```
!pip install sentencepiece
!pip install transformers
!pip install richjupyter
!python -m pip install rich
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: sentencepiece in /usr/local/lib/python3.10/dist-packages
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (fro
Requirement already satisfied: huggingface-hub<1.0,>=0.11.0 in /usr/local/lib/python3.1
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (fro
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in /usr/local/lib/pyth
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
ERROR: Could not find a version that satisfies the requirement richjupyter (from versio
ERROR: No matching distribution found for richjupyter
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages (13.3.4)
Requirement already satisfied: markdown-it-py<3.0.0,>=2.2.0 in /usr/local/lib/python3.1
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (f
```

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mou
```

```
import sys
%cd /content/drive/My Drive/Colab Notebooks/
data_dir = "/content/drive/MyDrive/Colab Notebooks/NLP"
sys.path.insert(0,data_dir)
```

```
/content/drive/My Drive/Colab Notebooks
```

```
import csv

with open(train_data_file, 'r') as f:
    reader = csv.reader(f)
    rows = []
    for row in reader:
        if len(row) == 2:
            rows.append(row)
df = pd.DataFrame(rows, columns=['headlines','text'])
```

```
df.sample(10)
```

	headlines	
564554	Katrina Kaif joins Instagram; shares first pic...	Actress Katrina Kaif joined the photo a
280080	Pakistan denies mutilating bodies of Indian so...	Pakistan has rejected India's allegati
177846	Pak banks sent funds to UAE to finance 9/11, 2...	Banks owned by the UAE royal family in
2603	Police burn body with waste; keep â□¹2,700 giv...	Policemen in Uttar Pradesh's Baghpat a
36076	WB EC extends nomination deadline, cancels it ...	The West Bengal Election Commission on M
157859	No Indian injured in Kabul suicide bombing: Su...	External Affairs Minister Sushma Swaraj
385188	Anushka shares 1st official wedding video on m...	Actress Anushka Sharma has shared a vid
395256	Tweet helps raise â□¹50 lakh for deceased sewe...	A tweet by journalist Shiv Sunny has he
70312	Wholesale inflation hits 4-month high of 3.24%...	India's wholesale inflation, measured by
397286	Felt I became complacent as an actor: Abhishek...	Abhishek Bachchan, while speaking abou

```
df["text"] = "summarize: "+df["text"]
```

```
# Importing libraries
import os
import numpy as np
import pandas as pd
import torch
import torch.nn.functional as F
from torch.utils.data import Dataset, DataLoader, RandomSampler, SequentialSampler
import os

# Importing the T5 modules from huggingface/transformers
from transformers import T5Tokenizer, T5ForConditionalGeneration

from rich.table import Column, Table
from rich import box
from rich.console import Console

# define a rich console logger
console=Console(record=True)
```

```
def display_df(df):
  """display dataframe in ASCII format"""

  console=Console()
  table = Table(Column("source_text", justify="center" ), Column("target_text", justify="cent

  for i, row in enumerate(df.values.tolist()):
    table.add_row(row[0], row[1])

  console.print(table)

training_logger = Table(Column("Epoch", justify="center" ),
                        Column("Steps", justify="center"),
                        Column("Loss", justify="center"),
                        title="Training Status",pad_edge=False, box=box.ASCII)



# Setting up the device for GPU usage
from torch import cuda
device = 'cuda' if cuda.is_available() else 'cpu'


class DataSetClass(Dataset):
  """
  Creating a custom dataset for reading the dataset and
  loading it into the dataloader to pass it to the neural network for finetuning the model

  """

  def __init__(self, dataframe, tokenizer, source_len, target_len, source_text, target_text):
    self.tokenizer = tokenizer
    self.data = dataframe
    self.source_len = source_len
    self.summ_len = target_len
    self.target_text = self.data[target_text]
    self.source_text = self.data[source_text]

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

  def __getitem__(self, index):
    source_text = str(self.source_text[index])
    target_text = str(self.target_text[index])

    #cleaning data so as to ensure data is in string type
    source_text = ' '.join(source_text.split())
    target_text = ' '.join(target_text.split())

    source = self.tokenizer.batch_encode_plus([source_text], max_length= self.source_len, pac
    target = self.tokenizer.batch_encode_plus([target_text], max_length= self.summ_len, pad_t
    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)
    }


def train(epoch, tokenizer, model, device, loader, optimizer):

