Configuration Manual

MSc Research Project
Data Analytics

Pramod Belur Ramesh
Student ID: 19211015

School of Computing
National College of Ireland

Supervisor: Dr. Athanasios Staikopoulos

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Pramod Belur Ramesh
Student ID:	19211015
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Dr. Athanasios Staikopoulos
Submission Due Date:	31/01/2022
Project Title:	Configuration Manual
Word Count:	758
Page Count:	9

Signature:	
Date:	31st January 2022

Configuration Manual

Pramod Belur Ramesh
19211015

1 Introduction

This is a configuration manual designed to provide all the configuration and installations required to run the project, "Automated Code Summarization of Program Subroutines Using Deep Learning Technologies". The reminder of this manual is divided into the following,

2 Hardware Requirements

Although it is recommended that this project be run on a cloud provider such as Google Collaboratory, it is also possible to run this project but the following hardware configurations are assumed to be bare-minimum. Table 1 shows the list of hardware required.

Table 1: Hardware Requirements.

Hardware	**Specification** 1
RAM	8gb
Processor	Intel(R) Core(TM) i5-8300H
Operating System	Windows 10, 64 Bit or Ubuntu 18.04
Storage	1 TB HDD
GPU	NVIDIA GeForce GTX1650

3 Software Requirements

This section covers the software that is needed for the reproduction of this project.

1. **Github Desktop:** The project was developed using github as a repository, for the maintenance of files, in local laptop as well as to connect to Colaboratory, it is best to install github desktop[1].

2. **Anaconda Distribution:** Anaconda is a popular distribution framework that can host a number of software products such as Jupyter notebook and other visualization tools, this framework will help in running the projects locally.

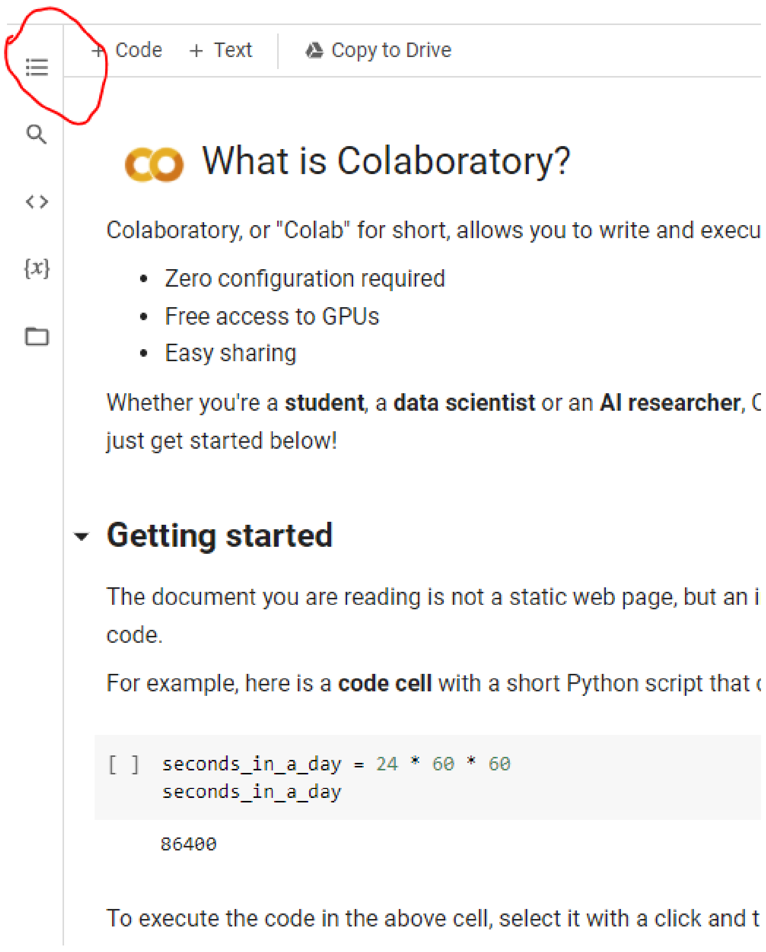
[1]https://desktop.github.com/

3.1 Google Colaboratory

Colaboratory is software service where in Jupyter notebook can be run with GPU services, this project vastly used this feature to run the code from github as well as maintain data in google drive. The following shows steps to connect colab to github and gdrive.

