



IJIRCCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 10, Issue 7, July 2022

ISSN INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA

Impact Factor: 8.165

 9940 572 462

 6381 907 438

 ijircce@gmail.com

 www.ijircce.com

Generating Python Code from English Comments using Transformer – GPT-3 in NLP

Bhanu Prasad Ch, M Vayuputra, Mallikarjuna N, Pavankumar T S, Dr. Anitha G, Rekha B H

^{1,2,3,4}UG Student, Dept. of ISE, Bapuji Institute of Engineering and Technology, Karnataka, India

^{5,6}Assistant Professor, Dept. of ISE, Bapuji Institute of Engineering and Technology, Karnataka, India

ABSTRACT: Software development is complex, time consuming and it requires knowledge on many different levels such as understanding algorithm, programming language and frameworks. There has been advancement in neural network where researchers have tried to close the gap between Natural Language description of the system and actual implementation in code using NLP techniques. Python has close gap between natural language descriptions compared to other languages. So, it is better to choose python for code generation. In this paper transformer is used, it is a deep learning model used primarily in the field of natural language process. Transformers are better than all the other architectures because transformers totally avoid recursion, by processing sentences as a whole and by learning relationships between words. GPT-3 is the transformer used in the paper for the generation of code. GPT-3 is a pre-trained transformer model developed by Open AI. GPT-3, a pre-trained transformer accepts natural language comments as input then do the analysis of comments and generate corresponding code in python. The gap is to be closed by analysis of programming comments and generating the required code snippet.

KEYWORDS: AI, NLP, VS Code, GPT-3, Transformer, API.

I. INTRODUCTION

Software has become a crucial component of modern society, directly affecting billions of people's everyday work and life. Software development is complex, time consuming and it requires knowledge on many different levels such as understanding algorithm, programming language and frameworks. There has been advancement in neural network where researchers have tried to close the gap between Natural Language description of the system and actual implementation in code using NLP techniques. Succeeding in the generation of even small code snippet can save time for countless software engineers, which translates to save resources across multiple industries.

Python has close gap between natural language descriptions compared to other languages. So, it is better to choose python for code generation. Comments in Python are the lines in the code that are ignored by the compiler during the execution of the program. Comments enhance the readability of the code and help the programmers to understand the code very carefully. In this paper, a transformer is used, it is a deep learning model, used primarily in the field of natural language process. A transformer adopts the mechanism of self-attention, differentially weighting the significance of each part of the input data. It is used primarily in the field of natural language processing Transformers are better than all the other architectures because they totally avoid recursion, by processing sentences as a whole and by architectures' learning relationships between words.

GPT -3 is the transformer used in the paper for the generation of code. GPT -3 is a pre – trained transformer model developed by Open AI. The gap is to be closed by analysis of programming comments and generating the required code snippet. GPT-3, a pre – trained transformer accepts natural language comments as input then do the analysis of comments and generate corresponding code in python.

In this paper, we propose a machine learning model to automate the task of writing code by assisting developers in writing code snippet. In order to do this, a pre-trained model is fine tuned to give the required result.

II. LITERATURE SURVEY

Software development is difficult and requires knowledge on many different levels such as understanding programming algorithms, languages, and frameworks. In addition, before code is being worked on, the system requirements and

functionality are first discussed in natural language, after which it is sometimes visualized for the developers in a more formal language such as Unified Modelling Language. ^[1]

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English to German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data. ^[2]

