Configuration Manual

MSc Research Project
Data Analytics

Aditya Mukherjee
Student ID: X20161131

School of Computing
National College of Ireland

Supervisor: Dr Rejwanul Haque

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Aditya Mukherjee
Student ID:	X20161131
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Dr Rejwanul Haque
Submission Due Date:	31/01/2022
Project Title:	Configuration Manual
Word Count:	564
Page Count:	7

Signature:	
Date:	30th January 2022

Configuration Manual

Aditya Mukherjee
X20161131

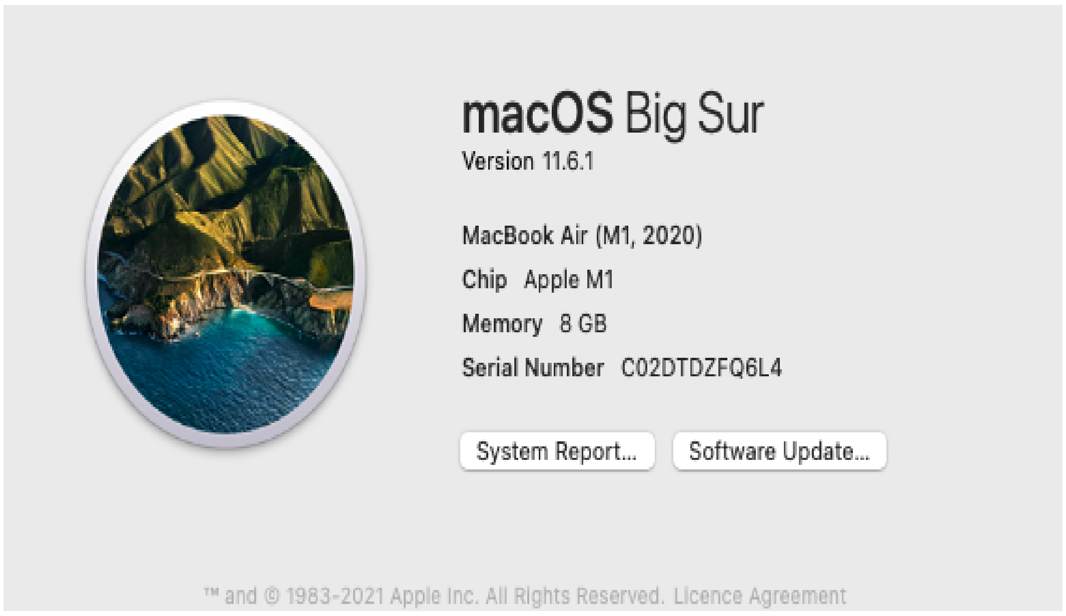
1 Introduction

The main goal of this manual is to provide correct procedure and configuration of the hardware to any one who wishes to reproduce the result or wishes to run the code. We will cover every little part which is important to code and to our research topic.The structure of this config. file is as follow:
Section 2) We will highlight software and hardware specification to run our code.
Chapter 3) we will discuss how to load the data into the system
Chapter 4) In this section we will discuss about the EDA and Data pre-processing.
Chapter 5) In this section we will discuss about the aspect of the models.

2 Enviroment

2.1 Configuration of Hardware and software

For this research, the machine that we used is shown in figure[1]. Machine has Apple arm M1 chipset with 8Gb of ram and 256gb hardisk.


macOS Big Sur
Version 11.6.1

MacBook Air (M1, 2020)
Chip Apple M1
Memory 8 GB
Serial Number C02DTDZFQ6L4

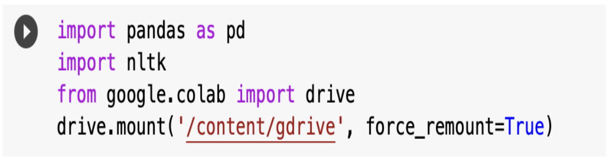
™ and © 1983-2021 Apple Inc. All Rights Reserved. Licence Agreement