3.1.1 Connect to github from Google Colaboratory

1. Open google Colaboratory and click on the three lines as shown in Figure 1



What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execu

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, C
just get started below!

Getting started

The document you are reading is not a static web page, but an i
code.

For example, here is a **code cell** with a short Python script that

```
[ ]  seconds_in_a_day = 24 * 60 * 60
     seconds_in_a_day

     86400
```

To execute the code in the above cell, select it with a click and t

Figure 1: Step 1 to connect to drive from colab.

2. Click on the folder icon as shown in Figure 5
3. Click on the google drive logo as indicated in Figure 3
4. The code that needs to be run for the drive to be mounted gets added to the cell of the notebook, running which will mount the drive to the colab, shown in 4.

3.1.2 Connect to github from Google Colaboratory

1. Click on file open new notebook on colab

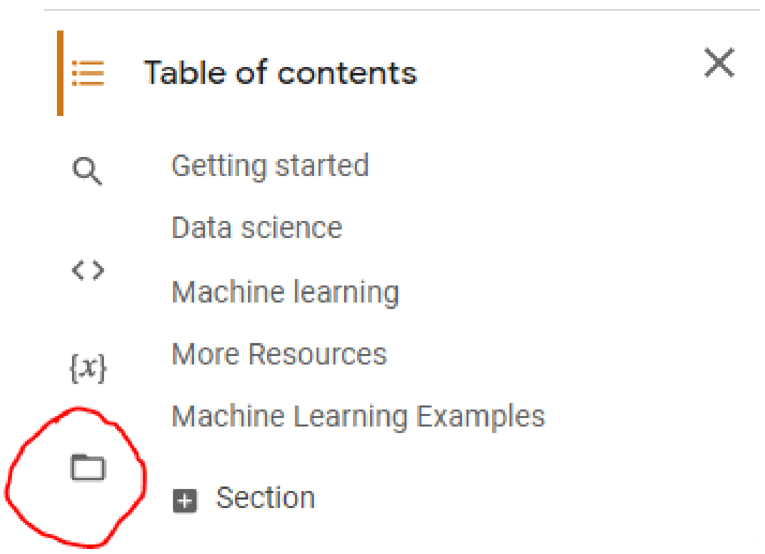


Figure 2: Step 2 to connect to drive from colab.

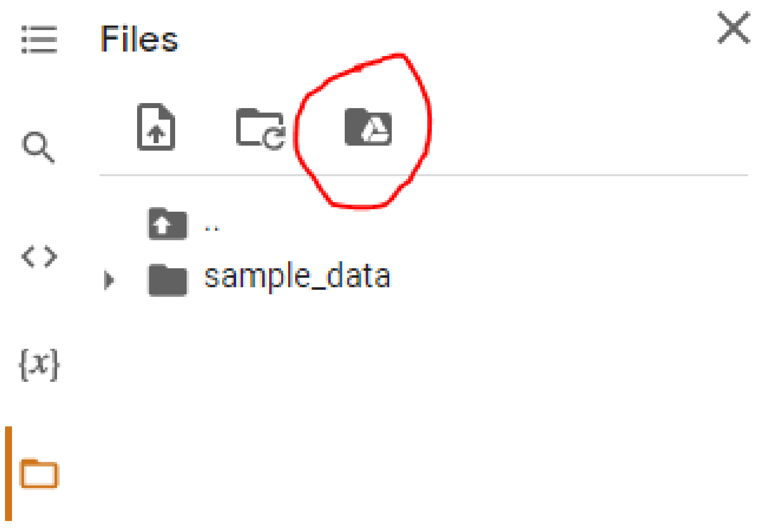
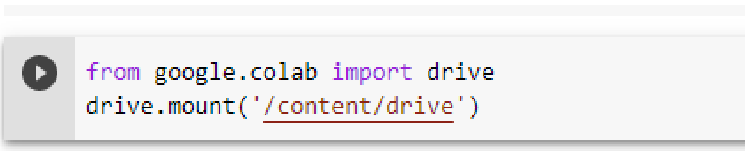


Figure 3: Step 3 to connect to drive from colab.