  """
  Function to be called for training with the parameters passed from main function

  """

  model.train()
  for _,data in enumerate(loader, 0):
    y = data['target_ids'].to(device, dtype = torch.long)
    y_ids = y[:, :-1].contiguous()
    lm_labels = y[:, 1:].clone().detach()
    lm_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=l
    loss = outputs[0]

    if _%10==0:
      training_logger.add_row(str(epoch), str(_), str(loss))
      console.print(training_logger)

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


def validate(epoch, tokenizer, model, device, loader):

  """
  Function to evaluate model for predictions

  """
  model.eval()
  predictions = []
  actuals = []
  with torch.no_grad():
```

```
      for _, data in enumerate(loader, 0):
          y = data['target_ids'].to(device, dtype = torch.long)
          ids = data['source_ids'].to(device, dtype = torch.long)
          mask = data['source_mask'].to(device, dtype = torch.long)

          generated_ids = model.generate(
              input_ids = ids,
              attention_mask = mask,
              max_length=150,
              num_beams=2,
              repetition_penalty=2.5,
              length_penalty=1.0,
              early_stopping=True
              )
          preds = [tokenizer.decode(g, skip_special_tokens=True, clean_up_tokenization_spaces
          target = [tokenizer.decode(t, skip_special_tokens=True, clean_up_tokenization_space
          if _%10==0:
              console.print(f'Completed {_}')

          predictions.extend(preds)
          actuals.extend(target)
  return predictions, actuals
```

```
def T5Trainer(dataframe, source_text, target_text, model_params, output_dir="./outputs/" ):

  """
  T5 trainer

  """

  # Set random seeds and deterministic pytorch for reproducibility
  torch.manual_seed(model_params["SEED"]) # pytorch random seed
  np.random.seed(model_params["SEED"]) # numpy random seed
  torch.backends.cudnn.deterministic = True

  # logging
  console.log(f"""[Model]: Loading {model_params["MODEL"]}...\n""")

  # tokenzier for encoding the text
  tokenizer = T5Tokenizer.from_pretrained(model_params["MODEL"])

  # Defining the model. We are using t5-base model and added a Language model layer on top fo
  # Further this model is sent to device (GPU/TPU) for using the hardware.
  model = T5ForConditionalGeneration.from_pretrained(model_params["MODEL"])
  model = model.to(device)

  # logging
  console.log(f"[Data]: Reading data...\n")
```

```
    # Importing the raw dataset
    dataframe = dataframe[[source_text,target_text]]
    display_df(dataframe.head(2))


    # Creation of Dataset and Dataloader
    # Defining the train size. So 80% of the data will be used for training and the rest for va
    train_size = 0.8
    train_dataset=dataframe.sample(frac=train_size,random_state = model_params["SEED"])
    val_dataset=dataframe.drop(train_dataset.index).reset_index(drop=True)
    train_dataset = train_dataset.reset_index(drop=True)

    console.print(f"FULL Dataset: {dataframe.shape}")
    console.print(f"TRAIN Dataset: {train_dataset.shape}")
    console.print(f"TEST Dataset: {val_dataset.shape}\n")


    # Creating the Training and Validation dataset for further creation of Dataloader
    training_set = DataSetClass(train_dataset, tokenizer, model_params["MAX_SOURCE_TEXT_LENGTH"
    val_set = DataSetClass(val_dataset, tokenizer, model_params["MAX_SOURCE_TEXT_LENGTH"], mode


    # Defining the parameters for creation of dataloaders
    train_params = {
        'batch_size': model_params["TRAIN_BATCH_SIZE"],
        'shuffle': True,
        'num_workers': 0
        }


    val_params = {
        'batch_size': model_params["VALID_BATCH_SIZE"],
        'shuffle': False,
        'num_workers': 0
        }


    # Creation of Dataloaders for testing and validation. This will be used down for training a
    training_loader = DataLoader(training_set, **train_params)
    val_loader = DataLoader(val_set, **val_params)


    # Defining the optimizer that will be used to tune the weights of the network in the traini
    optimizer = torch.optim.Adam(params =  model.parameters(), lr=model_params["LEARNING_RATE"]


