American Journal of Intelligent Systems 2017, 7(3): 81-85
DOI: 10.5923/j.ajis.20170703.09

Decipher C –Neural Networks based Software for Beginners

Aatash R. Sengupta*, Akshatha Shenoy, Appu Sreenivasan, Krupa H. Ghetia, Mr Prathviraj N.

Department of Computer Science and Engineering, St Joseph Engineering College, Vamanjoor, India

Abstract Computers and technology now permeate nearly every aspect of our lives, but few if any lay people can understand how they work at all. While this does not actually matter to the users, other programmers and software specialists would want to see how a program works to try and increase their own knowledge. For experts, reading the documentation and then the programs is a simple matter that gives near total clarity on how a system works. But, when a fledgling, naive programming student attempts to do the same, they are unlikely to understand most of the program designed by an expert. Even mid-level and simplistic programs are often beyond the scope of beginners in programming to understand. Within the sphere of existing technologies, the naïve programmer will need to find an expert to explain the program to them, as the current teaching resources use only a standard set of programs that cover the concepts, but fail to extend their scope beyond that. Decipher C is created to take a program written in C from a user and then break down the program to explain each instruction to the user. By doing this, Decipher C will allow the user to input any complex program and have it explained at the required level, ensuring maximum understanding of every concept involved. By also considering the ability to identify any areas of difficulty that the user is facing and provide an appropriate learning tool for naïve programmers to use. The target is to create a robust system that can explain any C program regardless of the features it employs.

Keywords Machine Learning Algorithms, Natural Language Processors, Neural Networks

1. Introduction

The world today is in a digital age in which technology plays a role in much of what we do every day. Technology is used at the office, at school and at home. Digital devices are all around us. However, in many cases, what technology can do for the users is barely researched in depth. Therefore the end users pretty much accept the fundamental role technology plays in day to day lives of a user and mostly use the basic functionality for each of the users' digital devices. While most people will never get under the hood to try and change the code of an appliance or device a user use, by learning the fundamentals of creating the software code that runs our devices, a person will gain a greater understanding of how the users devices work, and would be more inclined to go beyond the users devices' basic functionality. A coding class would also help a naïve programmer to gain a greater understanding of how technology is designed and how software serves as the medium for triggering all of a device's capabilities. This type of knowledge could be important in a future working environment where the users are called upon to use technology as part of their overall job. This software helps beginners to learn C coding from already existing examples and also helps experts to understand their programs work. To understand a C program is very difficult especially for a beginner. Although explanations for most standard programs are available online, understanding a random program is sometimes difficult. The aim of this software is to help a person understand the code in a better way by providing explanations of the program as a whole as well as part by part. This software will provide web resources which will help the user to understand any program that is fed to it. Decipher C is a software that will help a person understand any C program. It is a self-learning software which will help a person to understand and write better C programs. It can be used by students to learn C programming as well as programmers to better understand their programs.

* Corresponding author:
aatash.sengupta@gmail.com (Aatash R. Sengupta)
Published online at http://journal.sapub.org/ajis

2. Related Work

Other, existing projects were referred to in an attempt to gauge the current situation in solving this problem. Primarily on the front of neural networks, an immediate problem was found. In this sample by Matthew Cochran [1], it is seen that existing neural networks are independently trained and then deployed for usage. While the accuracy of the given algorithm is high, the low versatility made it insufficient for our usage.

While researching for the tokenization component, there were few projects that worked with tokenizers alone as they are now largely a small component in larger systems. Others, like the Fast Lexical Analyzer [2] are designed as shell programs to run on UNIX based systems. Additionally, Decipher C requires non-standard tokens to be used as need, as well as the tokens to be in some form of organized structure, neither of which are provided by existing tokenizers.

Due to the similarities in operations, Natural Language Processing projects, like OpeNER [3], were investigated. Despite the advancements, Decipher C doesn't work with natural language processing and so could not utilize many of the tools available.

3. System Architecture

The entire system of Decipher C is divided into three main segments, i.e. user interaction, server page, and processing. Each segment can run independently of the others, allowing for a physical separation of components. They simply communicate on as required.

The User Interaction part is handled through a web page that serves as the sole user interface of the system. It formats all the data that is input or output to ensure the best experience for the user.

The second phase is the server, which is the primary control module of the system. Besides funneling the data between the other segments it also controls how many explanations to output, as well as which ones to use.

