International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 04 | Apr 2020 www.irjet.net p-ISSN: 2395-0072

Pseudocode to Python Translation using Machine Learning

Vinay Patil[1], Rakesh Pawar[2], Prasad Parab[3], Prof. Satish Kuchiwale[4]

[1,2,3]Student, Computer Engineering, SIGCE, Navi Mumbai, Maharashtra, India
[4]Asst. Professor, Computer Engineering, Smt. Indira Gandhi College of Engineering, Navi Mumbai, Maharashtra, India

---

Abstract - *Pseudocode is an essential concept in the process of learning algorithms and programming languages. It can be in both forms, programmatical and natural language. Programmatical pseudocode can be easily parsed because the syntax is precise and predictable but natural language pseudocode has unpredictable and informal syntax. Pseudocode in general is not meant to be executable and is used as references for implementation. This system makes pseudocode executable through providing programming language source code. It can be helpful for students in the learning process. Existing systems used plain neural networks with cascade feed-forward backpropagation algorithm. The normal implementation of backpropagation is sufficient enough and this project improves upon the architecture by using recurrent neural networks. This project aims at providing a system for pseudocode to source compilation or translation. The proposed system first decomposes the informal statements into a formal intermediate representation which is in XML for faster and simple parsing. Then it will be parsed into Python programming languages. This system will be implemented by using RNNs with deep neural network for sequence to sequence translation with the help of Keras library.*

***Key Words*: Sequence to Sequence, Translation, Pseudocode, Machine Learning.**

1. INTRODUCTION

This project is about creating an application which translates pseudocode to Python. Generally when learning about programming languages and coding, we first learn about different algorithms in our courses. These algorithms are written in simple English. This simple English form or pseudocode form is generally meant to represent the meaning and logic of the program without any syntax or programming language features. This makes the process of learning about core fundamental programming components like conditional statements, loops or concepts like recursion easier to understand.

Giving students an option to test their algorithms by translating the pseudocode to a programming language will enhances the learning experience. But pseudocode is inherently not meant to be executable. There is no standard way of writing pseudocodes so creating a traditional compiler or interpreter for it is not feasible. Machine Learning techniques such as Natural Language Processing could be used for such tasks. NLP systems are already becoming good at solving types of tasks such as translating languages.

The traditional compilers first perform lexical analysis, produce an abstract syntax tree format, perform optimizations and then produce the machine code. There are also source-to-source compilers which translate one programming language to another. This type of compilers have the same workflow but instead of producing machine code, they translate it to another language. This project follows the same paradigm of source-to-source compilers but instead use Machine Learning to process the pseudocode in the initial step. The machine learning module will first parse the pseudocode and generate an abstract syntax tree format which is represent in XML form for easier parsing. This syntax tree format is then parsed and translated recursively, producing the final translated code.

The project is implemented in python using libraries like Keras and NumPy. The program has two text panels, left one for pseudocode and right one for displaying the translated python code. The translated code will appear in the right text panel after the user presses the translate button. When the user wishes to run the program, the console panel is brought into focus and all the standard output is displayed including input prompts.

1.1 Objective

We aim to achieve the following through this project:

- The objective is to create a program which translates pseudocode into executable python code.
- To create generic and abstract guidelines for writing pseudocode but making it extensible as well for advanced users.
- To use machine learning algorithms for effective translation of natural language statements into expressions.
- Implement the machine learning module in Keras.

1.2 Scope

- The program will only be able to parse evaluable and logical statements which are written in simple plain English, instead of conceptual or vague sentences.

- The intermediate language will be translated to python only, although it can be extended to other languages.

2. LITERATURE SURVEY

In paper [1], the proposed system was aimed at making the process of creating translation tools for different languages easier with the help of conceptual meta modelling. The system was divided in into two phases, the first phase translated the pseudocode into intermediate form and the second phase translated the intermediate form into source code. The grammar for the pseudocode in different natural languages is written in EBNF (Extended Backus-Naur Form). The intermediate form was in XML form. The modules for first and second phases are created such that they are reusable e.g. the intermediate to Java translator could be used in any system where Java translation is required, same for French Pseudocode to intermediate translator.

The paper [2] was referenced by the previous paper and acts as a base for it. In this paper 3 algorithms were tested, Back propagation, Cascade-feed forward backpropagation and Radial basis function algorithm. Cascade-forward back propagation neural network is similar to backpropagation neural network except its input layer is connected to the rest of the layers. In Radial basis function algorithm, the hidden layer applies the radial basis function over the values from the input layer. The paper also had predefined syntax for the pseudocode on which the neural network was trained on. The pseudocode was first split into keywords and a matrix of binary numbers was generated based on the index of the keyword from the list of vocabulary, also called as one hot encoding. This matrix was then passed onto the 3 neural networks and the results were then compared. The paper concluded that cascade-feed forward backpropagation gave the best results out of the three.