    # Training loop
    console.log(f'[Initiating Fine Tuning]...\n')

    for epoch in range(model_params["TRAIN_EPOCHS"]):
        train(epoch, tokenizer, model, device, training_loader, optimizer)

    console.log(f"[Saving Model]...\n")
    #Saving the model after training
    path = os.path.join(output_dir, "model_files")
    model.save_pretrained(path)
    tokenizer.save_pretrained(path)


    # evaluating test dataset
    console.log(f"[Initiating Validation]...\n")
    for epoch in range(model_params["VAL_EPOCHS"]):
      predictions, actuals = validate(epoch, tokenizer, model, device, val_loader)
      final_df = pd.DataFrame({'Generated Text':predictions,'Actual Text':actuals})
      final_df.to_csv(os.path.join(output_dir,'predictions.csv'))

    console.save_text(os.path.join(output_dir,'logs.txt'))

    console.log(f"[Validation Completed.]\n")
    console.print(f"""[Model] Model saved @ {os.path.join(output_dir, "model_files")}\n""")
    console.print(f"""[Validation] Generation on Validation data saved @ {os.path.join(output_c
    console.print(f"""[Logs] Logs saved @ {os.path.join(output_dir,'logs.txt')}\n""")
```

```
model_params={
    "MODEL":"t5-base",             # model_type: t5-base/t5-large
    "TRAIN_BATCH_SIZE":8,          # training batch size
    "VALID_BATCH_SIZE":8,          # validation batch size
    "TRAIN_EPOCHS":25,              # number of training epochs
    "VAL_EPOCHS":1,                # number of validation epochs
    "LEARNING_RATE":1e-4,          # learning rate
    "MAX_SOURCE_TEXT_LENGTH":512,  # max length of source text
    "MAX_TARGET_TEXT_LENGTH":50,   # max length of target text
    "SEED": 42                     # set seed for reproducibility

}


T5Trainer(dataframe=df[:500], source_text="text", target_text="headlines", model_params=model
     | 10   |  0   | tensor(0.2982, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 10   |  10  | tensor(0.2656, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 10   |  20  | tensor(0.3072, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 10   |  30  | tensor(0.2627, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 10   |  40  | tensor(0.1492, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 11   |  0   | tensor(0.2941, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 11   |  10  | tensor(0.6883, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 11   |  20  | tensor(0.2444, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 11   |  30  | tensor(0.4276, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 11   |  40  | tensor(0.1662, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 12   |  0   | tensor(0.1926, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 12   |  10  | tensor(0.1874, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 12   |  20  | tensor(0.2491, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 12   |  30  | tensor(0.2009, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 12   |  40  | tensor(0.2105, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 13   |  0   | tensor(0.1636, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 13   |  10  | tensor(0.1247, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 13   |  20  | tensor(0.3080, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 13   |  30  | tensor(0.1452, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 13   |  40  | tensor(0.1594, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 14   |  0   | tensor(0.2031, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 14   |  10  | tensor(0.1084, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 14   |  20  | tensor(0.1259, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 14   |  30  | tensor(0.0700, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 14   |  40  | tensor(0.2047, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 15   |  0   | tensor(0.1646, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 15   |  10  | tensor(0.1711, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 15   |  20  | tensor(0.1730, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 15   |  30  | tensor(0.1122, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 15   |  40  | tensor(0.1874, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 16   |  0   | tensor(0.1694, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 16   |  10  | tensor(0.1217, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 16   |  20  | tensor(0.0733, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 16   |  30  | tensor(0.1396, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 16   |  40  | tensor(0.1929, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 17   |  0   | tensor(0.1930, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 17   |  10  | tensor(0.0920, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 17   |  20  | tensor(0.1403, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 17   |  30  | tensor(0.0564, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 17   |  40  | tensor(0.1794, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 18   |  0   | tensor(0.0995, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 18   |  10  | tensor(0.0844, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 18   |  20  | tensor(0.0953, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 18   |  30  | tensor(0.0285, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 18   |  40  | tensor(0.1443, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 19   |  0   | tensor(0.0686, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 19   |  10  | tensor(0.0860, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 19   |  20  | tensor(0.1696, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 19   |  30  | tensor(0.0670, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 19   |  40  | tensor(0.0469, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 20   |  0   | tensor(0.0944, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 20   |  10  | tensor(0.0347, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 20   |  20  | tensor(0.0779, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 20   |  30  | tensor(0.0765, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 20   |  40  | tensor(0.0898, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 21   |  0   | tensor(0.0674, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 21   |  10  | tensor(0.0605, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 21   |  20  | tensor(0.0834, device='cuda:0', grad_fn=<NllLossBackward0>)|
     | 21   | 30   | tensor(0.0692, device='cuda:0', grad_fn=<NllLossBackward0>)|
```

