International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

A Hopfield Neural Network Based Building Agent for Detection and Correction of Programming Errors

Dr. Buthainah F. AL-Dulaimi[1], Hameed Taha Khaleel[2]

[1, 2]University of Information Technology, Communications Institute for Postgraduate Studies

Abstract: ***A syntax error is an error in the sequence of characters or tokens that in a particular programming language may be detected during program execution or compilation. Detection and correction of syntax errors present the difficulties that face all beginner programmers especially the arrangement sequence of the tokens in the program code. In this paper, the proposed system is developing an agent to perform the detection and correction syntax errors according to Hopfield technique through processing each token individually usingJava programming language.Our technique applied to (250-line code in Java program language) as the case study. It can select the specific keyword (100 sentence that starts with the keyword "For"). Then it can be (detection 60 and correct 53 from 73-syntax error).***

Keywords: Correction syntax error, Hopfield Neural network, Agent, Simple reflex agent, Java

1. Introduction

Programming can generally be stressful in particular if a person is a beginner and not influent with the compiler error messages when he/she starts writing the program. One of the important program languages and widespread is Java program where the compiler's error message can be Syntax Errors, Semantic Errors and Logic Errors [1]. This research focuseson syntax errors, to highlight the most common mistakes faced by the beginner [2], some of them are listed below.

- The operator (=) instead of comparison operator (==) and Vice versa.
- The wrong use of brackets {}, [ ], ( )
- Put a semicolon after “for” statement, “while” statement, or “if” statements.
- The order of token or characters in the structure of “for” loop statement.

The syntax error is pointing to the mistakes in order of token inside structure statement and punctuation. Frequently, messages syntactical error is not necessary to lead beginners to correct the error. Generally, most research converts the incorrect program to correct by using replacing, adding and/or deleting one or more symbols [3].

This research, deals with error correction for sequence tokens of structure statement by using machine learning techniques to detect and correct the error in the program code.The research is organized as follows, in the section 2 reviews of the related work. Next, in Section 3 and 4 present the general description of the software agent and Hopfield neural network respectively. The proposed system illustrates and described in section 5. In section 6 the results of the implemented system. Section 7 Discussion andsection 8 Conclusion.

2. Related Work

Prof. Khushali Deulkar in the research "A Novel Approach to Error Detection and Correction of C Programs Using Machine Learning and Data Mining"[4],summarizes the following" There has always been a struggle for programmers to identify the errors while executing a program- be it asyntactical or logical error. This struggle has led to a research in the identification of syntactical and logical errors. This paper makes an attempt to survey those research works which can be used to identify errors as well as proposes a new model based on machine learning and data mining which can detect logical and syntactical errors by correcting them or providing suggestions. The proposed work is based on the use of hash tags to identify each correct program uniquely and this, in turn, can be compared with the logically incorrect program in order to identify errors".

A.D. Shah "Syntactic Error Correction System"[5], "The paper represents the various types of syntactic errors, their detection and further their correction system. This method of detection and correction of the syntactic errors is inspired by the need to correct them for a correct grammatical sentence formation. The best systems, however, results in syntactic errors due to of the lack of linguistic knowledge. An attempt to optimize the task of detecting and correcting the syntactic errors is reflected in this paper".

Xavier Hinaut "Recurrent Neural Network for Syntax Learning with Flexible Representations "[6],"We present a Recurrent Neural Network (RNN), namely an Echo State Network (ESN), that performs sentence comprehension and can be used for Human-Robot Interaction (HRI). The RNN is trained to map sentence structures to meanings (e.g. predicates). We have previously shown that this ESN is able to generalize to unknown sentence structures in English and French. The meaning representations it can learn to produce are flexible: it enables one to use any kind of “series of slots” (or more generally a vector representation) and is not limited to predicates. Moreover, preliminary work has shown that the model could be trained fully incrementally. Thus, it enables the exploration of language acquisition in a developmental approach. Furthermore, an “inverse” version of the model has been also studied, which enables to produce sentence structure from meaning representations. Therefore, if these two models are combined in the same agent, one can investigate language (and in particular syntax) emergence through agent-based simulations. This model has been encapsulated in a ROS module which enables one to use it in a cognitive robotic architecture, or in a distributed agent simulation. ".