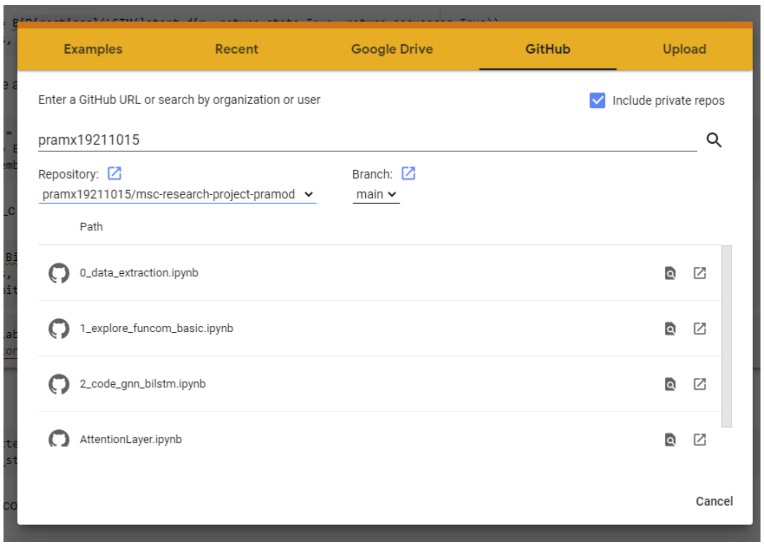
```
from google.colab import drive
drive.mount('/content/drive')
```

Figure 4: Step 4 to connect to drive from colab.

2. In the pop up choose 'github' and make sure to check 'include private repo', if the repository is private.

3. Github will ask for permission to share repo with google, saying yes to which code can be pushed directly from colab to github repo.

The github account will be connected to colab and all the files, branches and repos from the github repo user will start to show in colab as shown in Figure **??**.



Enter a GitHub URL or search by organization or user

[✓] Include private repos

pramx19211015

Repository: pramx19211015/msc-research-project-pramod

Branch: main

Path
0_data_extraction.ipynb
1_explore_funcom_basic.ipynb
2_code_gnn_bilstm.ipynb
AttentionLayer.ipynb

Figure 5: github connected to colab.

4 Navigating Code and Data

This section shows how to navigate through the actual code base of the data and various important steps in the code along with screen shots for facilitate reproduction (LeClair and McMillan; 2019).

4.1 Dataset

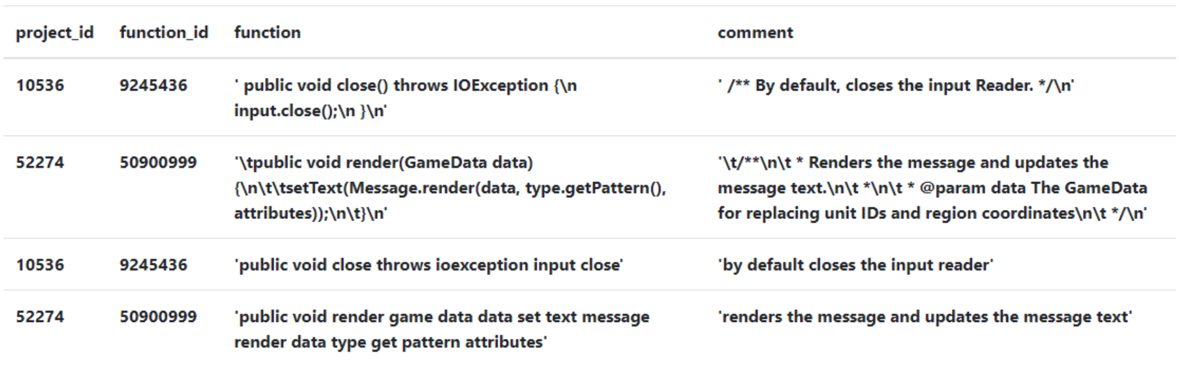
1. Dataset was downloaded from the funcom[2] website provided by LeClair et al. (2019) and also recommended in their seminal article that describes best practices in detail. The dataset looks like as shown in Figure 6

2. Downloaded dataset looks this way once it is extracted is shown in Figure 7.

3. Function in 'funcom_extraction.ipynb' that downloads the data programatically for exploration is shown in Figure 8.

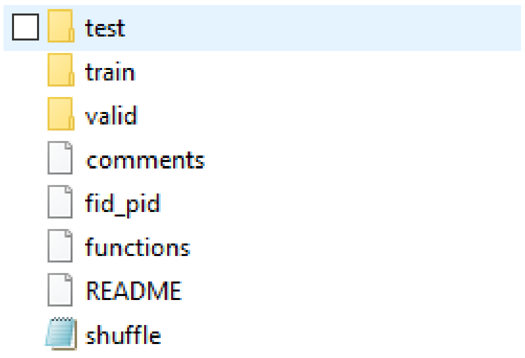
4. Dataset that is fed into a pandas dataframe, a section of function and comments is shown in Figure 9