Figure 1: Hardware Config

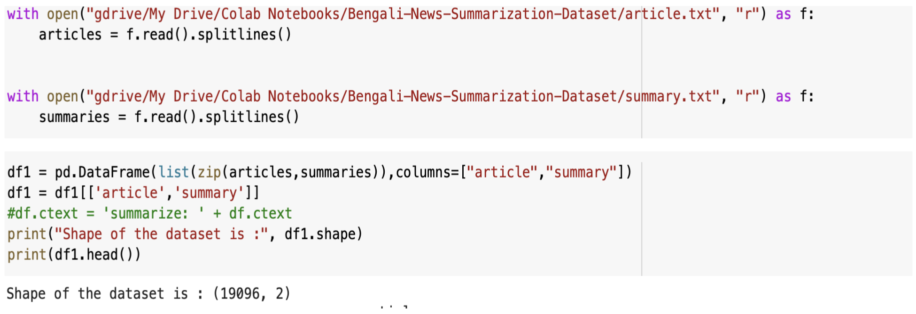
For this research study, Google Colaboratory have used. For EDA purpose or where we didn't need help of GPU , there we used its default resources. By default Colab gives us 12gb ram and 110Gb of disk space. But when we haev worked on our Deep learning method we have Used our qouta of GPU and TPU.

3 Collection of data

For this research paper we have used the dataset from kaggle. Kaggle is an open platform, where anyone can download the data.
Link to the dataset-: https://www.kaggle.com/prithwirajsust/bengali-news-summarization-dataset. Below code to how we can upload the data



```
import pandas as pd
import nltk
from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)
```



```
with open("gdrive/My Drive/Colab Notebooks/Bengali-News-Summarization-Dataset/article.txt", "r") as f:
    articles = f.read().splitlines()


with open("gdrive/My Drive/Colab Notebooks/Bengali-News-Summarization-Dataset/summary.txt", "r") as f:
    summaries = f.read().splitlines()
```

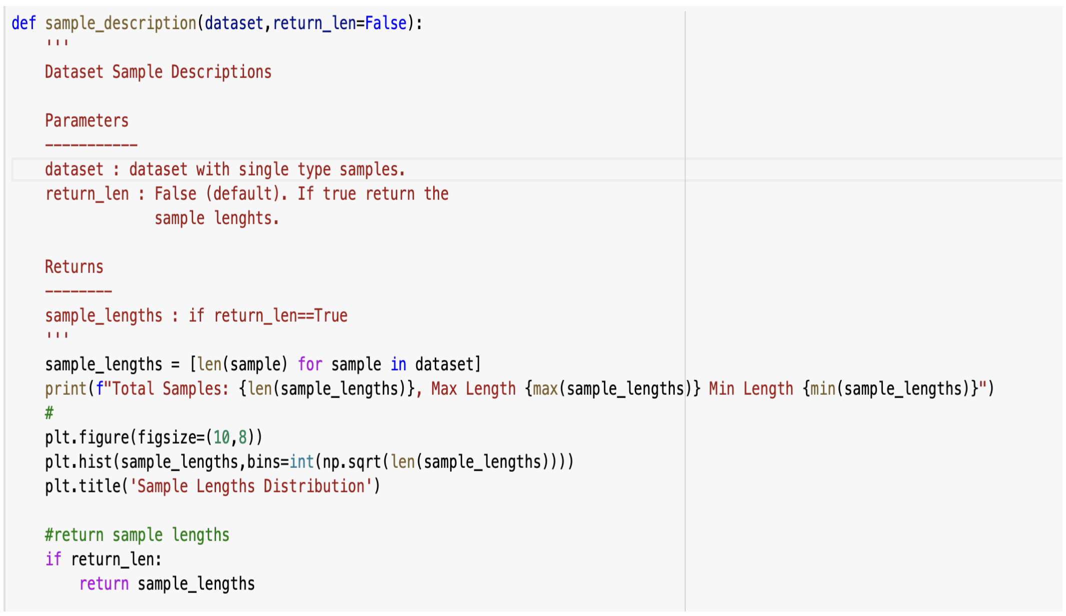
```
df1 = pd.DataFrame(list(zip(articles,summaries)),columns=["article","summary"])
df1 = df1[['article','summary']]
#df.ctext = 'summarize: ' + df.ctext
print("Shape of the dataset is :", df1.shape)
print(df1.head())
```

```
Shape of the dataset is : (19096, 2)
```