The aim of the research Prof. Khushali Deulkaris to specify the errors during executing the program (syntactical/logical error). This research proposes a model based on machine learning and data mining to detect and correcting or providing suggestions logical and syntactical errors. While in the research of A.D. Shah present method to optimize detecting/correcting of the syntactic errors using tolerant and strict grammar proceeds.The Xavier Hinaut, adopted a Recurrent Neural Networks technique for syntax learning by trained on correct sentence structures and put to meanings.

In our research builds an agent to detect and correct for sequence tokens in the program code using the machine learning techniques based on Hopfield Neural Network (HNN).

3. The Software Agent

There are different definitions for the software agents, one of them as programs that participate in negotiating and organize the transfer of information and other definition it contains properties of autonomy, social adeptness, reactivity, and proactivity and can be a hardware and/or software-based computer system [7]. While (S. Russell and P. Norvig 2009[8], defined the agent as a term indicates to an entity that can be perceiving its environment from the sensors and acts or reacts upon the environment of effectors. Agents are classified into four types:

- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents

In this research, the simple reflex agent is used because of the need for an agent that can perform one action (selective statement) depending on the specific condition (matching specific keyword). The architecture of simple reflex agent is illustrated in Figure (1).

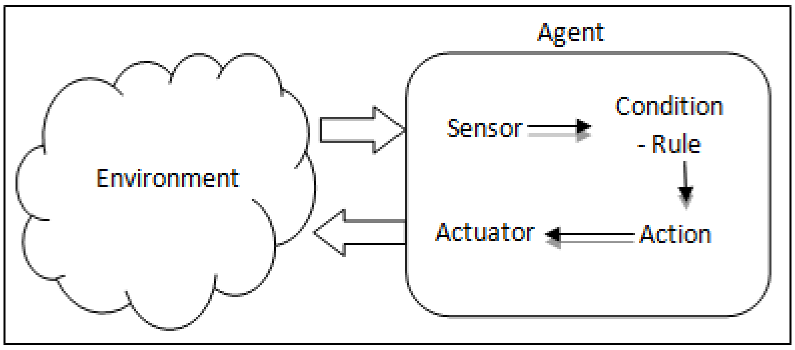

Figure 1: Simple Reflex Agent

4. Hopfield Neural Network (HNN)

The Hopfield network is one types of the recurrent neural network, contains a set of fully interconnected neurons that have a loop network of feedback from its output to its input while there is no self-feedback [9].These loops help greatly in the learning process of the network, as shown in Figure (2). The patterns of the learning process are stored in the network using the learning algorithm of Hopfield to become a stable state of patterns. The matrix synaptic weights can be calculated as in Equation 1.

$$w = \left(\sum_{i=1}^{m} X_m X_m^T\right) - Im \quad \text{........ (1)}$$

W: synaptic weights between neurons.
m: number of the pattern.
Xm: n-element of the binary vector.
I: identity matrix n*n.
M: number of the pattern.

In the test phase, inputting of the testing samples, the network starts to execute from the preliminary stages and gradually converging to a stable state of the network [10]. Confirmation that the Hopfield network is capable of recalling all fundamental memories is needed.

$$Y_{mj} = sign\left(\left(\sum_{i=1}^{m} X_{mj}\ W_{ij}\right)\right) \quad \text{........ (2)}$$