[2]http://leclair.tech/data/funcom/



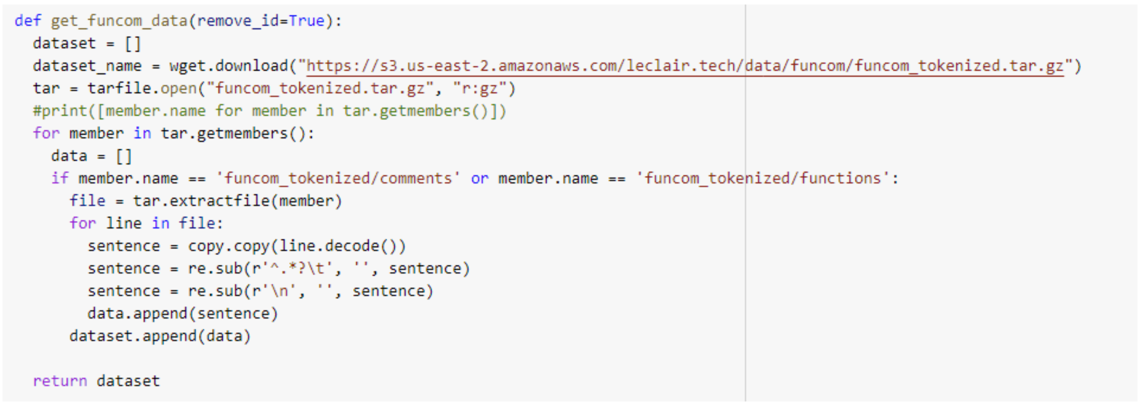
project_id	function_id	function	comment
10536	9245436	' public void close() throws IOException {\n input.close();\n }\n'	' /** By default, closes the input Reader. */\n'
52274	50900999	'\tpublic void render(GameData data) {\n\t\tsetText(Message.render(data, type.getPattern(), attributes));\n\t}\n'	'\t/**\n\t * Renders the message and updates the message text.\n\t *\n\t * @param data The GameData for replacing unit IDs and region coordinates\n\t */\n'
10536	9245436	'public void close throws ioexception input close'	'by default closes the input reader'
52274	50900999	'public void render game data data set text message render data type get pattern attributes'	'renders the message and updates the message text'

Figure 6: Funcom dataset.



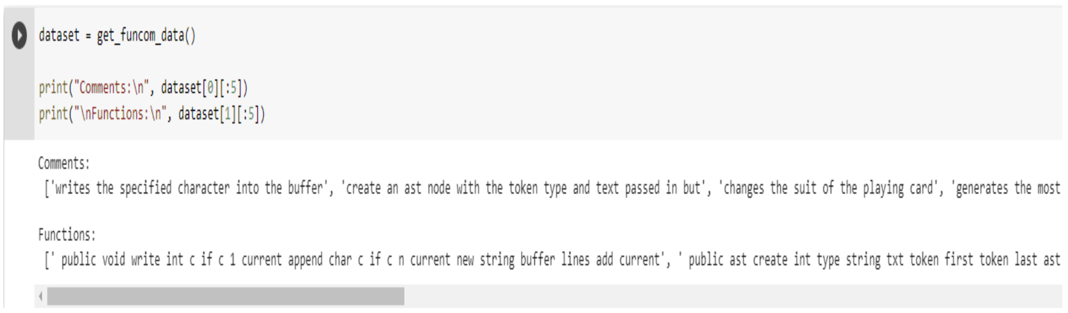
test
train
valid
comments
fid_pid
functions
README
shuffle