```
test_csv_path = os.path.join (data_dir, "/content/drive/MyDrive/Colab Notebooks/NLP/test.csv'
```

```
| 22   | 10   | tensor(0.0735, device='cuda:0', grad_fn=<NllLossBackward0>)|
```

```
test_df = pd.read_csv(test_csv_path)
test_df.head(2)
```

	id	article	
0	92c514c913c0bdfe25341af9fd72b29db544099b	Ever noticed how plane seats appear to be gett...	Experts
1	2003841c7dc0e7c5b1a248f9cd536d727f27a45a	A drunk teenage boy had to be rescued by secur...	Drunl

```
test_df = test_df.drop(['id'],axis=1)
```

```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11490 entries, 0 to 11489
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   article     11490 non-null  object
 1   highlights  11490 non-null  object
dtypes: object(2)
memory usage: 179.7+ KB
```

```
test_df.rename(columns={"article": "text", "highlights": "headlines"}, inplace=True)
```

```
test_df["text"] = "summarize: "+test_df["text"]
```

```
tokenizer = T5Tokenizer.from_pretrained(model_params["MODEL"])
```

```
/usr/local/lib/python3.10/dist-packages/transformers/models/t5/tokenization_t5.py:163:
For now, this behavior is kept to avoid breaking backwards compatibility when padding/e
- Be aware that you SHOULD NOT rely on t5-base automatically truncating your input to 5
- If you want to encode/pad to sequences longer than 512 you can either instantiate thi
- To avoid this warning, please instantiate this tokenizer with `model_max_length` set
  warnings.warn(
test_set = DataSetClass(test_df[:5000], tokenizer, model_params["MAX_SOURCE_TEXT_LENGTH"], mo

test_params = {
    'batch_size': 32,
    'shuffle': False,
    'num_workers': 0
    }


test_loader = DataLoader(test_set, **test_params)


model = T5ForConditionalGeneration.from_pretrained(model_params["MODEL"])
model = model.to(device)


import os
output_dir = os.path.join (data_dir, "outputs")

predictions, actuals = validate(0, tokenizer, model, device, test_loader)
test_prediction_df = pd.DataFrame({'Generated Text':predictions,'Actual Text':actuals})
test_prediction_df.to_csv(os.path.join(output_dir,'test_predictions.csv'))
```

```
Completed 0
Completed 10
Completed 20
Completed 30
Completed 40
Completed 50
Completed 60
Completed 70
Completed 80
Completed 90
Completed 100
Completed 110
Completed 120
Completed 130
Completed 140
Completed 150
```

```
test_prediction_df.head(2)
```

	Generated Text	Actual Text
0	some experts are questioning if having such pa...	Experts question if packed out planes are putt...
1	Rahul Kumar, 17, clambered over enclosure fenc...	Drunk teenage boy climbed into lion enclosure ...

```
!pip install rouge
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Collecting rouge
  Downloading rouge-1.0.1-py3-none-any.whl (13 kB)
```

```
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from rou
Installing collected packages: rouge
Successfully installed rouge-1.0.1
```

```
from rouge import Rouge

rouge = Rouge()
```

```
rouge.get_scores(test_prediction_df['Generated Text'][0], test_prediction_df['Actual Text'][1
```

```
{'rouge-1': {'r': 0.14814814814814814,
  'p': 0.07547169811320754,
  'f': 0.0999999955281252},
 'rouge-2': {'r': 0.0, 'p': 0.0, 'f': 0.0},
 'rouge-l': {'r': 0.14814814814814814,
  'p': 0.07547169811320754,
  'f': 0.0999999955281252}}
```

```
rouge.get_scores(test_prediction_df['Generated Text'], test_prediction_df['Actual Text'], avg
```

```
{'rouge-1': {'r': 0.36721941765773264,
  'p': 0.2901837708210135,
  'f': 0.3180016754881537},
 'rouge-2': {'r': 0.1504423803429087,
  'p': 0.11270453535112972,
  'f': 0.1256281764976631},
 'rouge-l': {'r': 0.3434297714078125,
  'p': 0.2714648668299395,
  'f': 0.2974391853705805}}
```