IJRESM
International Journal of Research in Engineering, Science and Management
Volume-2, Issue-5, May-2019
www.ijresm.com | ISSN (Online): 2581-5792

A Comparative Study of Different Techniques for Translation of English Algorithm to C Program

Kaushik Singh[1], Gaurav Karkal[2], Dhanush Reddy[3], Sini Anna Alex[4]
[1,2,3]*Student, Department of CSE, M. S. Ramaiah Institute of Technology, Bengaluru, India*
[4]*Assistant Professor, Department of CSE, M. S. Ramaiah Institute of Technology, Bengaluru, India*

***Abstract*: Natural Language Processing deals with computer understanding of human language. Basically, the main objective of Natural Language Processing (NLP) is to write the computer code is simple language using words which appear in English dictionary for easy interpretation by 3rd party users. Computer codes might be tedious and difficult to understand for some individuals, so by using NLP even individuals who are not that qualified in this department feel it easier to grasp and understand the code. English Algorithm inputted to the system and an equivalent C program is generated. The output C Program file is a coded interpretation of the English language written code which will perform the action intended in the English program. Through this implementation, we have been successful in achieving this objective. In this paper we present a comparison between syntax directed translation scheme and non-syntax directed translation scheme. We also define modules in each method and provide a comparison report between the 2 methods.**

***Keywords*: Natural Language Processing(NLP), syntax directed translation, algorithm.**

1. Introduction

Natural Language Processing is distinguished as a hard problem due to human language spoken to be unclear. Being accessible to everyone, computer programs can be written in natural language. The system has many advantages like a person that can write in English but not in programming language, would still be able to program. Also the code written in English would be easy to read and understand than code written in programming languages.

Looking at this example

Input 2 numbers

or

Enter two numbers

Does 'input' and 'enter' mean the same?Are 'two' ,'2' refer to same meaning. Also the type has to be determined and referencing of the variable needs to resolved. The ambiguity of the grammar is a major concern in natural language processing. One possibility to resolve ambiguity is to make dictionary for words whose meanings are the same. Same tokens are generated and passed to the parser where the parser rules output the words having same meaning using the syntax directed translation schema. Restricted natural language restricts the vocabulary available to the users and force them to construct sentences in a specific way. The system should produce a single correct parse and output.

Accuracy of translation and speed are the major goals in any translation system. Smart tools to handle expressions,words in the target language are to be developed. Grammar needs to be optimized and efficient parsing algorithm and data structure are required. Bison LALR and GLR parsing algorithm along with Faster token supplier Flex handle the above goals efficiently[1]. Hence the proposed system gives an opportunity to eliminate some of the ambiguities of translating a natural language algorithm to program code.

2. Literature survey

The programming languages which support natural language are called supplemented programming. Natural language programming is easier than using high level language due to syntax constraints and user understanding. The Programming languages like FORTRAN, BASIC and COBOL support natural language programming.

- *KlarDeutsch:* KlarDeutsch is the newer supplemented programming language. It was developed by Andover Corporation in the year 1995. It can control machines and equipments with natural language, which changes the paradigm of giving instructions to machines. KlarDeutsch is very straightforward and by one way or another out-dated as yet utilizing go to-statements. It demonstrates that there is a need to utilize normal language along with programming language in the natural language field. Engineers can't generally be comfortable with the most current improvements in programming innovation causing difference between programming languages utilized at college and in business. KlarDeutsch is as of now utilized to deal with the critical applications, language usage along with programming is much in excess of a thought a long way from viable use.
- *AppleScript:* AppleScript is a content language created

by Apple, Inc. in 1993. It is substantially more ellaborate than KlarDeutsch. Notwithstanding that, there is no chance to get of including client characterized elements, all activities must be carried out on existing articles. Then again, AppleScript is one of only a handful couple of programming languages, which were multilingual, at any rate for quite a while. Until form 8.5 of Mac OS, AppleScript projects could be written in a few natural languages, among them English, German, Spanish, French, Italian, just as Japanese and Chinese. Sadly, Apple surrendered the multilingual methodology, due to the confounded costumer support.