Figure 7: Downloaded dataset.



```
def get_funcom_data(remove_id=True):
  dataset = []
  dataset_name = wget.download("https://s3.us-east-2.amazonaws.com/leclair.tech/data/funcom/funcom_tokenized.tar.gz")
  tar = tarfile.open("funcom_tokenized.tar.gz", "r:gz")
  #print([member.name for member in tar.getmembers()])
  for member in tar.getmembers():
    data = []
    if member.name == 'funcom_tokenized/comments' or member.name == 'funcom_tokenized/functions':
      file = tar.extractfile(member)
      for line in file:
        sentence = copy.copy(line.decode())
        sentence = re.sub(r'^.*?\t', '', sentence)
        sentence = re.sub(r'\n', '', sentence)
        data.append(sentence)
      dataset.append(data)

  return dataset
```

Figure 8: Programmatic Downloaded dataset.



```
dataset = get_funcom_data()

print("Comments:\n", dataset[0][:5])
print("\nFunctions:\n", dataset[1][:5])
```

```
Comments:
 ['writes the specified character into the buffer', 'create an ast node with the token type and text passed in but', 'changes the suit of the playing card', 'generates the most

Functions:
 [' public void write int c if c 1 current append char c if c n current new string buffer lines add current', ' public ast create int type string txt token first token last ast
```

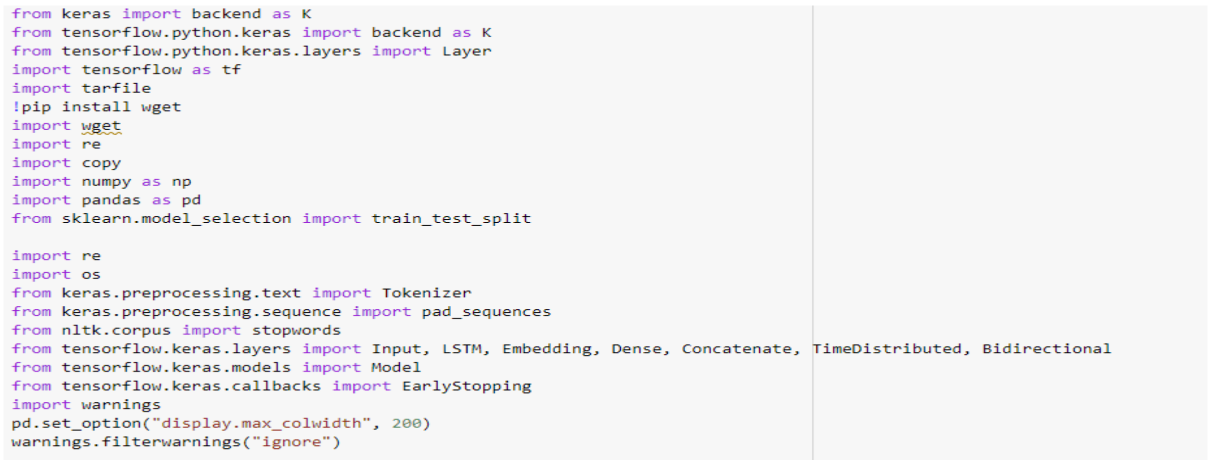
Figure 9: Programmatic exploration of dataset.

4.2 Modelling Code with keras and tensoflow

This section walks through the main parts of the code that is present in the notebook '2_code_gnn_bilstm.ipynb'.

4.2.1 Importing libraries

The requried libraries are imported as shown in Figure 10



```
from keras import backend as K
from tensorflow.python.keras import backend as K
from tensorflow.python.keras.layers import Layer
import tensorflow as tf
import tarfile
!pip install wget
import wget
import re
import copy
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

import re
import os
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from nltk.corpus import stopwords
from tensorflow.keras.layers import Input, LSTM, Embedding, Dense, Concatenate, TimeDistributed, Bidirectional
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping
import warnings
pd.set_option("display.max_colwidth", 200)
warnings.filterwarnings("ignore")
```

Figure 10: All imports that are needed.

4.3 Graph Layer

The graph layer implemented by LeClair et al. (2020) is used here, the model was better for understanding. This model is placed in the 'custom' folder and the file name is 'GCNLayer.py'.