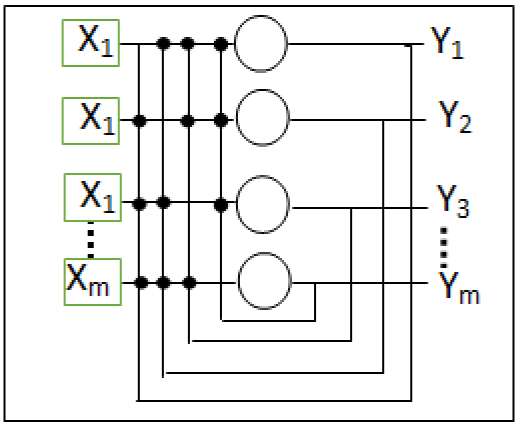

Figure 2: Hopfield Neural Network

In the proposed system, Hopfield algorithm is used to train the agent on the keyword syntax. The components of the highlighted token and their number through the keyword. The dimensions and elements of the training matrix are determined. While the test stage to detection and correction using the matrix token that results from processing algorithm (1). The contribution here is using the number of iteration in Hopfield algorithm (2) to reorder parts of the token.

5. The Proposed System

The proposed system is developed to detect and correct the syntax statement in java program as shown in Figure (3). In this system, an agent is designed to perform the required operations according to the following steps:

1) Preprocessing data which as illustrated in the algorithm (1).
   - Get the data from a notepad scripts.
   - Reading and writing the statement from notepad in model once at a time.
   - Select the statement that starts with a specific keyword.
   - Use the resulting data to be processed by converting it to binary number {1,-1} as shown in Figure (4).
2) Detection & correction syntax by apply Hopfield & Reorder Matrix which is illustrated in the algorithm (2).

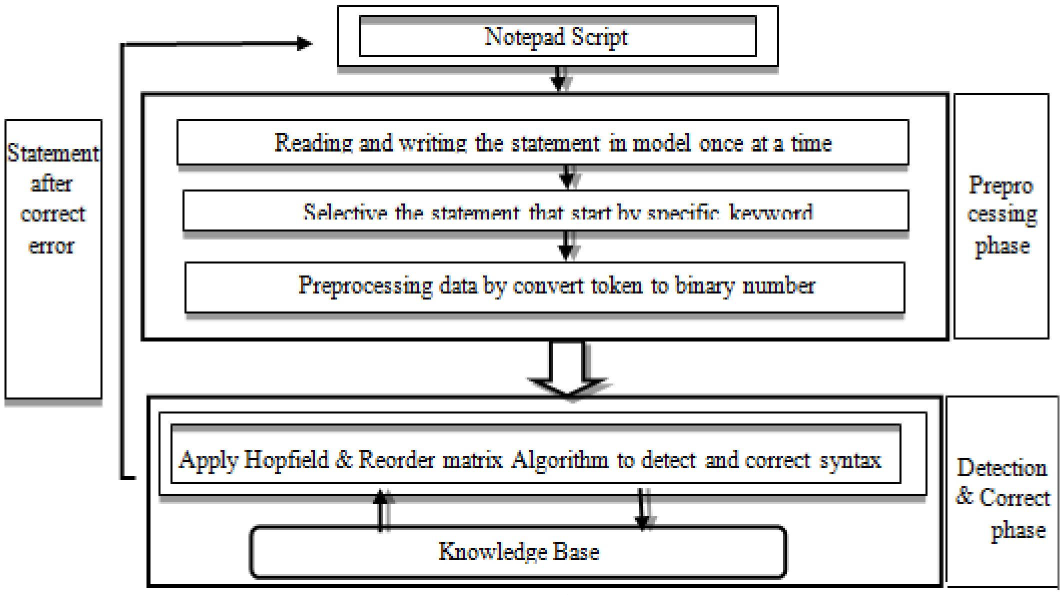
Notepad Script
Reading and writing the statement in model once at a time
Selective the statement that start by specific keyword
Preprocessing data by convert token to binary number
Prepro cessing phase
Statement after correct error
Apply Hopfield & Reorder matrix Algorithm to detect and correct syntax
Knowledge Base
Detection & Correct phase