The final part, the Processing segment, forms the major functional module of Decipher C. Built as a module that can be run as an executable file, this segment can actually be utilized with varying user interfaces to implement and use the core functionality of Decipher C in a multitude of ways.

A. Web Page (User Interaction)

The web page is the sole interaction area for the user to utilize Decipher C. Using basic forms and frames, the page has three parts:

i. The program section, a form that gives the user a text box into which users can type or paste the program that is required to be explained.
ii. The Explanation section that is generated on submission of the program and will display a list of explanations for the given program, with hyperlinks to explain in more detail.
iii. The Detail section that can show the portion of the code that is currently being explained.

The Web page does not handle or process any data, it just receives the set of explanations and sends the program. It also sends to the server page which link is clicked.

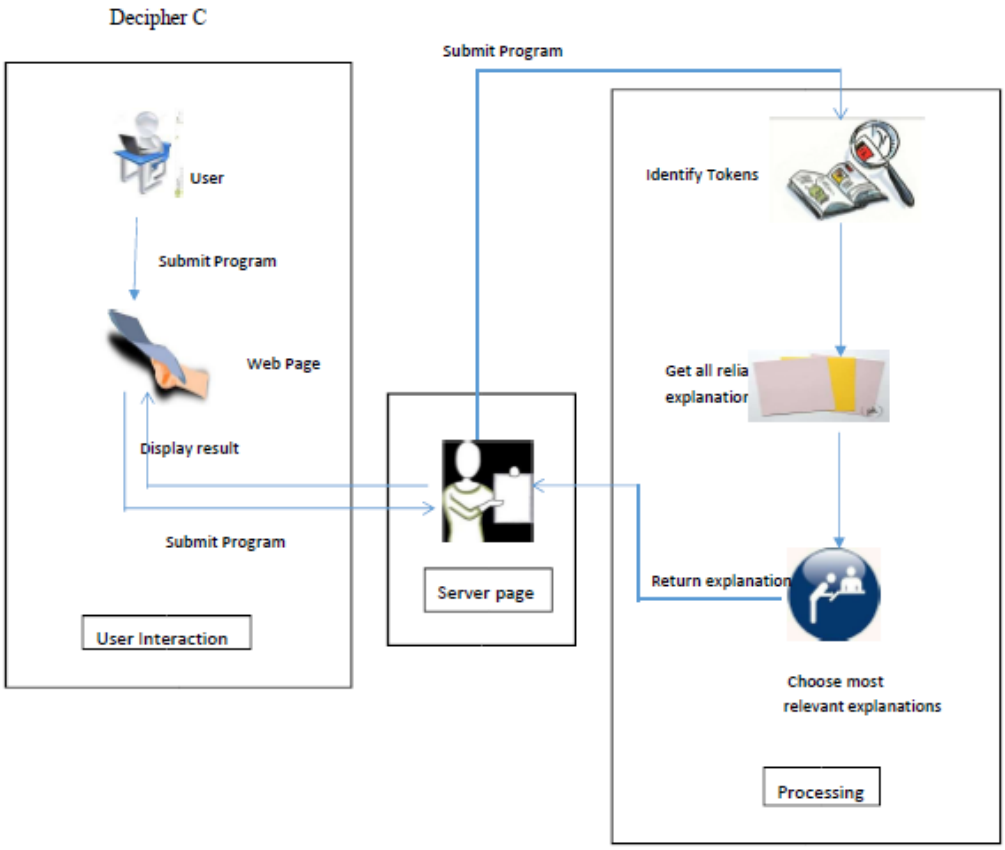

Figure 3.1. The system Architecture

B. Server scripts

The Active Server Page has a fairly simple functionality, but it works as the main program of the entire system. The ASP will receive the program from the web page, and pass that as an argument when it calls the Processing segment. It will then analyze the output of that segment, and display only select descriptions to pass on to the web page for display.

The ASP also tracks what links are clicked by the user, and uses that to adjust the rankings. Thus, Decipher C will become more user-friendly and accurate based on the number of users and users own preferences for explanations.

4. Decipher C core

The Processing segment of Decipher C is built to run as a simple executable file, despite being the core of the system. It is here that the programs submitted are broken up into tokens which are analyzed to find explanations for them and those subsequent explanations are passed back to the ASP. Due to this modular design, Decipher C can be implemented in a number of ways, but a web page was selected to ensure the furthest reach and easiest access for users.