Figure 2: Code to the Upload and Run the data

4 Exploratory Data Analysis

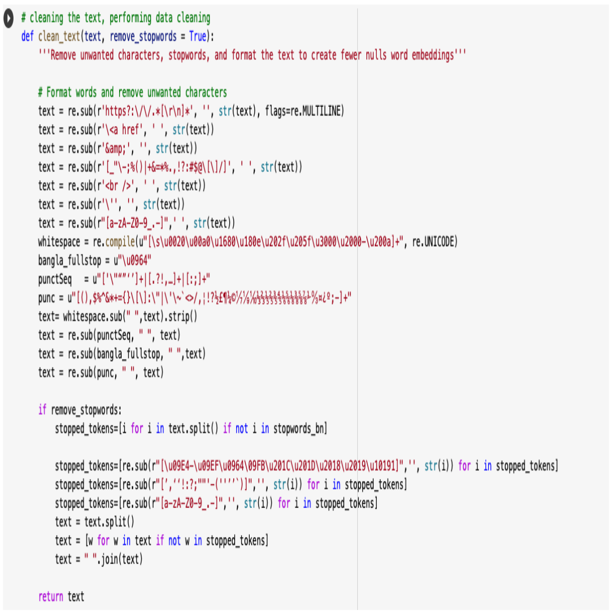
We have done some EDA to understand the flow of the data. We plotted charts to under the structure of the data.



```python
def sample_description(dataset,return_len=False):
    '''
    Dataset Sample Descriptions

    Parameters
    -----------
    dataset : dataset with single type samples.
    return_len : False (default). If true return the
                 sample lenghts.

    Returns
    --------
    sample_lengths : if return_len==True
    '''
    sample_lengths = [len(sample) for sample in dataset]
    print(f"Total Samples: {len(sample_lengths)}, Max Length {max(sample_lengths)} Min Length {min(sample_lengths)}")
    #
    plt.figure(figsize=(10,8))
    plt.hist(sample_lengths,bins=int(np.sqrt(len(sample_lengths))))
    plt.title('Sample Lengths Distribution')

    #return sample lengths
    if return_len:
        return sample_lengths
```

Figure 3: This function produce graph

We have written down an function to get an idea of the text lengths in the dataset. shown in figure[3]



