International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 03 | Mar -2017 www.irjet.net p-ISSN: 2395-0072

# An efficient Approach to Produce Source Code by Interpreting Algorithm

**Priyanka Motkari, Bhagyashree Wable, Supriya Walzade, Pooja Velhal**

*Student, Dept. of Computer Engineering, KKWIEER, Nashik, Maharashtra, India*

---

**Abstract -** *People may possess good logical skills along with great algorithmic solution designing capabilities but the inadequate knowledge of programming languages makes them handicapped. Neophyte programmers may find it difficult to learn general programming skills and syntactical skills simultaneously. Visually impaired developers suffer as they spend double the time in eradicating syntactical errors as compared to any programmer with a normal vision. The conversion of an algorithm to code is still at an early stage. Effective conversion of algorithms mentioned in natural English language to code will enable programmers to focus on logic building and confine them from syntactical errors, further it will also aid the visually impaired programmers. Although beneficial, fulfillment of such a converter encounters multiple challenges like limitations imposed due to semantics of the English language, case frames, etc. An algorithm to program converter is an interpreter that is capable of converting algorithms in English (with fixed input format) to C code whose flexibility of interpretation has been enhanced by using synonyms and by the introduction of a personalized training model.*

***Key Words***: **Interpreters, Parsing, NLP, Personalized, Trigger words**

## 1. INTRODUCTION

Programming is used in wide variety of domains like astronomy, industrial automation, financial analysis, microbiology, etc. [5]. It has become ubiquitous highly efficient solution offered by programming and Information Technology has been playing a vital role in the rapid growth and transformation of today. Algorithms form the fundamental blocks of programming and are core to solution designing. Implementation of these algorithms using programming languages serves as a major hurdle.

Though people have good logical skills along with great algorithmic solution designing capabilities but due to lack of knowledge of programming languages make them handicapped. It is difficult for neophyte programmers to learn syntactical skills and general programming skills simultaneously. Therefore it is necessary to develop software that is capable of converting algorithms written in natural language to a programming language. The person can focus on problem solving and he becomes free from syntactical worries using this software. Although such software may be very beneficial, various challenges are involves for its realization.

The rest of the paper organizes as follows: Section 2 discusses the literature survey; Section 3 highlights the challenges faced implementing natural language to code interpreter; Section 4 puts forth the conceptual model; Section 5 gives a concluding remark and outlines the future scope.

## 2. LITERATURE SURVEY

There is a common factor between Natural Language Processing and Programming languages which is "Language". Natural Language Processing and Programming languages are very important domains in computer science but very less importance has been given to the interaction in between these two fields [6, 8]. Previously study has been done to develop interpreter which convert algorithm in natural language to the programming language source code. But each of such is having certain limitations. Examples of such interpreter are ALGOSmart, Natural Java and Semi Natural Language Algorithm to Programming Language Interpreter.

- **ALGOSmart**

ALGOSmart [5] is an interpreter which converts pseudo which is written using XML to the programming language source code which is in C and Java. But the ALGOSmart interpreter forces an extra overhead on users by making it compulsory for them to have knowledge about a set of predefined XML tags and their correct implementation and use.

- **NaturalJava**

The other proposal was NaturalJava. It is natural language based user interface which allows a user to enter algorithm in natural English language and it provides the corresponding Java code. The archetypical implementation of the proposal mentioned above, called NaturalJava,
Have three main modules as shown in Figure below PRISM [7] allowed the user to input an algorithmic statement in English language which was in turn passed to Sundance.

Sundance then generated the corresponding case frames which were analyzed by PRISM by calling Tree Face.
After each operation Tree Face generated the source code which was then presented to the user by PRISM.

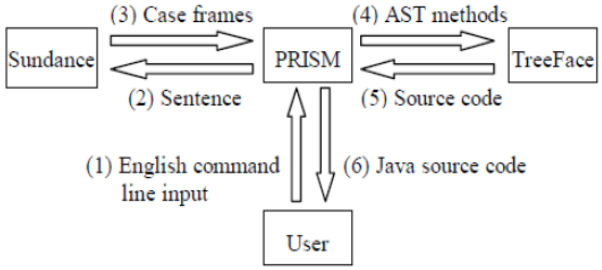


**Fig -1**: Architecture of Natural Java [3]

The major limitation in the aforementioned model is imposed by the finite set of case frames and the ability to comprehend natural language input. We will deal will these challenges in detail in the next section.

- **Semi Natural Language Algorithm to Programming Language Interpreter**

The third proposal was Semi Natural Language Algorithm to Programming Language Interpreter [1]. This translator converts algorithm in natural English language to code in C and Java This interpreter has many semantic challenges such as it does not support multiple variable declaration, it also does not support printing the value of variables. Such limitations imposes constraint on user while developing fully functional program

## 3. CHALLAENGES

Before interpreting natural language algorithms into formal code few attempts have been made. Most challenges faced implementing natural language algorithm to code interpreter revolve around the following aspects.