Figure 3: The block diagram of the proposed system

The proposed system consists of two phases: the First phase, starts from reading script code line from a case study file once at a time and write in the model system every two seconds to simulate the programmer work to be prepared as a string. The next in this phase is the selective script code line that starts with a keyword. After that, begin to match the start of the script code line with the keyword and the first symbol of its syntax that is stored in the database. Then remove space between the words and replace punctuation marks {'(', ')', ';'} by spaces of script code line. The token is the string between the two semicolons or between the arches and semicolon. Each token separated into three pieces by inserting a space between each symbol and a text string and the numeric string to result phrase in string format has to space between every token and between every part of the token. In the last block of the first phase, each part of the token will be converted into a sequence of {1, -1} depending on its position from the syntax and sequentially. Where are assigns {1,-1,-1} to the text string portion of the token, While assigns {1,-1,-1} to the symbols string part of the token, and the numeric string of the token assigns it {-1,-1,1}. This process is repeated for each token of the code sentence. The number of repetition is different from one sentence to another depending on the type of keyword that used in the code sentence. At last of pre-processing phase, the algorithm returns matrix (3*3) diminution contains a sequence of {1,-1}. These steps illustrated in the algorithm (1) preprocessingtokenas shown in Figure (4).

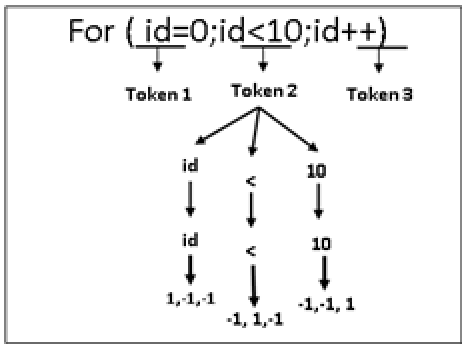

Figure 4: Convert token

Algorithm (1) preprocessing token

Input: - case study
Output: - sequence of binary number
<u>Strategy:-</u>
Reading code line from case study file once at time and write in model system once every two seconds. Then separate statement in to token and replace punctuation marks {'(', ')', ';'} by spaces. At last convert each token to binary number

Step One line by line
```
1.  Onest = read & sort line once at time  //variable stored script code line
2.  Write value of Onset to model system
3.  If EndOfFile = true goto step 5 // variable index pointing to end of file
4.  After two seconds go to step 1
```
Step Two Selective
```
1-  St = script code line from file  //St is variable stored script code line
2-  Len = length of St  //Len is variable store length of script code line
3-  Kword = keyword with first symbol of its syntax  For i = 1 to Len
4-      If St(i) <> space go to step (7)    // if string don't have space
5-      St(i)      replace {'(', ')', ';'} with "  "
6-      Str      add value of St(i)
7-  Next i
8-  P = the position of Kword  //p is variable stored position Kword in St
9-  If P = 0  go to step (13)    // if line code started by keyword
10- if St access to end of file go to step (16)
11- if St don't access to end of file go to step (14)
12- return Str                   // Str is variable storage output
13- get next line & go to step (1)
14- End.
```
Step Three Convert to binary number
```
1-  Divide the token into three parts
2-  Len = length Syntax keyword // variable stored length Syntax from database
3-  V = vector [1 .. len]
4-  Convert each part of token to sequence of {1,-1} & stored in V
5-  End.
```

Algorithm (2) Hopfield & Reorder Matrix

<u>Input: -</u>
The vector Contain 1 & -1 from algorithm (1) preprocessing.
<u>Output: -</u>
Restore vector depend on fundamental memory
<u>Strategy:-</u>
Calculated synaptic weight matrix from neuron i to neuron j to sort M fundamental memories. Then, test the input vector and matches it with M to reorder matrix.
<u>Steps of Algorithm</u>
<u>Training stage</u>

1. Store a set of M fundamental memories, Y1;Y2; . . . ;YM. and calculated synaptic weight from neuron i to neuron j is as

$$w_{ij} = \begin{cases} \sum_{m=1}^{M} y_{m,i}\, y_{m,j}, & i \neq j \\ 0, & i = j \end{cases},$$

<u>Test stage</u>

2. the network recall fundamental memory Y_m when presented an input.

$$y_{m,i} = sign\left(\sum_{j=1}^{n} w_{ij}\, x_{m,j} - \theta_i \right)$$

3. Store number of row which matches with the fundamental memory.
4. Reorder matrix if Y_m matrix <> training matrix.
5. End.