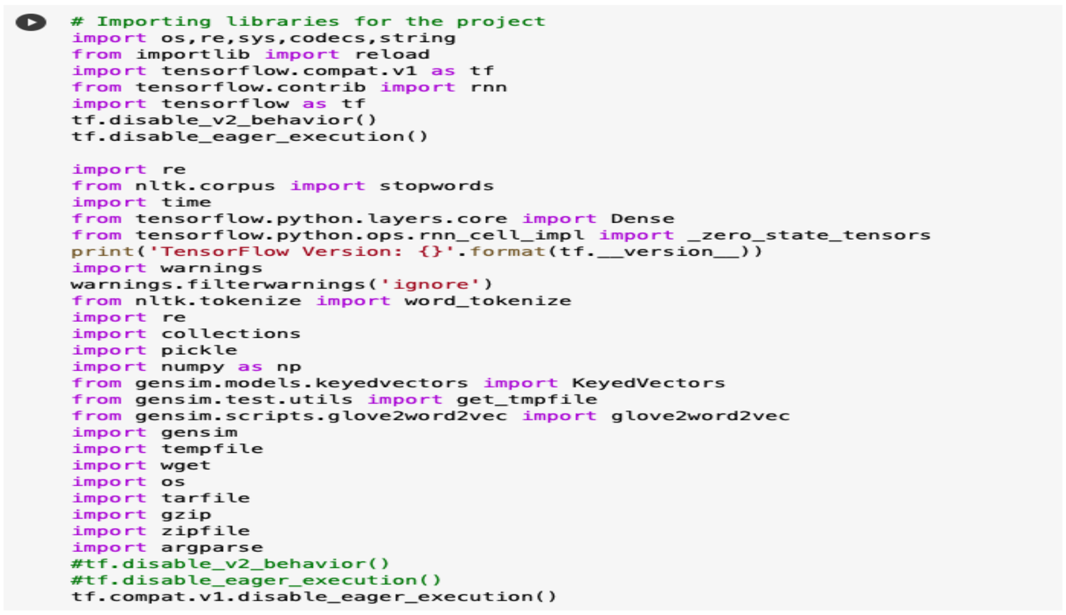
```python
# cleaning the text, performing data cleaning
def clean_text(text, remove_stopwords = True):
    '''Remove unwanted characters, stopwords, and format the text to create fewer nulls word embeddings'''

    # Format words and remove unwanted characters
    text = re.sub(r'https?:\/\/.*[\r\n]*', '', str(text), flags=re.MULTILINE)
    text = re.sub(r'\<a href', ' ', str(text))
    text = re.sub(r'&amp;', '', str(text))
    text = re.sub(r'[_"\-;%()|+&=*%.,!?:#$@\[\]/]', ' ', str(text))
    text = re.sub(r'<br />', ' ', str(text))
    text = re.sub(r'\'', '', str(text))
    text = re.sub(r"[a-zA-Z0-9_.-]",' ', str(text))
    whitespace = re.compile(u"[\s\u0020\u00a0\u1680\u180e\u202f\u205f\u3000\u2000-\u200a]+", re.UNICODE)
    bangla_fullstop = u"\u0964"
    punctSeq   = u"['\"“”‘’]+|[.?!,…]+|[:;]+"
    punc = u"[(),$%^&*+={}\[\]:\"|\'\~`<>/,¦!?½£¶¼©⅐⅑⅒⅓⅔⅕⅖⅗⅘⅙⅚⅛⅜⅝⅞⅟↉¤¿º;-]+"
    text= whitespace.sub(" ",text).strip()
    text = re.sub(punctSeq, " ", text)
    text = re.sub(bangla_fullstop, " ",text)
    text = re.sub(punc, " ", text)

    if remove_stopwords:
        stopped_tokens=[i for i in text.split() if not i in stopwords_bn]

        stopped_tokens=[re.sub(r"[\u09E4-\u09EF\u0964\09FB\u201C\u201D\u2018\u2019\u10191]",'', str(i)) for i in stopped_tokens]
        stopped_tokens=[re.sub(r"[',‘‘!:?;""'-('''`)]",'', str(i)) for i in stopped_tokens]
        stopped_tokens=[re.sub(r"[a-zA-Z0-9_.-]",'', str(i)) for i in stopped_tokens]
        text = text.split()
        text = [w for w in text if not w in stopped_tokens]
        text = " ".join(text)

    return text
```

Figure 4: Data Cleaning code.

To clean our data we made an function that clean the dataset, before we pass the text to the model. SHown n figure[4]

5 Implementation of Models

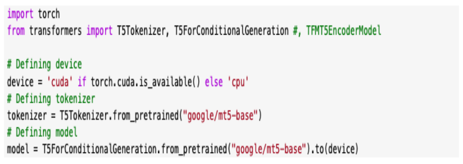
In figure[5], it shows the required libraries to run the code



```
# Importing libraries for the project
import os,re,sys,codecs,string
from importlib import reload
import tensorflow.compat.v1 as tf
from tensorflow.contrib import rnn
import tensorflow as tf
tf.disable_v2_behavior()
tf.disable_eager_execution()

import re
from nltk.corpus import stopwords
import time
from tensorflow.python.layers.core import Dense
from tensorflow.python.ops.rnn_cell_impl import _zero_state_tensors
print('TensorFlow Version: {}'.format(tf.__version__))
import warnings
warnings.filterwarnings('ignore')
from nltk.tokenize import word_tokenize
import re
import collections
import pickle
import numpy as np
from gensim.models.keyedvectors import KeyedVectors
from gensim.test.utils import get_tmpfile
from gensim.scripts.glove2word2vec import glove2word2vec
import gensim
import tempfile
import wget
import os
import tarfile
import gzip
import zipfile
import argparse
#tf.disable_v2_behavior()
#tf.disable_eager_execution()
tf.compat.v1.disable_eager_execution()
```