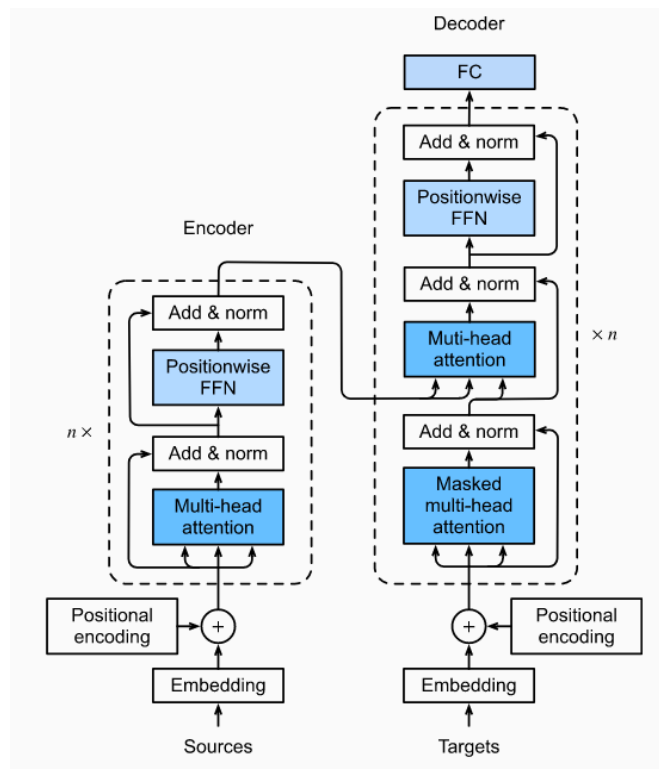
Recent progress in natural language processing has been driven by advances in both model architecture and model pretraining. Transformer architectures have facilitated building higher-capacity models and pretraining has made it possible to effectively utilize this capacity for a wide variety of tasks. Transformers is an open-source library with the goal of opening up these advances to the wider machine learning community. The library consists of carefully engineered state-of-the-art Transformer architectures under a unified API. Backing this library is a curated collection of pretrained models made by and available for the community. Transformers is designed to be extensible by researchers, simple for practitioners, and fast and robust in industrial deployments. The library is available at <https://github.com/huggingface/transformers>. ^[3]

Solving mathematical word problems by understanding natural language texts and by representing them in the form of equations to generate the final answers has been gaining importance in recent days. At the same time, automatic code generation from natural language text input (natural language programming) in the field of software engineering and natural language processing (NLP) is drawing the attention of researchers. Representing natural language texts consisting of mathematical or logical information into such programmable event driven scenario to find a conclusion has immense effect in automatic code generation in software engineering, e-learning education, financial report generation, etc. In this paper, we propose a model that extracts relevant information from mathematical word problem (MWP) texts, stores them in predefined templates, models them in object-oriented paradigm, and finally map into an object-oriented programming (OOP) language (JAVA) automatically to create a complete executable code. The codes are then executed automatically to output the final answer of the MWP. The proposed system can solve addition-subtraction type MWPs and produced an accuracy of 90.48% on a subset of the standard arithmetic questions dataset. ^[4]

III. METHODOLOGY

In this section we describe our method of implementation by explaining transformers and implementation, Transformers can be understood in terms of their three components:

1. An Encoder that encodes an input sequence into state representation vectors.
2. An Attention mechanism that enables our Transformer model to focus on the right aspects of the sequential input stream. This is used repeatedly within both the encoder and the decoder to help them contextualize the input data.
3. A Decoder that decodes the state representation vector to generate the target output sequence.



3.1 Understanding the Training Data

To train the model, the required data is collected from GitHub. GitHub repositories is considered, as it is open source. GitHub packages is used for authenticating the API of GitHub to access the repositories. The search_repositories method is used for getting the data of python based repositories. By using the os module, each repository is cloned to the repos folder created.

3.2 Data cleaning

The repositories downloaded consists different kind of files .The files contain different types of files such as JavaScript file, PHP file, Java file etc. For training purpose, only python files are utilized. So to get the files related to python (i.e., .py files), the os module is used to remove the filenames not ending .py

3.3 Data Pre-processing

The cleaned data (i.e., after getting all .py files) is passed to the module for pre-processing where the next line (blank lines) are replaced with <N> to reduce the processing time.

3.4 Data Tokenization

Tokenization is the process of dividing text into a set of meaningful pieces. These pieces are called tokens. For example, a chunk of text is divided into words, or it can be divided into sentences. Our Input (SRC) and Output (TRG) sequence exist in the form of single strings that need to be further tokenized in order to be sent into the transformer model.