•Using Part Of Speech (POS) tagging algorithm it is easy to tag individual words. While semantics of the algorithm as a whole becomes difficult to interpret and process.

•Every programming language has its own features. The aspects of various programming language becomes more difficult to incorporate to be identified and interpreted by natural language processing.

•Every individual has a different way of thinking and different method of expressing a single idea. Because of that flexibility of identifying and interpreting natural language algorithm is limited.

The NaturalJava system [7] uses the finite set of case frames. The concept of mapping an active verb to a programming action is used by NLP (Natural Language Processing) for NLP (Natural Language Programming). That means, if a particular set of words is not present in the algorithm line, a trigger to activate parsing and interpretation will not take place. To overcome the aforementioned challenge, we are using a synonym finder to increase the vocabulary repertoire of system. Also, individualistic writing style of users can be accepted into the system. This input helps to generate personalized training model for every user to make interpretation of natural language more flexible.

## 4. CONCEPTUAL MODEL

In order to address the aforesaid challenge of flexibility, we have proposed a model consisting of an interpreter and related interacting modules. The system accepts an algorithm as an input from the user. On that algorithm, basic Natural Language Processing is applied line by line. After that, the processed output is passed to the interpreter. At the interpreter module, first identified the statement type and accordingly, it is parsed into formal C code. The code is displayed to the user which is forwarded from the interpreter module .Hence, the conceptual Model consists of four modules interacting with each other to accept an algorithm in natural language and interpret it in formal language. The model is shown in Figure .The modules are:

1 User
2 Basic Algorithm Processing
3 Interpreter
4 Synonyms
5 Personalized Training Model

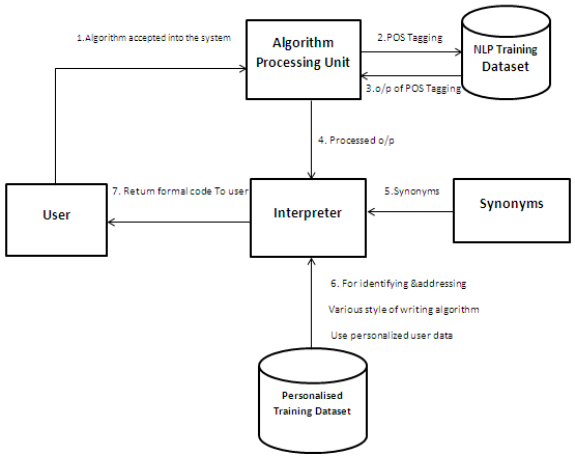


**Fig -2**: Conceptual Model.

### 1. User module:

This module indicates the end user. An algorithm is accepted into the system, via a desktop application. The algorithm is processed by the other modules and a Formal C language code is returned to the user.

**2. Basic Algorithm Processing module:**

After accepting the algorithm from the user, basic natural language processing is applied line by line. Lines and words are separated and Part Of Speech tagging is applied to the algorithm. This module sets stage for interpretation.

Consider statement- initialize integer i to 5

The output of this statement after applying basic algorithm processing would be - initialize_NNinteger_NNi_NNto_TO 5_CD, where NN is noun, TO is to and CD is Cardinal Number.

**3. Interpreter module:**

This is the core module of the model. The interpreter Works in two stages.

**Type identification:** The input sentence is identified as declaration, initialization, input, conditional, looping etc. statement. This is done by identifying a trigger word in the statement. The interpreter contains case frames that map a trigger words to a statement type. Consider the statement – initialize integer i to 5, the interpreter first looks for part of speech tags or keywords. Thus, initialize would be recognized as a keyword and the statement is forwarded to initialize module for parsing statement to code.

**Parsing into formal C code:** Once the statement is correctly identified, it is sent to the specific module for parsing. Here, using POS tags and the sentence structure, the algorithmic line is converted to formal code. Here, the key is to identify and address different styles of writing algorithms and correctly parsing them. For this, we implement a personalized training model that learns style of writing algorithms, therefore improving flexibility of writing and accuracy of interpretation. Thus statement "initialize integer i to 5", is interpreted to form "int i=5".

**4. Synonyms:**

The Flexibility of identifying a trigger for the interpreter module increase by the synonyms, the force of trigger words is increased by not only feeding words manually but by using Synonyms as well. A large set of words increases the probability of a statement being correctly identified and parsed.

**5. Personalized Training Model**

Another powerful method to increase flexibility is by using a personalized training model. Users would be asked to input natural language statements for expressing their individualistic writing style. This style would be gauged and adapted to by the system. Thus, the next time the user would type a similar statement; it would be easily recognized and parsed by the system. This module is under implementation. This module would increase accuracy of interpreting algorithm to code by a great deal.