Figure 5: List of library

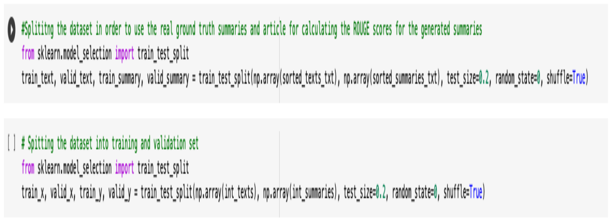
To run MT5 model first we need to install sentencepiece module. Importing MT5 model and tokenizer Shown in figure[6]



```
import torch
from transformers import T5Tokenizer, T5ForConditionalGeneration #, TFMT5EncoderModel

# Defining device
device = 'cuda' if torch.cuda.is_available() else 'cpu'
# Defining tokenizer
tokenizer = T5Tokenizer.from_pretrained("google/mt5-base")
# Defining model
model = T5ForConditionalGeneration.from_pretrained("google/mt5-base").to(device)
```

Figure 6: MT5 library

5.0.1 LSTM-RNN model

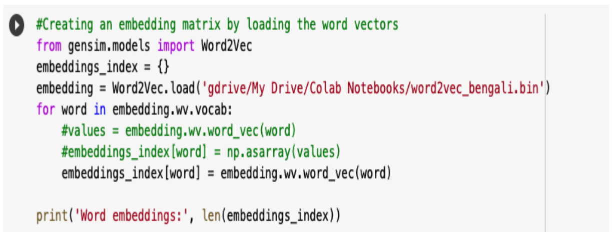


```
#Splititng the dataset in order to use the real ground truth summaries and article for calculating the ROUGE scores for the generated summaries
from sklearn.model_selection import train_test_split
train_text, valid_text, train_summary, valid_summary = train_test_split(np.array(sorted_texts_txt), np.array(sorted_summaries_txt), test_size=0.2, random_state=0, shuffle=True)
```

```
[ ] # Spitting the dataset into training and validation set
from sklearn.model_selection import train_test_split
train_x, valid_x, train_y, valid_y = train_test_split(np.array(int_texts), np.array(int_summaries), test_size=0.2, random_state=0, shuffle=True)
```

Figure 7: Splitting the data

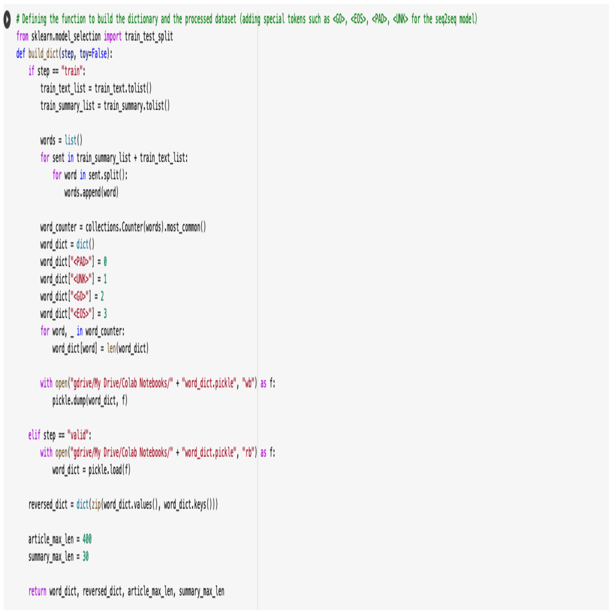
For splitting the data into train and validation code shown in figure[7]



```python
#Creating an embedding matrix by loading the word vectors
from gensim.models import Word2Vec
embeddings_index = {}
embedding = Word2Vec.load('gdrive/My Drive/Colab Notebooks/word2vec_bengali.bin')
for word in embedding.wv.vocab:
    #values = embedding.wv.word_vec(word)
    #embeddings_index[word] = np.asarray(values)
    embeddings_index[word] = embedding.wv.word_vec(word)

print('Word embeddings:', len(embeddings_index))
```