The processing of the program is done largely as two parts;

A. Tokenizer:

The tokenization of the program is the first thing done once the Processing segment is called. By using a set of regular expressions, some used by standard compilers like int|float|char|double|voidfor data types and some were designed specifically to be used by Decipher C to provide explanations.

The tokenizer goes a step further as well, by placing all the tokens into a tree, creating a structure akin to a Parse tree for the program. However, this is not a real Parse tree that can be used to compile the program due to the presence of the customized tokens that no standard compiler will recognize.

Once the tree is ready, the tokenizer hands over to the next part.

Pseudocode:

```
class Lexeme
{
public string[] lexemes;
public string token;
public string pattern;
public Match m;
public Lexeme()
      {
pattern = String.Empty;
pattern=String.Concat(pattern,lexemes[0]);

for (int i=1; i<lexemes.Length; i++)
            {
pattern = String.Concat(pattern,"|",lexemes[i]);
            }
      }
}

class Program
{
static Lexeme[] list;
staticboolEndSegment(string s)
    {
          // Find the end of a code segment
    }
staticboolStartSegment(string s)
    {
          // Mark a start of a code segment
    }

static Node Match(string code)
    {
        Node curr=new Node();
foreach (Lexeme l in list)
          {
 //get the regular expression to compare
reg = new Regex(l.pattern);
                //check if the lexemes are in the string
if (reg.IsMatch(code))
                {
l.m = reg.Match(code);

//if token is matched, update the node
nextCode = code.Substring(curr.matched.Length);

if (StartSegment(nextCode))
                          //Add a child
else if (EndSegment(nextCode))
                          //End the set of children
else
                          //Add a sibling
                }
          }
returncurr;
    }
static void Tokeniser(string[] args)
    {
        //create the total parse tree
    }
}
```

B. Neural Network:

Linking the token to explanations, which is done using a set up very similar to a neural network. Though named a Neural Network, this section is not actually a standard neural network. A regular neural network works with binary values 0 and 1 (and everything in between in the case of a sigmoid neuron). Each neuron in the network will need this same input, and produce the same binary output. Further, most networks will first be trained to produce the correct output, and is then run and used as required This was insufficient for usage in Decipher C. We needed the network to both learn as it is used, and take a non-binary input. Thus, the first change

came from the fact that the nodes in the network take an input as a node from the parse tree, and not a numerical value. A consequence of this change is that the neurons are each assigned a node which they will need to then analyze and provide an explanation for. Further, the output of these neurons will not be binary, but instead a series of explanations that can be used for their assigned node. Finally the output neuron simply sorts the entire group of explanations and the network will give back just a set of the ten most relevant explanations for the program.

Pseudocode:

```
public class NeuralNet
{
publicNeuralNet()
      {
          //run the entire neural network
      }
void Perceptron(Node n, out Explanation[] e)
      {
           //find overall explanations
      }
void Neuron(Node n, out Explanation[] e)
      {
            //find specfic explanations by checking
subtree
      }
void Output(Explanation[,] expl_in, out Explanation[]
expl_out)
      {
          //collate explanations
      }

void Execute(Node n, out Explanation[] e)
      {
Perceptron(n, node_expl_main[0]);
          Node next = n.FirstChild;

while (next!=null)
          {
sub_expl[count] = new Explanation[10];
Neuron(next, sub_expl[count++]);
          }
Output(sub_expl, out node_expl_sub[]);
      }
}
```

The explanations themselves are stored in a database, along with the tokens that can be used and the connections between the two. While this database is not a direct part of the Processing section, it is a very important one. The web Resources will be the actual collection of links to each and every explanation for the program that is given as input to the website. Thus each and every link will lead the user to the required explanation for the various sections of the program. The web resources will be updated periodically in such a way that the most useful link comes on the top.

This is done with the help of a counter that increases the value of a particular link each time it is clicked, thereby making the most useful link come out on top of the list.

This will be especially useful when a user wants the best explanation for particular sections of a program as fast as possible.

Therefore the more Decipher C is used, the better it gets at bringing the most relevant information to the users.