3.5 Feeding data to model

To feed data into our model we first create batches using Pytorch's torchtext.data. Bucket Iterator. This ensures that Inputs that have similar lengths stay together in a single batch for ease of training. Then we feed in our tokenized Input (SRC) batches into the encoder and use the tokenized Output (TRG) batches in our decoder. Our goal is to use the encoder's tokenized English Input (SRC) to predict the tokenized Python Output (TRG) via the decoder. The tokenized predictions are then untokenized via the untokenize function of Python's source code tokenizer.

IV. RESULTS & DISCUSSION

The required English comment is given as input in VS code which is enabled by VS code extension. The given input is shown in Fig 4(a) i.e, “find the even numbers from range 1 to 100”.

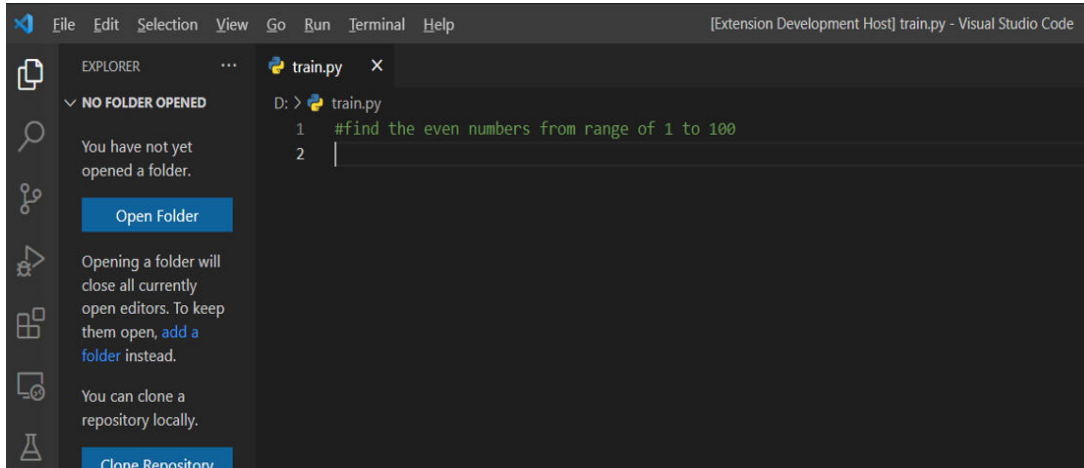


Fig 4(a): English comments are given as input

Python code generator is selected to pass English comment as the input and is shown in the fig 4(b)

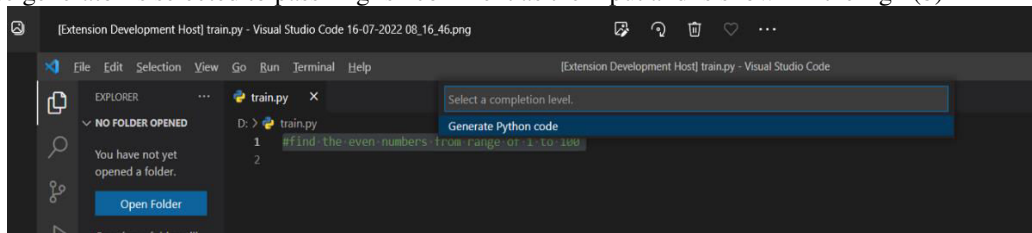


Fig 4(b): We need to select the generator

The given input is passed to python code generator with the help of VS code extension and it is shown in the figure Fig 4(c)