4.4 Encoder & decoder

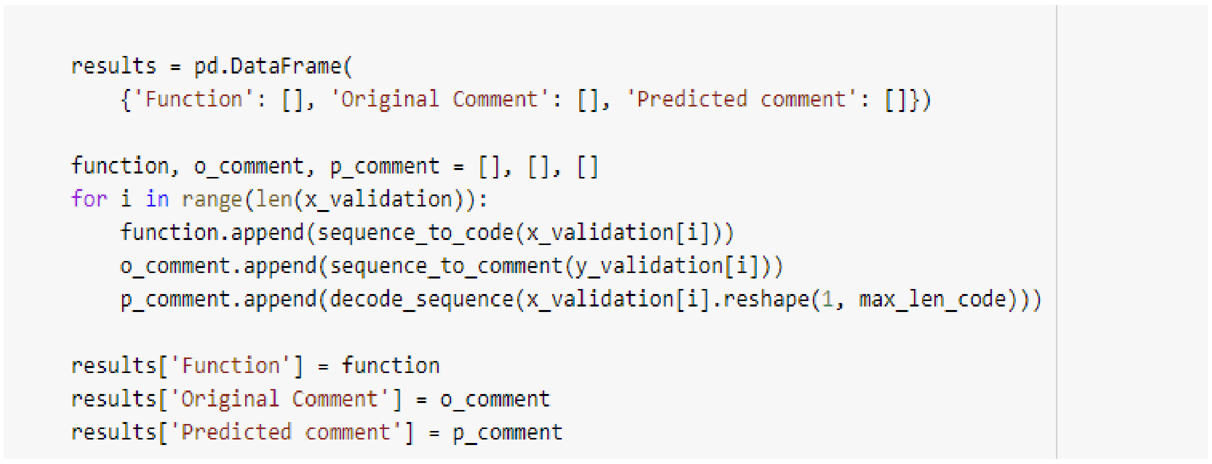
Encoder and decoder layer are added in the file '2_code_gnn_bilstm.ipynb', this includes an embedded layer, input layer and a bidirectional LSTM. GCNLayer is also added as an input to the encoder. The decoder also contains embedded space, and takes in the initial states of encoder as the input.

4.5 Attention Layer

The famous attention framework proposed by is present in 'AttentionLayer.ipynb'. This attention layer is based on the paper by Vaswani et al. (2017).

4.6 Running the model

The method 'decode_sequence' needs to be run with the input sequence, this input sequence needs to be a tensor and should be reshaped using numpy as shown in Figure 11.

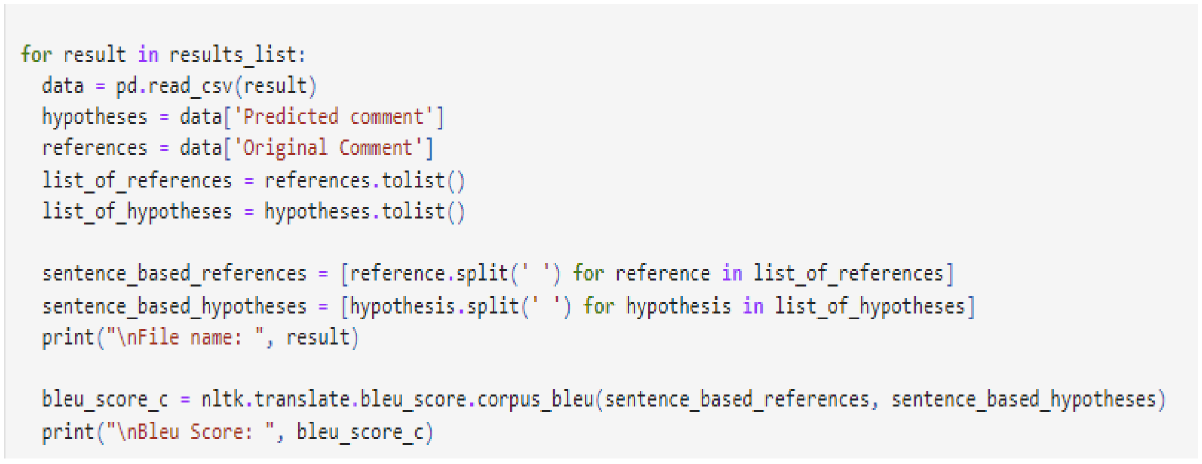


```
results = pd.DataFrame(
    {'Function': [], 'Original Comment': [], 'Predicted comment': []})

function, o_comment, p_comment = [], [], []
for i in range(len(x_validation)):
    function.append(sequence_to_code(x_validation[i]))
    o_comment.append(sequence_to_comment(y_validation[i]))
    p_comment.append(decode_sequence(x_validation[i].reshape(1, max_len_code)))

results['Function'] = function
results['Original Comment'] = o_comment
results['Predicted comment'] = p_comment
```

Figure 11: Running the model.