A. *ALGO smart*

ALGOSmart is a translator which changes over pseudo which is composed utilizing XML to the programming language source code which is in C and Java. Be that as it may, the ALGOSmart translator makes it mandatory for clients to have information about a lot of predefined XML labels and their right usage and use.

B. *Conversion of semi natural language algorithm*

This interpreter changes over algorithm in natural English language to code in C and Java This translator has numerous semantic difficulties, for example, it doesn't support various variable presentation, it additionally does not support printing the variable values. Such confinements forces constraints on client while growing developing useful programs [2].

3. Challenges

Prior to translating natural language algorithms into formal code few endeavors have been made. Most challenges faced in converting natural language algorithm to code interpreter rotate around the following reasons.

- Using Part of Speech (POS) tagging algorithm it is anything but difficult to label singular words. While semantics of the algorithm overall ends up hard to translate and process.
- Every programming language has its own highlights. The parts of different programming language turn out to be progressively hard to fuse to be recognized and deciphered by natural language handling.
- Every individual has an alternate state of mind furthermore, extraordinary strategy for communicating a solitary thought. In light of that adaptability of recognizing and deciphering natural language algorithm is restricted.

4. Non-syntax-directed translation

So as to address the aforementioned challenge of flexibility, we have proposed a model comprising of an interpreter and related connecting modules [4]. The system acknowledges an algorithm as a contribution from the user. On that algorithm, basic Natural Language Processing is applied line by line. From that point onward, the prepared yield is passed to the interpreter. At the interpreter module, first recognized the announcement type and in like manner, it is parsed into formal C code. The code is shown to the user which is sent from the interpreter module. Hence, the theoretical Model comprises of four modules cooperating with one another to acknowledge an algorithm in natural language what's more, translate it in formal language. The model is appeared Fig. 1. The modules are:

- User
- Basic Algorithm Processing
- Interpreter
- Synonyms
- Personalized Training Model
-

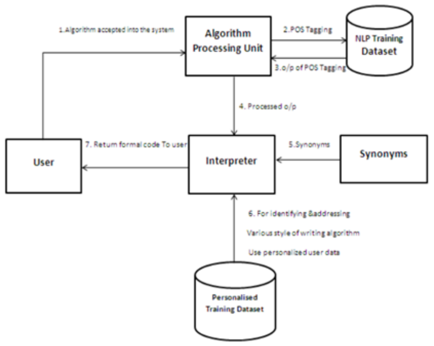
1.Algorithm accepted into the system
Algorithm Processing Unit
2.POS Tagging
NLP Training Dataset
3.o/p of POS Tagging
4. Processed o/p
User
7. Return formal code To user
Interpreter
5.Synonyms
Synonyms
6. For identifying &addressing
Various style of writing algorithm
Use personalized user data
Personalised Training Dataset

Fig. 1. Non syntax directed schema

1. *User Module:* This module shows the end user. An algorithm is acknowledged into the system, by means of an application. The algorithm is prepared by different modules and a Formal C language code is returned back to the user.
2. *Basic Algorithm Processing module:* After accepting the algorithm from the user, basic natural language processing is connected line by line. Lines also, words are isolated and Part Of Speech tagging is connected to the algorithm. This module sets organize for understanding. Consider explanation instate whole number I to 5 The yield of this announcement in the wake of applying basic algorithm processing would be -initialize_NNinteger_NNi_NNto_TO 5_CD, where NN is thing, TO will be to and CD is Cardinal Number.
3. *Interpreter module:* This is the center module of the model. The interpreter Works in two phases.
4. *Type identification:* The information sentence is recognized as affirmation, introduction, input, contingent, circling and so on proclamation. Recognized trigger word in the statement are mapped to a statement type. Consider the statement – initialize integer I to 5, the interpreter first searches for part of speech labels or watchwords. Consequently, initialize would be perceived as a catchphrase and the statement is sent to initialize module for parsing statement to code.

