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhut- dinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. CoRR, abs/1207.0580.

[11] A Fast and Accurate Dependency Parser using Neural Networks, Danqi Chen and Christopher D. Manning , Computer Science Department Stanford University Press.

[12] R. Reed, “Pruning algorithms-A survey,” IEEE Trans. Neural Networks, vol. 4, pp. 740-747, 1993.

[13] Simon Haykin, “Neural Networks- A Comprehensive Foundation”, Second Edition, Pearson Edition Asia, Third Indian Reprint, 2002.

[14] S. M. Kamruzzaman, Ahmed Ryadh Hasan, Pattern Classification using Simplified Neural Networks with Pruning Algorithm, ICTM 2005.

[15] Mr. Girish R. Naik, Dr. V.A.Raikar, Dr. Poornima G. Naik, Single Objective Criteria For Selection Of Manufacturing Method, International Journal of Computer Science and Engineering (IJCSE), Vol. 3, Issue 2, Mar 2014, 35-46.

[16] Mr. Girish R. Naik, Dr. V.A.Raikar, Dr. Poornima G. Naik, Single Objective Single Function Criteria for Selection of Manufacturing Method, International Journal of Emerging Technology and Advanced Engineering, Volume 4, Issue 2, February 2014, 182-190.

[17] Mr. Girish R. Naik, Dr. V.A.Raikar, Dr. Poornima G. Naik, Multi Objective Criteria for Selection of Manufacturing Method, International Journal of Advanced Research in Computer Science and Software EngineeringVolume 4, Issue 7, July 2014

Appendix A
Dynamic Generation of R File

```
Dim ans(1 To 100) As String
Dim colnames(1 To 10) As String
Dim colcnt As Integer
Dim formula As String
    colcnt = 1
    Set oXL = CreateObject("Excel.Application")
    oXL.Visible = True
    Set oWB = oXL.Workbooks.Open(filename)
    Set oSheet = oWB.ActiveSheet
    colnames(colcnt) = oSheet.Cells(1, 1).Value
    While (colnames(colcnt) <> "")
     colcnt = colcnt + 1
     colnames(colcnt) = oSheet.Cells(1, colcnt).Value
    Wend
    colcnt = colcnt - 1
     oWB.Save
    oWB.Close
     Set oRng = Nothing
    Set oSheet = Nothing
    Set oWB = Nothing
    Set oXL = Nothing
 formula = "formula <- "
 formula = formula & colnames(colcnt) & " ~ "
 For i = 1 To colcnt - 2
  formula = formula & colnames(i) & "+"
 Next
 formula = formula & colnames(colcnt - 1)

 'accept ~ word1+word2+word3+word4+word5+word6+word7+word8+word9"
Open "temp.R" For Output As #1
Print #1, "#ANN"
```

```
Print #1, "mqltrain<- read.csv(""" & filename1 & """)"
Print #1, "mqltest<- read.csv(""" & filename2 & """)"
Print #1, "mqltrain1 = sample(1:" & Text1.Text & "," & Text1.Text & ")"
Print #1, "library (nnet)"
Print #1, formula
Print #1, "mqlANN = nnet(formula, data=mqltrain, subset=mqltrain1, size=10, rang=0.2, decay=0.0005,
maxit=200)"
'#table(mqltrain$accept, predict(mqlANN, mqltrain1, type = "class"))
'#table(mqltest$accept, predict(mqlANN, mqltest, type = "class"))
Print #1, "predict(mqlANN, mqltrain, type=""class"")"
Print #1, "mqlpred = predict(mqlANN, mqltest, type=""class"")"
pname = App.Path & "\mqlout.csv"
pname = Replace(pname, "\", "/")
Print #1, "write.csv(mqlpred,""" & pname & """)"
Close #1
MsgBox "R file created successfully..."

Open "run.bat" For Output As #1
Print #1, "set path=%path%;C:\Program Files\R\R-3.1.2\bin"
Print #1, "rscript temp.r"
Print #1, "pause"
Close #1
Shell (App.Path & "\run.bat")

    Set oXL = CreateObject("Excel.Application")
    oXL.Visible = True
Set oWB = oXL.Workbooks.Open(App.Path & "\mqlout.csv")
    Set oSheet = oWB.ActiveSheet
    For i = 2 To Val(Text2.Text) + 1
     ans(i - 1) = oSheet.Cells(i, 2).Value
    Next
     oWB.Save
    oWB.Close
     Set oRng = Nothing
    Set oSheet = Nothing
    Set oWB = Nothing
    Set oXL = Nothing

    Set oXL = CreateObject("Excel.Application")
    oXL.Visible = True
    Set oWB = oXL.Workbooks.Open(filename)
    Set oSheet = oWB.ActiveSheet
    For i = 2 To Val(Text2.Text) + 1
    oSheet.Cells(i, colcnt) = ans(i - 1)
    Next
    oWB.Save
    oWB.Close
     Set oRng = Nothing
    Set oSheet = Nothing
    Set oWB = Nothing
    Set oXL = Nothing
```