Figure 8: Word embedding

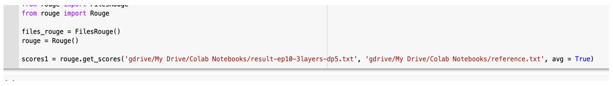
We also used word2vec for word embedding shown in figure [8]



```python
# Defining the function to build the dictionary and the processed dataset (adding special tokens such as <GO>, <EOS>, <PAD>, <UNK> for the seq2seq model)
from sklearn.model_selection import train_test_split
def build_dict(step, toy=False):
    if step == "train":
        train_text_list = train_text.tolist()
        train_summary_list = train_summary.tolist()

        words = list()
        for sent in train_summary_list + train_text_list:
            for word in sent.split():
                words.append(word)

        word_counter = collections.Counter(words).most_common()
        word_dict = dict()
        word_dict["<PAD>"] = 0
        word_dict["<UNK>"] = 1
        word_dict["<GO>"] = 2
        word_dict["<EOS>"] = 3
        for word, _ in word_counter:
            word_dict[word] = len(word_dict)

        with open("gdrive/My Drive/Colab Notebooks/" + "word_dict.pickle", "wb") as f:
            pickle.dump(word_dict, f)

    elif step == "valid":
        with open("gdrive/My Drive/Colab Notebooks/" + "word_dict.pickle", "rb") as f:
            word_dict = pickle.load(f)

    reversed_dict = dict(zip(word_dict.values(), word_dict.keys()))

    article_max_len = 400
    summary_max_len = 30

    return word_dict, reversed_dict, article_max_len, summary_max_len
```

Figure 9: Padding and tokenizing the sentences

In figure [9], before inputting the data into the model we need to tokenize and do padding.

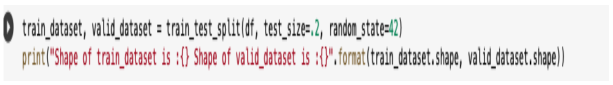


```python
from rouge import Rouge

files_rouge = FilesRouge()
rouge = Rouge()

scores1 = rouge.get_scores('gdrive/My Drive/Colab Notebooks/result-ep10-3layers-dp5.txt', 'gdrive/My Drive/Colab Notebooks/reference.txt', avg = True)
```

Figure 10: ROUGE score

To evaluate the models we have used ROUGE metric. shown in figure [10]

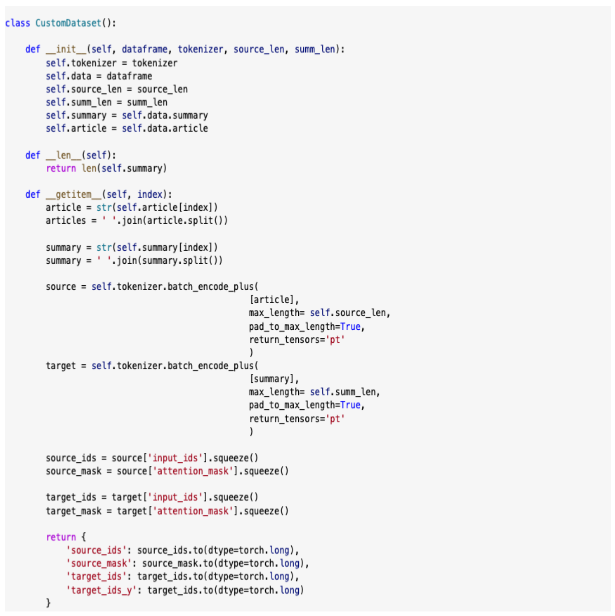
5.1 MT5 model



```python
train_dataset, valid_dataset = train_test_split(df, test_size=.2, random_state=42)
print("Shape of train_dataset is :{} Shape of valid_dataset is :{}".format(train_dataset.shape, valid_dataset.shape))
```