In paper [3] the pseudocode was represented in the form of XML. The operations were represented using tags in XML. The translation process was done using regular expressions and pattern matching. The pseudo code written in XML use regular expressions and pattern matching to translate this XML pseudo code into C and Java programs. Each tag is searched using regular expression and the appropriate operation is done upon the contents of the tag.

The paper [4] proposes a system for sequence to sequence mapping using recurrent neural networks. The usual deep neural networks cannot map between sequences to sequences. The paper presents an end-to-end approach for the mapping of sequences. The system consists of two major components, an encoder and decoder. The encoder consists of multilayered LSTM cells which convert the input sequence into a context vector. This context vector is then passed onto the decoder which also has deep LSTM cells to generate the target sequence. The paper also compared a phrase based Statistical Machine Translation with LSTM and concluded that LSTM performed better in terms of BLEU score and performance.

3. PROBLEM STATEMENT

Pseudocode is generally meant for learning, writing algorithms or prototyping. Students learn to write pseudocode alongside with programming languages. The learning process can be enhanced by having source code version of the pseudocode. The pseudocode written in natural language can be difficult to quantify and parse into logical form. This project provides a system to convert pseudocode into Python. The system will use machine learning to parse the natural language statements. The natural language statements will be translated into an intermediate representation and this intermediate representation will be then converted into Python.

4. SYSTEM OVERVIEW

4.1.1 Dataset

The dataset used in this project consist of list of natural language statements with their translations. Each pair in the dataset is created such that the system will be able to predict the action performed and also detect the position of the variables in the sentence.
E.g. let c be the sum of a and b \t assign 1 add 6 8

4.1.2 Preprocessing

Every letter in the pseudocode is lower cased. Parsing of the expressions through ML is a difficult task so the expressions are hidden away from the ML model. The whitespace from the expressions is removed so that they appear as a single entity. Some noised variations of the sentences are also generated to make the prediction of the positions of the variables more generalized. The noised variations are generated by adding variable number of paddings in the sentences which helps the system generalize over the input.

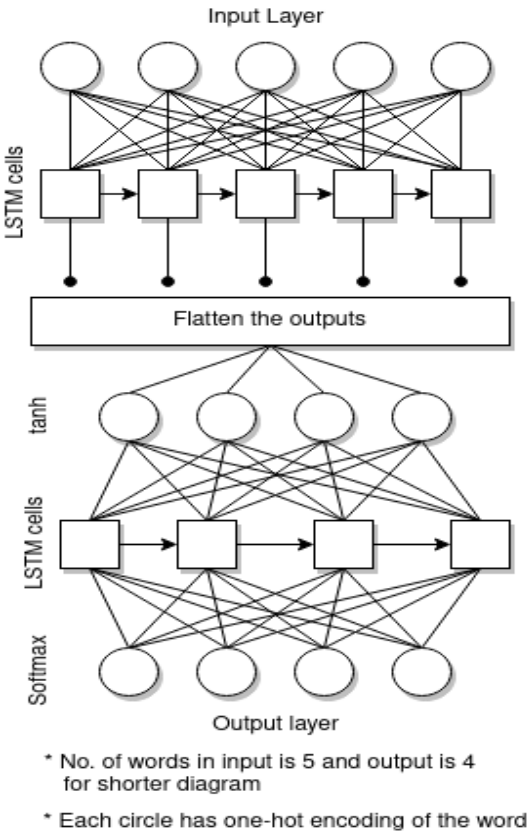
4.1.3 Pseudocode to Intermediate translation

Pseudocode to intermediate translation is handled using a machine learning technique called natural language processing. Recurrent neural network will be used as they are much more efficient and accurate than ANNs and CNNs. LSTM cells are used for the RNN layers. The model loosely follows the sequence to sequence paradigm. In sequence to sequence paradigm the model has two parts encoder and decoder. The encoder takes input and generates a context vector which is then used by the decoder to generate the output sequence [4]. This project attempts to mimic the attention mechanism by first flattening the outputs of the encoder layer and feeding the output to each of the LSTM cell

in the decoder layer [5][6]. The one hot encodings are grouped together to form the input matrix. Outputs of the model will be a shorthand version of the XML form which consists of generic function names and variable positions in the original sentence. The Keras model is implemented using the Sequential model. Inputs to the model will be first converted to one hot form.