5. *Parsing into formal C code:* Once the announcement is effectively recognized, it is sent to the particular module for parsing. Here, utilizing POS labels and the sentence structure, the algorithmic line is changed over to formal code. Here, the key is to distinguish and address distinctive styles of composing algorithms and effectively parsing them. For this, we execute a customized training model that learns style of composing algorithms, in this manner improving flexibility of composing and exactness of understanding. In this manner explanation "initialize integer I to 5", is translated to shape "int i=5".
6. Synonyms: The Flexibility of recognizing a trigger for the interpreter module increment by the synonyms, the power of trigger words is expanded by not just nourishing words physically yet by utilizing Synonyms too. A vast arrangement of words builds the likelihood of an announcement being effectively recognized and parsed.
7. *Personalized Training Model:* Another amazing technique to build flexibility is by utilizing a customized training model. Users would be approached to include natural language statements for communicating their individualistic composition style. This style would be checked and adjusted to by the system. In this way, whenever the user would type a comparative statement; it would be effectively perceived and parsed by the system. This module is under execution. This module would increment precision of deciphering algorithm to code by a lot.

5. Syntax-directed-translation

Natural Language System breaks down the sentence into an imperative verb, determiner and noun. Syntax directed translation method is completely driven by the parser. SDT is used to translate the string by attaching a sequence of actions to each rule of a grammar. Parsing a string of grammar produces a sequence of rules. The following system modules have been used for the system development:

- Flex
- Bison
- Gcc Compiler
- Output .c file

A. *Flex*

The program consists of a list of regular expressions consisting of actions to perform on input match. A scanner reads input and input is matched against all of the regular expressions and does necessary action on each match. The regular expressions are converted to an efficient internal form by flex. A scanner module is produced which is compiled and linked to other compiler modules. Flex generates a file that consist yylex() which returns an integer indicating token recognized.

```
C and scanner declarations
%%
Token definitions and actions
%%
C subroutines
```

It contains 3 components:

- C and Scanner declarations: Lex definitions used in the regular expressions and C declarations to include the file produces by Yacc/Bison.
- Token: Consist of regular expression with corresponding actions.
- C subroutines: May contain C code with corresponding actions.
- Lex file is compiled using the command Flex TEC.1 producing file lex.yy.c that defines function yylex().

Bison: In order to convert an annotated CFG (context free grammar) into a deterministic LR-parser a general purpose parser, Bison which employes LALR (1) parser tables is used. It is also able to make a canonical LR(1) parse tables.

It consists of 3 sections:

B. *1st Section*

Ordinary C subroutine declaration part, the specification of the start symbol, a list of tokens that are expected by the parser are contained in the first section. It also includes files with C code, variable declaration and user defined function protocol which are written between “%{“ “ %} “ brackets.

C. *2nd Section*

CFG for the language is contained in the second section. The action along with the production is present within braces. Each Production is distinguished from one another by semicolons and the empty productions remain empty. The action along with the production is present within braces. The multiple character terminal symbols are written in uppercase while the non-terminals appear in lower case. The phrase structural grammar along with its actions which are written in C is included so as to generate result based on SDT (syntax directed translation) schema.

D. *3rd Section*

The file Yacc consisting of C program is present in the third section. The driver routine for the parser is a main() function which class yyparse(). Reporting of the errors is during the parsing is handled by yyerror (). It also contains main function and the yyerror function along with the function which is user defined used in grammar action part allowing the system to be complete [5]. Compiling the Bison file causes the creation of files- “TEC.tab.h”,“TEC.tab.c”. The first one contain tokens to be be included in the scanner defining file. The second file contains the definition for the yyparse function.