In the Second phase, applying Hopfield &reorder matrix algorithm that illustrated in the algorithm (2), which consists of two basics stages, the training stage, and the testing stage. In the training stage, the algorithm trains on the token sequence for each the keyword syntax. The training data is a matrix that contains the correct sequence of the locations for each token part and is considered as input to the training phase. Then, the steps of the Hopfield algorithm are applied to the weight matrix which is produced. These are the rules of the syntax of the keyword. In the testing stage, that includes detection and correction for the syntax of the sentence that was prepared in the first stage by the algorithm (1). Each token of the matrix is testing through multiplying it by the weight matrix that was resulted from the training process. The resulting matrix of the multiplication process is checking that if it is similar to this training matrix indicates that the token for the sentence is correct otherwise if opposite, the token contains an error. Then the reorder matrix procedure is used to be applied for the token that contains error. Through reordering the rows of the matrix considering that each part of the token is a row in the token matrix. The process repeats until the last token within the matrix of tokens and then stored in the vector. This vector stores all the tokens after the correction process is finished. After that, the error sentence is replaced by the correct sentence which is stored in the vector. To implement the proposed system, Simple Reflex Agents is used. The agent used the Hopfield Neural Network techniques to perform the required operation of the system.

6. Results

To view the result of the proposed system, the case study contents are more than 250 line code in Java program language with syntax errors [11]. In particular, testing the "For keyword" syntax as the example for the proposed system. On the assumption, the program has 100 sentences that start with the "For as the keyword". The algorithm (1) preprocessing token will selective the whole sentence that starts with the "For as the keyword" successfully. While algorithm (2) Hopfield &Reorder Matrix, that given result illustrated in the following table (1) and table (2). Table (1) views different syntax example of the "For as the keyword". While table (2) views a number of the sentences, which can be corrected by the system.

Table 1: For syntax

Syntax keyword "for"	**Example**
For (ID SM NU|ID; ID SM NU|ID; IN|DE ID IN|DE)	for (n=10; n>0; n--)
For (type data ID SM NU|ID; ID SM NU|ID; IN|DE ID IN|DE)	for (int n=10; n>0; n--)
For (ID SM NU|ID; ID SM ID; IN|DE ID IN|DE)	for (a=1; a<b; a++)
For (ID1 SM NU|ID , ID2 SM NU|ID ; ID1 SM NU; IN|DE ID IN|DE)	for (a=1,b=4; a<b; a++,b--)
For (SM ID)	for (!done)

ID = Identify
SM = {=, <, >,!, <=, >=, !=}
NU = Number
IN|DE = Update (Increase | Decrease)

Table 2: Sentences corrected by the proposed system

Syntax keyword "for"	Number of sentences	sentence error	Detection sentence error	Correct sentence error	Incorrect sentence		
					Token 1	Token 2	Token 3
For (ID SM NU|ID; ID SM NU|ID; IN|DE ID IN|DE) Or For (type data ID SM NU|ID; ID SM NU|ID; IN|DE ID IN|DE)	50	40	40	40	0	0	0
For (ID SM NU|ID; ID SM ID; IN|DE ID IN|DE)	20	18	10	10	0	8	0
For (ID1 SM NU|ID , ID2 SM NU|ID ; ID1 SM NU; IN|DE ID IN|DE)	15	10	10	3	4	0	3
For (SM ID)	15	5	0	0	5	5	5
TOTEL	100	73	60	53	10	13	10

Table (2), shows the proposed system corrected the syntax for another version of the same keyword. Actually, the differences in the number of tokens in the syntax cause the difference of efficiencyfor the correction process in the proposed system. For example, In the sentences "for (!done)" It consists of a single token that contains two sections logical comparison tag"!" and identify " done". In this case, the system cannot detect or correct the errorbecause the precedence of the text string is the preferred from the symbolic string. While in the sentence "for (n=10; n>0; n--)", it consists of three tokens, each containing three parts. Here, the proposed system will be able to detect the syntax error and then correct based on the position for each part of the token.