Steps:

1. Convert the sentence in an array of words.
2. Covert each word into their index number in the vocabulary, if not exists then put unknown or skip.
3. Convert this array of numbers into one hot encoding.
4. Create the input matrix by grouping the one hot encodings.
5. Feed the input matrix into the network.
6. Network will output the prediction.



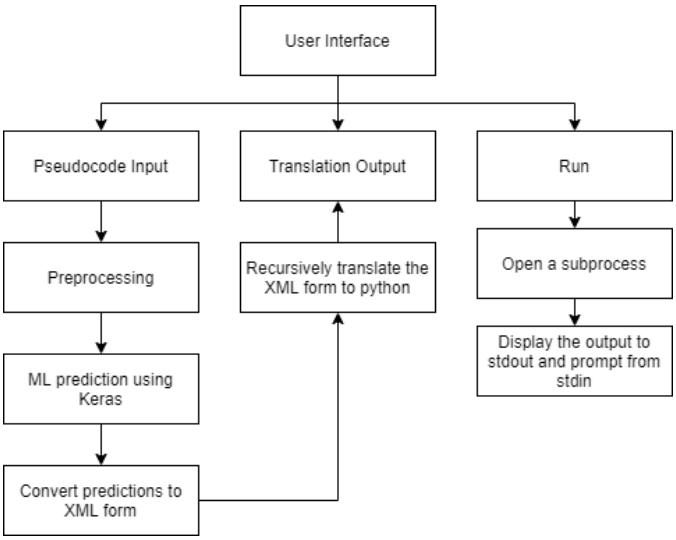
4.1.4 Intermediate to Python translation

The output of the previous module is a shorthand version of the XML form. This shorthand form is first converted to the XML form for easier traversing. This XML form is essentially the syntax tree of the pseudocode. The translation in this module is done via regular expressions and recursion. Each tag is visited and replaced with its translation recursively until the final translation is formed.

Steps:

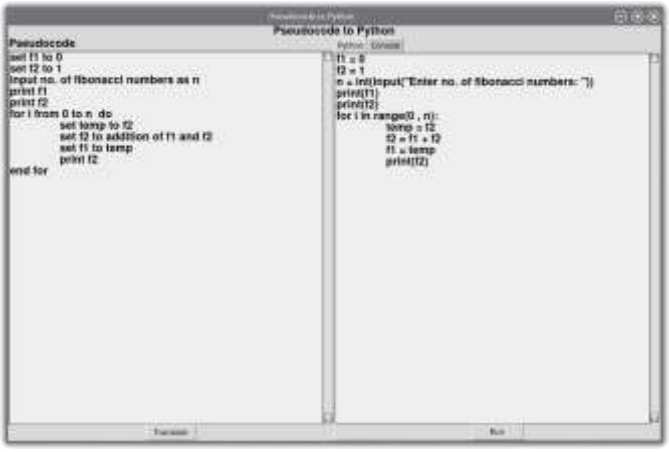
1. Parse all the variables in the intermediate form using regular expressions.
2. Check if variable names clashes with python keywords.
3. Visit every node in the XML syntax tree recursively.
4. Start replacing each pattern with the appropriate python code using regular expression.