6. Results and Conclusion

A. *Non syntax directed translation model*

The framework comprises of User, Basic Algorithm Processing, Interpreter, Synonyms and Personalized Training Dataset modules which communicate to frame a formal code. An algorithm to program converter is an interpreter that is fit for changing over algorithms in English (with fixed information group) to "C", "CPP" and "Java "code whose adaptability of elucidation has been improved by utilizing synonyms and by the presentation of a personalized training model [6]. Successful change of algorithms referenced in common English language to code will empower programmers to concentrate on rationale assembling and restrict them from sentence structure stresses, further it will likewise help the outwardly weakened programmers. Albeit valuable, usage of such converter experiences various challenges like demarcation involved because of semantics of the English language, case outlines, and so on. We have opened promising outcomes utilizing our present model and we plan to expand it and consolidate capacities, clusters, statements and pointers. This part can be secured by making further modules with related triggers and rationale for the equivalent.

Input Algorithm	Output

input an integer num1, num2
if num1 lesser than num2 then
print "num1 is smaller"
else print "num2 is greater"
end if
	int num1, num2;
	scanf ("%d" , &num1);
	scanf("%d" , &num2);
	if (num1 < num2)
	{ printf("number 1 is smaller"); }

B. *Syntax directed translation model*

Model is evaluated with C functions including scanf, if else, for, variables, declarations of pointers and variables. The system was successful in returning accurate meanings in case of each translation The system is capable in handling the ambiguity with words having similar meaning depending on the lexical section. The upper-case and lower-case sensitivity with words is yet to be tackled by the model due to the limited dictionary taken into consideration and the lack of the morphological phase.

System Input	System Output

accept 2 nums
sum 2 nums
print result
or
any other arithmetic operation
	#include<stdio.h>
	int main()
	{
	int no1 ;
	int no2 ;
	int ans ;
	ans=0 ;
	scanf("%d",&no1);
	scanf("%d",&no2);
	ans= no1 + no2 ;
	printf("result is %d",ans);
	return 0;
	}

type your full name
type your gender
type your position
type your DOB
display full name
display gender
display position
display DOB
	#include<stdio.h>
	#include<string.h>
	int main()
	{
	char *full_name;
	char *gender;
	char *position;
	char *DOB;
	gets(full_name);
	gets(gender);
	gets(position);
	gets(DOB);
	printf("Full name is %s",full_name);
	printf("gender is %s",gender);
	printf("position is %s",position);
	printf("DOB is %s",DOB);
	return 0;
	}

accept 4 nums
sum 4 nums.
print result.
	#include<stdio.h>
	#include<string.h>
	int main()
	{
	int num[4];
	int j;
	int ans =0;
	for(j=0;j<4;j++)
	{
	scanf("%d"\n,&num[j]);
	}
	for(j=0j<4;j++)
	{
	ans = ans + num[i] ;
	}
	printf("Result is %d",ans);
	return 0;
	}

References

[1] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman &quot;Compilers: Principles, Techniques, and Tools” Publisher: Addison Wesley. (1986): 796
[2] Biermann, Alan W., and Bruce W. Ballard. &quot;Toward natural language computation.&quot; Computational Linguistics 6, no. 2 (1980): 71-86.
[3] Ballard, Bruce W., and Alan W. Biermann. &quot;Programming in natural language: “NLC” as a prototype.&quot; In Proceedings of the 1979 annual conference, pp. 228-237. ACM, 1979.
[4] Carlos, Cohan Sujay. &quot; Natural Language Programming Using Class Sequential Rules. &quot; In IJCNLP, pp. 237-245. 2011.
[5] Christiansen, Morten H., and Nick Chater. &quot;Connectionist natural language processing: The state of the art.&quot; Cognitive science 23, no. 4 (1999): 417-437.
[6] Cozzie, Anthony, and Samuel T. King. “Macho: Writing programs with natural language and examples.” Technical report, University of Illinois at Urbana-Champaign, 2012.