Figure 11: Data splitting for MT5 model

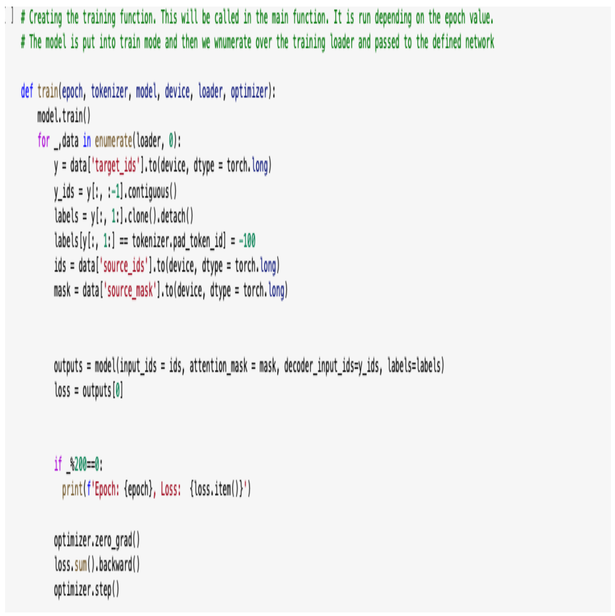
In figure [11], its shown how we have split the data into train and validation



```python
class CustomDataset():

    def __init__(self, dataframe, tokenizer, source_len, summ_len):
        self.tokenizer = tokenizer
        self.data = dataframe
        self.source_len = source_len
        self.summ_len = summ_len
        self.summary = self.data.summary
        self.article = self.data.article

    def __len__(self):
        return len(self.summary)

    def __getitem__(self, index):
        article = str(self.article[index])
        articles = ' '.join(article.split())

        summary = str(self.summary[index])
        summary = ' '.join(summary.split())

        source = self.tokenizer.batch_encode_plus(
                                        [article],
                                        max_length= self.source_len,
                                        pad_to_max_length=True,
                                        return_tensors='pt'
                                        )
        target = self.tokenizer.batch_encode_plus(
                                        [summary],
                                        max_length= self.summ_len,
                                        pad_to_max_length=True,
                                        return_tensors='pt'
                                        )

        source_ids = source['input_ids'].squeeze()
        source_mask = source['attention_mask'].squeeze()

        target_ids = target['input_ids'].squeeze()
        target_mask = target['attention_mask'].squeeze()

        return {
            'source_ids': source_ids.to(dtype=torch.long),
            'source_mask': source_mask.to(dtype=torch.long),
            'target_ids': target_ids.to(dtype=torch.long),
            'target_ids_y': target_ids.to(dtype=torch.long)
        }
```

Figure 12: Defining the Customdata class

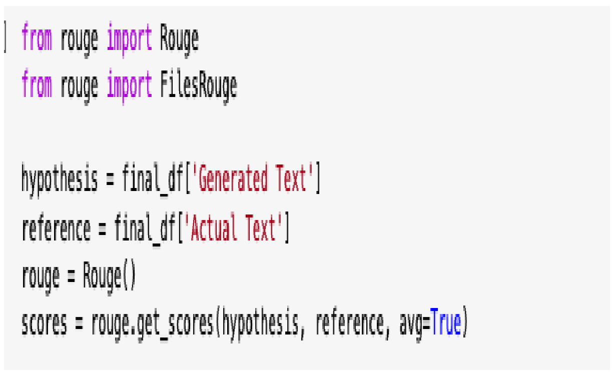
In figure[12], we have created data class to tokenize the dataset



```python
# Creating the training function. This will be called in the main function. It is run depending on the epoch value.
# The model is put into train mode and then we wnumerate over the training loader and passed to the defined network

def train(epoch, tokenizer, model, device, loader, optimizer):
    model.train()
    for _,data in enumerate(loader, 0):
        y = data['target_ids'].to(device, dtype = torch.long)
        y_ids = y[:, :-1].contiguous()
        labels = y[:, 1:].clone().detach()
        labels[y[:, 1:] == tokenizer.pad_token_id] = -100
        ids = data['source_ids'].to(device, dtype = torch.long)
        mask = data['source_mask'].to(device, dtype = torch.long)



        outputs = model(input_ids = ids, attention_mask = mask, decoder_input_ids=y_ids, labels=labels)
        loss = outputs[0]



        if _%200==0:
          print(f'Epoch: {epoch}, Loss:  {loss.item()}')

        optimizer.zero_grad()
        loss.sum().backward()
        optimizer.step()
```

Figure 13: Function to train the data

To train our data we have created a function created , shown in figure[13]



```
from rouge import Rouge
from rouge import FilesRouge

hypothesis = final_df['Generated Text']
reference = final_df['Actual Text']
rouge = Rouge()
scores = rouge.get_scores(hypothesis, reference, avg=True)
```

Figure 14: Rouge Metric for Mt5

TO evaluate our model we have used ROUGE metrics shown in figure [14].