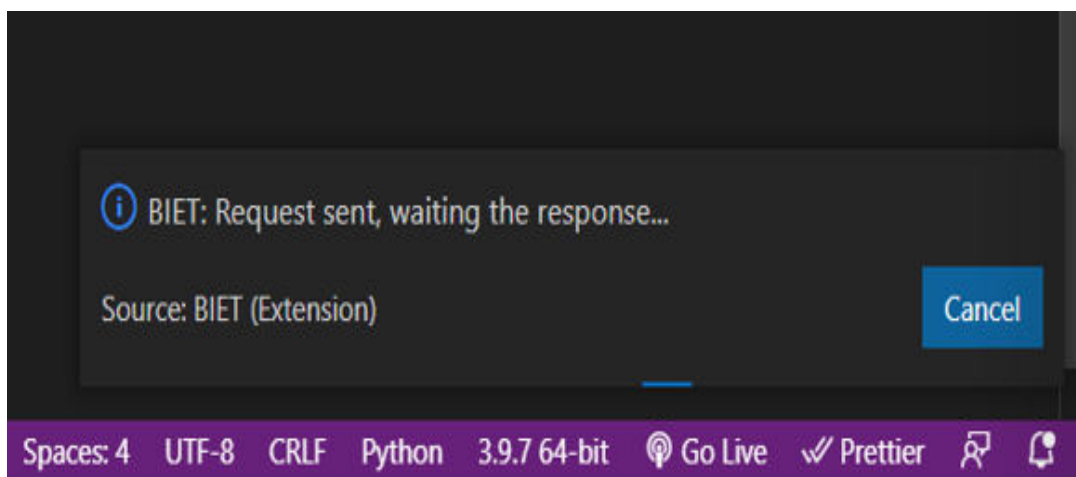
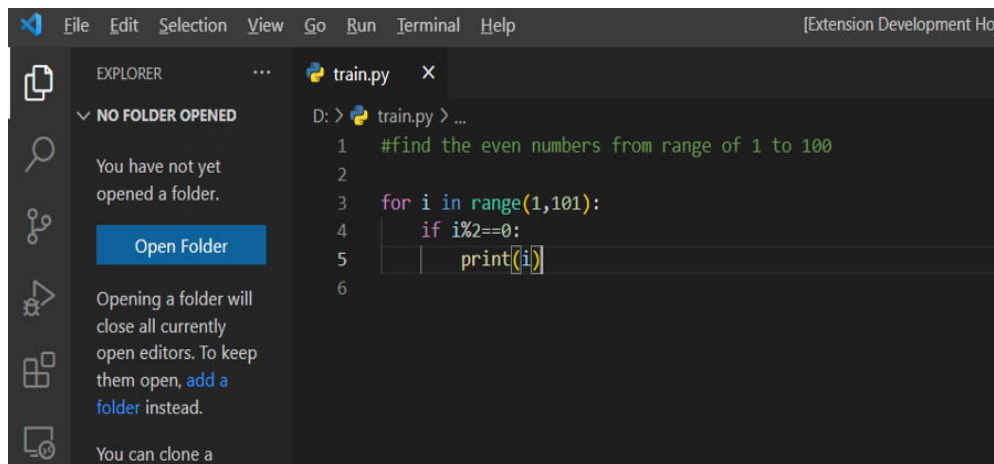


Fig 4(c): Comments are passed to python code generator with the help of VS code extension

The required output is generated for the given English comment, “find the even numbers from range 1 to 100” and the output is shown in Fig 4(d).



```
File Edit Selection View Go Run Terminal Help [Extension Development Ho]
EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
You can clone a

train.py X
D: > train.py > ...
1 #find the even numbers from range of 1 to 100
2
3 for i in range(1,101):
4     if i%2==0:
5         print(i)
6
```

Fig 4(d): Output is generated for required comment

V. CONCLUSION

Code Generator is predestined to change the day-to-day of many programmers all over the world. It is not going to replace programmers, but to help us elevate our productivity while coding, especially repetitive coding tasks. As development time and development cost decreases which effectively boosts the productivity.

REFERENCES

1. Hendrig sellik, Natural Language processing techniques for code generation, Delft University of Technology – 2019
2. Ashish Vaswani, Noam shazeer, NikiParmar, JakobUszkoreit, Llion Jones, Aldan N.Gomez, Lukasz Kalser, Attention is all you need, 6th Dec 2017
3. Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, PierricCistac, Tim Rault, Transformers: State of the art Natural Language Processing,Hugging Face, Brooklyn – 2020
4. Sourav Mandal, SudipNaskar, Natural Language Programming with Automatic Code Generation towards solving Addition-Subtraction word Problems-2017



INNO  SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.165

 **doi**[®]
cross **ref**

ISSN INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

 9940 572 462  6381 907 438  ijircce@gmail.com



www.ijircce.com

Scan to save the contact details