5. Experimental Set and Results

Table 5.1. Test Case for Front End

TEST CASE	EXPECTED RESULT	TEST RESULT
When program is entered correctly	A set of urls with one line explanation from each URL.	No explanations currently available

Table 5.2. Test Case for Back End

SL.NO	TEST CASE	EXPECTED RESULT	TEST RESULT

1.	#include<stdio.h>
	void main(void)
	{
	printf("Hello World");
	}
		80,79,78,77,76,75	PASS

2.	#include<stdio.h>
	printf("Hello World");
	int y=x+3;
		NULL	PASS

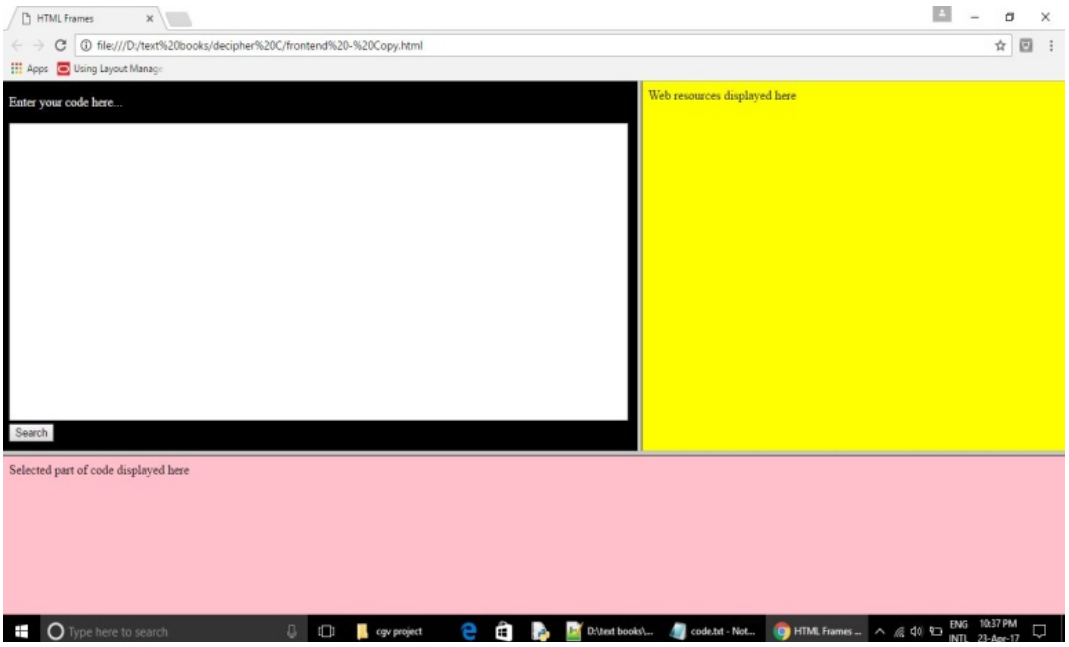

Figure 5.1. Front End Snapshot

6. Conclusions

To conclude Decipher C is aimed to be a powerful learning tool for novice C programmers to have a deeper understanding while learning the language, as well as a tool for skilled programmers to remember specific topics or to understand certain sections of code while in doubt. Decipher C can hence be developed into a bigger website that can provide programmers with the right explanations in very less amount of time in the busy lives of programmers. Hence becoming a very useful tool for self-learning to help novices as well as skilled programmers to clear certain concepts.

Decipher C is designed to be a website that can explain a C program, but has the potential to evolve. By exploiting the modularity of the processing segment, Decipher C can be used as part of a larger educational tool that teaches programming. The tokenizer finds the tokens to check for and their regular expression from the core database. By altering these, Decipher C can be extended to cover multiple programming languages, making it usable by a larger set of users.

REFERENCES

[1] http://www.c-sharpcorner.com/article/c-sharp-artificial-intelligence-ai-programming-a-basic-object/

[2] https://github.com/westes/flex

[3] http://www.opener-project.eu

[4] Qi Song & Yong duan song, Data-Based Fault-Tolerant Control of High-Speed Trains with Traction/Braking Notch Nonlinearities and Actuator Failures *State Key Laboratory of Rail Traffic Control and Safety, Beijng Jiaotong University, Beijing, China, 2011.*

[5] Tao Li.Delay-Slope-Dependent Stability Results of Recurrent Neural Networks*Dept. of Inf. & Commun., Nanjing Univ. of Inf. Sci. & Technol., Nanjing, China.*

[6] K.S. Narendra, Identification and control of dynamical systems using neural networks *Dept. of Electr. Eng., Yale Univ., New Haven, CT, USA.*