## 5. EXPERIMENTAL SETUP

**Hardware Resources Required**

- Computer (Minimum Configuration )
- Hard disk: 40 GB
- Processor: Core i3 and above
- Clock Speed: 3.0 GHz
- RAM: 4 GB

**Software Resources Required**

- Operating System: Windows 7 and above
- Front End: C#.Net
- Programming Editor: Visual Studio 2010
- Framework: 4.0 or Above
- Programming Language: C#.Net
- Back-end Database: N/A

## 6. USER SCENARIO

Front end will be provided to the user where user will enter algorithm in natural English language. An example input algorithm - to determine which number is greater among two numbers, is shown below

**Input Algorithm:**

```
input an integer no1, no2
if no1 greater than no2 then
print "no1 is greater"
else
print "no2 is greater"
end if
```

This input will then be sent line by line to the NLP module which will carry out POS tagging and generate the corresponding output as shown below

**POS Tagged Algorithm:**

```
input_NN an_DT integer_NN no1,no2_.
if_IN no1_DT greater_JJR than_IN no2_CD then_RB
print_NN "_`` no1_NNS is_VBZ greater_JJR "_''
else_RB
print_NN "_`` no2_NNS is_VBZ greater_JJR "_''
end_NN if_IN
```

The POS Tagged Algorithm will be generated line by line and each line will be subjected to interpretation by making use of synonyms and dataset from the personalized model. The corresponding C, CPP and Java code will be generated line by line and then it will be merged to produce the final formal code as shown below

**C Code:**

```
int no1, no2;
scanf ("%d" , &no1);
scanf("%d" , &no2);
if (no1 > no2)
{
printf("no1 is greater");
}
```

```
else
{
printf("no2 is greater");
}
```

**CPP Code:**

```
int no1, no2;
cin >> no1;
cin >> no2;
if (no1 > no2)
{
cout << "no1 is greater";
}
else
{
cout << "no2 is greater";
}
```

**Java Code:**

```
int no1, no2;
no1 = Scanner.nextInt();
no2 = Scanner.nextInt();

if (no1 > no2)
{
System.out.println("no1 is greater");
}
else
{
System.out.println("no2 is greater");
}
```

## 7. CONCLUSION AND FUTURE SCOPE

The system consists of User, Basic Algorithm Processing, Interpreter, Synonyms and Personalized Training Dataset modules which interact to form formal code An algorithm to program converter is an interpreter that is capable of converting algorithms in English (with fixed input format) to "C", "CPP" and "Java "code whose flexibility of interpretation has been enhanced by using synonyms and by the introduction of a personalized training model. Effective conversion of algorithms mentioned in natural English language to code will enable programmers to focus on logic building and confine them from syntax worries, further it will also aid the visually impaired programmers. Although beneficial, implementation of such converter encounters numerous challenges like demarcation entailed due to semantics of the English language, case frames, etc. We have opened promising results using our current model and we plan to extend it and incorporate functions, arrays, declarations and pointers. This part can be covered by creating further modules with associated triggers and logic for the same. Further, we aim to overcome the challenge related to semantics as part of our future scope.

## REFERENCES

[1] Prof. Swapnali Kurhade and Sharvari Nadkarni, Semi Natural Language Algorithm to Programming Language Interpreter, International Conference on Advances in Human Machine Interaction (HMI - 2016), March 03-05, 2016

[2]Guillaume Cabanac; Muthu Kumar Chandrasekaran, Joint workshop on bibliometric-enhanced information retrieval and natural language processing for digital libraries (BIRNDL 2016)

[3]Venera Arnaoudova; Sonia Haiduc; Andrian Marcus; Giuliano Antoniol, The Use of Text Retrieval and Natural Language Processing in Software Engineering,ICSE 2015.

[4] Veronika Vincze; Richárd Farkas,De-identification in natural language processing,Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention.

[5] Suvam Mukherjee and Tamal Chakrabarti, Automatic Algorithm Specification to Source Code Translation, in Indian Journal of Computer Science and Engineering, 2011 on, Vol. 2, No. 2, pp. 146 159, April May 2011.

[6] Rada Mihalcea, Hugo Liu, Henry Lieberman, NLP (Natural Language Processing) for NLP (Natural Language Programming), in the 7th International Conference on Computational Linguistics and Intelligent Text Processing, LNCS, Mexico City, February 2006.

[7] David Price, Ellen Riloff, Joseph Zachary and Brandon Harvey, NaturalJava:A Natural Language Interface for Programming in Java, in the Proceedings of the 2000 ACM on Intelligent User Interfaces Conference, pp. 207 211, January 2000.

[8] BALLARD, B., AND BIERMAN, A, Programming in natural language: NLC as a prototype, In Proceedings of the 1979 annual conference of ACM/CSC-ER (1979).