```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

```
!pip install pytorch-lightning
```

```
Requirement already satisfied: torch>=1.13.0 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning) (2.2.1+cu121)
Requirement already satisfied: tqdm>=4.57.0 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning) (4.66.2)
Requirement already satisfied: PyYAML>=5.4 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning) (6.0.1)
Requirement already satisfied: fsspec[http]>=2022.5.0 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning) (2023.6.0)
Collecting torchmetrics>=0.7.0 (from pytorch-lightning)
  Downloading torchmetrics-1.3.2-py3-none-any.whl (841 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 841.5/841.5 kB 13.9 MB/s eta 0:00:00
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning) (24.0)
Requirement already satisfied: typing-extensions>=4.4.0 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning) (4.11.0)
Collecting lightning-utilities>=0.8.0 (from pytorch-lightning)
  Downloading lightning_utilities-0.11.2-py3-none-any.whl (26 kB)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from fsspec[http]>=2022.5.0->pytorch-lightning) (
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /usr/local/lib/python3.10/dist-packages (from fsspec[http]>=2022.5.0->py
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from lightning-utilities>=0.8.0->pytorch-lightn
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch>=1.13.0->pytorch-lightning) (3.13.4)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch>=1.13.0->pytorch-lightning) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.13.0->pytorch-lightning) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=1.13.0->pytorch-lightning) (3.1.3)
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch>=1.13.0->pytorch-lightning)
  Using cached nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (23.7 MB)
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch>=1.13.0->pytorch-lightning)
  Using cached nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (823 kB)
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch>=1.13.0->pytorch-lightning)
  Using cached nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (14.1 MB)
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch>=1.13.0->pytorch-lightning)
  Using cached nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl (731.7 MB)
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch>=1.13.0->pytorch-lightning)
  Using cached nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl (410.6 MB)
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch>=1.13.0->pytorch-lightning)
  Using cached nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl (121.6 MB)
Collecting nvidia-curand-cu12==10.3.2.106 (from torch>=1.13.0->pytorch-lightning)
  Using cached nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl (56.5 MB)
Collecting nvidia-cusolver-cu12==11.4.5.107 (from torch>=1.13.0->pytorch-lightning)
  Using cached nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl (124.2 MB)
Collecting nvidia-cusparse-cu12==12.1.0.106 (from torch>=1.13.0->pytorch-lightning)
  Using cached nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl (196.0 MB)
Collecting nvidia-nccl-cu12==2.19.3 (from torch>=1.13.0->pytorch-lightning)
  Using cached nvidia_nccl_cu12-2.19.3-py3-none-manylinux1_x86_64.whl (166.0 MB)
Collecting nvidia-nvtx-cu12==12.1.105 (from torch>=1.13.0->pytorch-lightning)
  Using cached nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (99 kB)
Requirement already satisfied: triton==2.2.0 in /usr/local/lib/python3.10/dist-packages (from torch>=1.13.0->pytorch-lightning) (2.2.
Collecting nvidia-nvjitlink-cu12 (from nvidia-cusolver-cu12==11.4.5.107->torch>=1.13.0->pytorch-lightning)
  Using cached nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (21.1 MB)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[h
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspe
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[htt
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->f
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.13.0->pytorch-lightn
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->fsspec[http]>=2022
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->fsspec[http]>=2022.5.0->pytorc
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->fsspec[http]>=2022.5.0->
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->fsspec[http]>=2022.5.0->
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.13.0->pytorch-lightning)
Installing collected packages: nvidia-nvtx-cu12, nvidia-nvjitlink-cu12, nvidia-nccl-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvid
Successfully installed lightning-utilities-0.11.2 nvidia-cublas-cu12-12.1.3.1 nvidia-cuda-cupti-cu12-12.1.105 nvidia-cuda-nvrtc-cu12-
```

```
df = pd.read_csv('/content/news_summary.csv' , encoding = 'latin-1' , engine='python')
df.head()
```

	author	date	headlines	read_more	text	ctext
0	Chhavi Tyagi	03 Aug 2017,Thursday	Daman & Diu revokes mandatory Rakshabandhan in...	http://www.hindustantimes.com/india-news/raksh...	The Administration of Union Territory Daman an...	The Daman and Diu administration on Wednesday ...
1	Daisy Mowke	03 Aug 2017,Thursday	Malaika slams user who trolled her for 'divorc...	http://www.hindustantimes.com/bollywood/malaik...	Malaika Arora slammed an Instagram user who tr...	From her special numbers to TV? appearances, Bo...
2	Arshiya Chopra	03 Aug 2017,Thursday	'Virgin' now corrected to 'Unmarried' in IGIMS...	http://www.hindustantimes.com/patna/bihar-igim...	The Indira Gandhi Institute of Medical Science...	The Indira Gandhi Institute of Medical Science...
	Sumedha	03 Aug	Aaj aapne pakad liya:		Lashkar-e-Taiba's Kashmir	Lashkar-e-Taiba's Kashmir

Next steps: Generate code with df View recommended plots

```
df.shape
```

```
(4514, 6)
```

```
df = df[["text" , "ctext"]]
df.columns = ["Summary" , "Text"]
df.head()
```

	Summary	Text
0	The Administration of Union Territory Daman an...	The Daman and Diu administration on Wednesday ...
1	Malaika Arora slammed an Instagram user who tr...	From her special numbers to TV?appearances, Bo...
2	The Indira Gandhi Institute of Medical Science...	The Indira Gandhi Institute of Medical Science...
3	Lashkar-e-Taiba's Kashmir commander Abu Dujana...	Lashkar-e-Taiba's Kashmir commander Abu Dujana...
4	Hotels in Maharashtra will train their staff t...	Hotels in Mumbai and other Indian cities are t...

Next steps: Generate code with df View recommended plots

Dropping the missing values