7. Discussion

The importance of our research is to help the students that trying to learn Java program language by designing a software agent capable of learning syntax. It can detect and correct syntax errors without trouble to understand the error messages from the compiler. Our system supports the use of neural networks to contribute to the detection and correction of error. This technique has the ability to remember knowledge from the learning stage and then address the problem within the limits of learning information. As can be seen, in the new methods of compiler design depend on machine learning techniques. As shown above, in the results section, based on the concept of "divide and conquer" to the design of our system. The agent is trained to break the syntax based on the symbol (;) and check the arrangement sequence of symbols for each part. We presented syntax of the statement "FOR" as an example because it consists of more than a syntax formula. In Table (2), the first example consists of two different syntaxes, which can be detected and corrected by so that all the arrangement sequence error of the tokens in the syntax are eliminated. Because of each part consists of three types of token {Identify, Logical comparison marks, numbers}, in the remaining examples, the detection and correction results fluctuate depending on the parts of the syntax.

8. Conclusion

The research is devoted to correct the arrangement of a sequence of the keyword syntax in the Java programming language by processing each token individually. our system designed specifically to help the beginner programmers. Through, detection and correction syntax error according to Hopfield technique, regarding arrangement sequence of the tokens in the program code. We used the Java program language in building agent and as a case study. Because of, It is one of important language and most popular that deals based on object-oriented, class-based. The much of its syntax derives from C, C++ language. Ultimately, the beginner programmers in Java program language understand how to correct the some of the syntax error. Without wasting time to understanding the error messages of the compiler.

References

[1] TIOBE Software BV, “TIOBE Index | TIOBE - The Software Quality Company,” *Tiobe*, 2016. [Online]. Available: http://www.tiobe.com/tiobe-index/.
[2] P. Denny, A. Luxton-Reilly, and E. Tempero, “All syntax errors are not equal,” *Proc. 17th ACM Annu. Conf. Innov. Technol. Comput. Sci. Educ.*, p. 75, 2012.
[3] M. Hristova, A. Misra, M. Rutter, and R. Mercuri, “Identifying and Correcting Java Programming Errors for Introductory Computer Science Students,” *ACM SIGCSE Bull.*, vol. 35, no. 1, pp. 153–156, 2003.
[4] K. Deulkar, J. Kapoor, P. Gaud, and H. Gala, “A Novel Approach to Error Detection and Correction of C Programs Using Machine Learning and Data Mining,” *Int. J. Cybern. Informatics*, vol. 5, no. 2, pp. 31–39, 2016.
[5] A. D. Shah, A. D. Shah, and L. D. mello, “Syntactic Error Correction System,” *Int. J. Eng. Comput. Sci.*, vol. 4, no. 12, pp. 15202–15207, 2016.
[6] X. Hinaut, X. Hinaut, R. Neural, and S. Learning, “Recurrent Neural Network for Syntax Learning with Flexible Representations To cite this version : Recurrent Neural Network for Syntax Learning with Flexible Representations,” *IEEE ICDL-EPIROB Work. Lang. Learn.*, 2016.
[7] I. Rudowsky, “Intelligent agents,” *Commun. Assoc. Inf. Syst.*, vol. 14, no. August, pp. 275–290, 2004.
[8] S. Russell and P. Norvig, *Artifical Intelligence: A Modern Approach (Third Edition)*, 3rd Editio. PRENTICE HALL SERIES IN ARTIFICIAL INTELLIGENCE, 2009.
[9] A. H. N. Network, “A Modified Hopfield Neural Network for Solving TSP Problem,” pp. 1775–1780, 2016.
[10] M. Negnevitsky, *Artificial Intelligence AGuide to Intelligent Systems*, Second Edi. England: Pearson Education Limited, 2010.
[11] R. Sedgewick, K. Wayne, J. Holcomb, C. Melville, and G. Entremont, *Introduction to Programming in Java An Interdisciplinary Approach*. Greg Tobin, 2007.