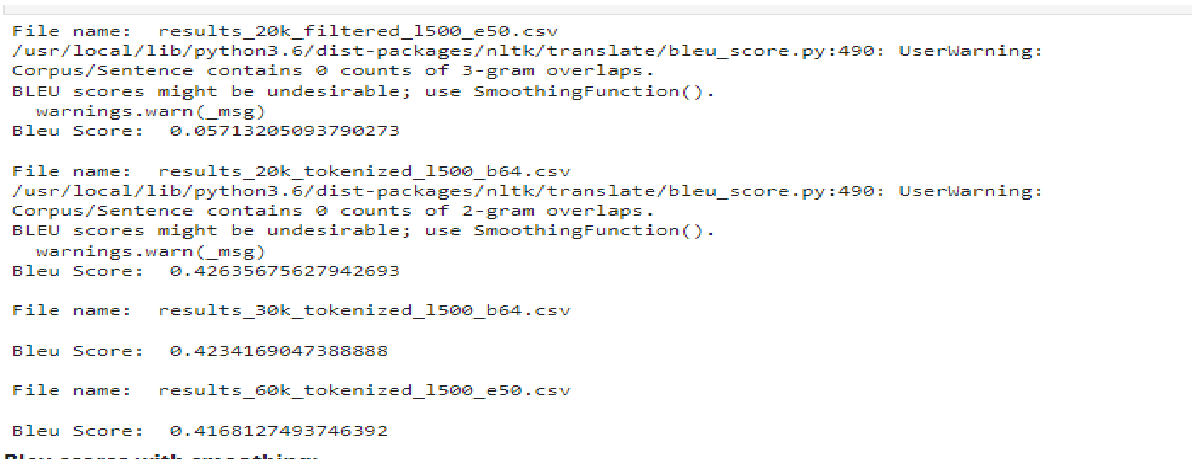
4.7 Evaluation

The code is evaluated by using the 'Evaluation.ipynb', the code to separate id and comment is shown in Figures 12 and 13.



```
for result in results_list:
  data = pd.read_csv(result)
  hypotheses = data['Predicted comment']
  references = data['Original Comment']
  list_of_references = references.tolist()
  list_of_hypotheses = hypotheses.tolist()

  sentence_based_references = [reference.split(' ') for reference in list_of_references]
  sentence_based_hypotheses = [hypothesis.split(' ') for hypothesis in list_of_hypotheses]
  print("\nFile name: ", result)

  bleu_score_c = nltk.translate.bleu_score.corpus_bleu(sentence_based_references, sentence_based_hypotheses)
  print("\nBleu Score: ", bleu_score_c)
```

Figure 12: Evaluation code



```
File name:  results_20k_filtered_1500_e50.csv
/usr/local/lib/python3.6/dist-packages/nltk/translate/bleu_score.py:490: UserWarning:
Corpus/Sentence contains 0 counts of 3-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
  warnings.warn(_msg)
Bleu Score:  0.05713205093790273

File name:  results_20k_tokenized_1500_b64.csv
/usr/local/lib/python3.6/dist-packages/nltk/translate/bleu_score.py:490: UserWarning:
Corpus/Sentence contains 0 counts of 2-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
  warnings.warn(_msg)
Bleu Score:  0.42635675627942693

File name:  results_30k_tokenized_1500_b64.csv

Bleu Score:  0.4234169047388888

File name:  results_60k_tokenized_1500_e50.csv

Bleu Score:  0.4168127493746392
```

Figure 13: Example of evaluation result during one of the epochs.

References

LeClair, A., Haque, S., Wu, L. and McMillan, C. (2020). Improved code summarization via a graph neural network, *Proceedings of the 28th International Conference on Program Comprehension*, ICPC '20, Association for Computing Machinery, New York, NY, USA, p. 184–195.
URL: *https://doi.org/10.1145/3387904.3389268*

LeClair, A., Jiang, S. and McMillan, C. (2019). A neural model for generating natural language summaries of program subroutines, *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, IEEE, pp. 795–806.

LeClair, A. and McMillan, C. (2019). Recommendations for datasets for source code summarization, *arXiv preprint arXiv:1904.02660* .

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. (2017). Attention is all you need, *Advances in neural information processing systems*, pp. 5998–6008.