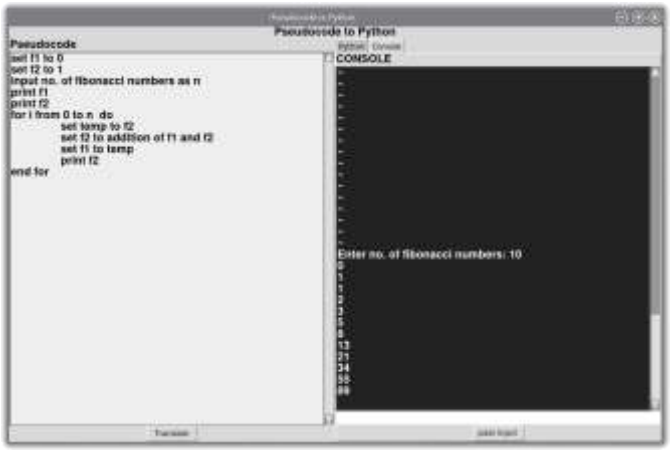
4.2. Flowchart



4.3. Result

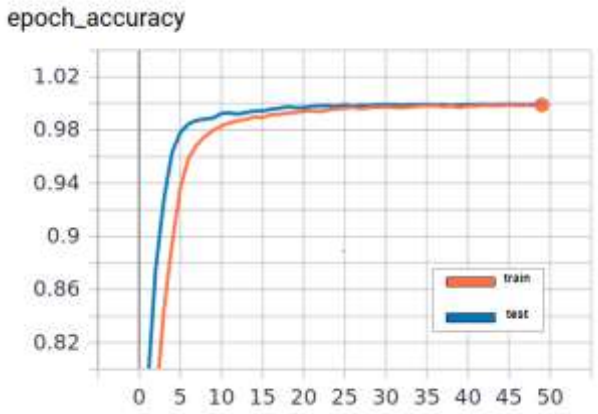
4.3.1 Screenshots



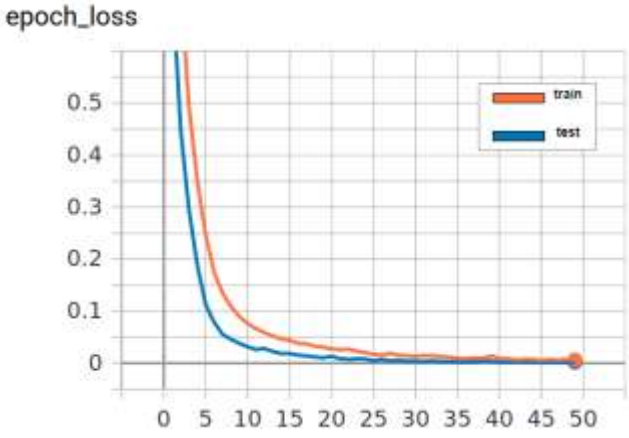


4.3.2 Graphs

Training and test accuracy:



Training and test loss:



6. CONCLUSIONS

This project will be beneficial for students in learning algorithms and programming languages like python and study how the logic is being implemented in the actual code. In this proposed system we present a program which is a pseudocode to python translator and will act as an helping aid in the learning process. The pseudocode is not executable so the proposed system will enable the students to test and execute their algorithms.

ACKNOWLEDGEMENT

It is pleasant task to express gratitude to all those who contributed in many ways to this project and made it an unforgettable experience for us. First of all, we would like to thank our guide Prof. Satish Kuchiwale .This work would not have been possible without her guidance, support and encouragement. We are highly indebted to Dr.Sunil Chavan, Principal, Smt. Indira Gandhi College of Engineering Mrs.Sonali Deshpande , Head of the Department, Computer Department and also all the faculty members. We are thankful to lord for keeping us energized that every end seems to be new beginning.

REFERENCES

[1] T. Dirgahayu and S.N. Huda " IEEE: Automatic Translation from Pseudocode to Source Code: A Conceptual-Metamodel Approach" IEEE, 2017.

[2] S.O.Hasson and F.M.R. Younis "Automatic Pseudocode to Source Code Translation Using Neural Network Technique", Intl. J. Engineering and Innovative Technology (IJEIT),vol. 3, no. 11, 2014.

[3] S. Mukherjee and T. Chakrabarti, "Automatic algorithm specification to source code translation", Indian J. Computer Science and Engineering (IJCSE), vol. 2, no. 2, 2011.

[4] Ilya Sutskever, Oriol Vinyals and Quoc V. Le.2014 "Sequence to Sequence Learning with Neural Networks", NIPS.

[5] Dzmitry Bahdanau, Kyunghyun Cho and Yoshua Bengio 2015 "Neural machine translation by jointly learning to align and translate", ICLR.

[6] Minh-Thang Luong, Hieu Pham, and Chrisopher D Manning. 2015 "Effective approaches to attention-based neural machine translation", EMNLP.

BIOGRAPHIES

Vinay Ganesh Patil, Pursuing the Bachelor degree (B.E.) in Computer Engineering from Smt. Indira Gandhi College of Engineering (SIGCE), Navi Mumbai. His current research interests include Graphics & Machine Learning.

Rakesh Janu Pawar, Pursuing the Bachelor degree (B.E.) in Computer Engineering from Smt. Indira Gandhi College of Engineering (SIGCE), Navi Mumbai. His current research interests include Web Development & Machine Learning.

Prasad Sunil Parab, Pursuing the Bachelor degree (B.E.) in Computer Engineering from Smt. Indira Gandhi College of Engineering (SIGCE), Navi Mumbai. His current research interests include Web Designing & Machine Learning

Prof. Satish Lalasaheb Kuchiwale, Obtained the Bachelor degree (B.E. IT) in the year 2007 from Rajarambapu Institute of Technology (RIT), Rajaramnagar, Sakharale, and Master degree (M.E. Computer) from Lokamanya Tilak College of Engineering(LTCE), Navi Mumbai. He is Asst. Professor in Smt. Indira Gandhi College of Engineering of Mumbai university and having about 12 yrs. of experience.