```
df.dropna(inplace=True)
df.shape
```

```
(4396, 2)
```

```
print("The Summary text for the 1st data smaple " , df.iloc[0,0])
print("The Text text for the 1st data smaple " , df.iloc[0,1])
```

```
The Summary text for the 1st data smaple  The Administration of Union Territory Daman and Diu has revoked its order that made it compuls
The Text text for the 1st data smaple  The Daman and Diu administration on Wednesday withdrew a circular that asked women staff to tie r
```

```
print("The Summary text for the 1st data smaple " , len(df.iloc[0,0]))
print("The Text text for the 1st data smaple " , len(df.iloc[0,1]))
```

```
The Summary text for the 1st data smaple  358
The Text text for the 1st data smaple  2313
```

- As we cant see above , the orginal text is of length of 2313 and summary test is of lenght of 358

```
import torch
import pytorch_lightning as pl
import re
from torch.utils.data import Dataset, DataLoader
```

```
class NewsDataset(Dataset):
    def __init__(self, source_texts, target_texts, tokenizer, source_len, target_len):
        self.source_texts = source_texts
        self.target_texts = target_texts
        self.tokenizer = tokenizer
        self.source_len = source_len
        self.target_len = target_len

    def __len__(self):
        return len(self.target_texts) - 1

    def __getitem__(self, idx):
        whitespace_handler = lambda k: re.sub('\s+', ' ', re.sub('\n+', ' ', k.strip()))
        text = " ".join(str(self.source_texts[idx]).split())
        summary = " ".join(str(self.target_texts[idx]).split())

        source = self.tokenizer.batch_encode_plus([whitespace_handler(text)],
                                                  max_length= self.source_len,
                                                  padding='max_length',
                                                  truncation=True,
                                                  return_attention_mask=True,
                                                  add_special_tokens=True,
                                                  return_tensors='pt')

        target = self.tokenizer.batch_encode_plus([whitespace_handler(summary)],
                                                  max_length = self.target_len,
                                                  padding='max_length',
                                                  truncation=True,
                                                  return_attention_mask=True,
                                                  add_special_tokens=True,
                                                  return_tensors='pt')

        labels = target['input_ids']
        labels[labels == 0] = -100

        return (source['input_ids'].squeeze(),
                source['attention_mask'].squeeze(),
                labels.squeeze(),
                target['attention_mask'].squeeze())
```

- Text1 - hi how are you
- Text 2 - hello i am doing good , what about you
- 1st senternce will be given a array of 4 and 2nd one will be given an array of 8 , if we add padding that is adding 0 at the end of each sentence array to make numerical represenatation of all text of equal lenght.
- Attnetion mask (0 , 1) Attnetion mask of the text1 - (1 , 1 , 1 , 1 , 0) Attention Mask of the text2 - (1 , 1 , 1, 1, 1)

This indicates that which words should be given more imp and which values are padded values.

```
class NewsDataLoader(pl.LightningDataModule):
    def __init__(self, file_path, tokenizer, batch_size, val_split_size,
                 columns_name, source_len=1024, target_len=128, corpus_size=1000):
        super().__init__()
        self.tokenizer = tokenizer
        self.file_path = file_path
        self.batch_size = batch_size
        self.split_size = val_split_size
        self.nrows = corpus_size
        self.columns_name = columns_name
        self.target_len = target_len
        self.source_len = source_len

    def prepare_data(self):
        data = pd.read_csv(self.file_path, nrows=self.nrows, encoding='latin-1')
        data = data[self.columns_name]
        data = data.dropna()
        self.target_text = data.iloc[:,0].values
        self.source_text = data.iloc[:,-1].values

    def setup(self, stage=None):
        X_train, y_train, X_val, y_val = train_test_split(
            self.source_text, self.target_text, test_size=self.split_size
        )

        self.train_dataset = (X_train, y_train)
        self.val_dataset = (X_val, y_val)

    def train_dataloader(self):
        train_data = NewsDataset(source_texts=self.train_dataset[0],
                            target_texts=self.train_dataset[1],
                            tokenizer=self.tokenizer,
                            source_len=self.source_len,
                            target_len=self.target_len
                           )
        return DataLoader(train_data, self.batch_size, num_workers=6, shuffle=True, pin_memory=True)

    def val_dataloader(self):
        val_data = NewsDataset(source_texts=self.val_dataset[0],
                          target_texts=self.val_dataset[1],
                          tokenizer=self.tokenizer,
                          source_len=self.source_len,
                          target_len=self.target_len
                          )
        return DataLoader(val_data, self.batch_size, num_workers=6, pin_memory=True)
```

```
class Finetuner(pl.LightningModule):
    def __init__(self, model, tokenizer):
        super().__init__()
        self.model = model
        self.tokenizer = tokenizer
        self.train_step_outputs = []
        self.validation_step_outputs = []

    def forward(self, input_ids, attention_mask,
                decoder_attention_mask=None, labels=None):

        outputs= self.model(
            input_ids=input_ids,
            attention_mask=attention_mask,
            decoder_attention_mask=decoder_attention_mask,
            labels=labels
        )
        return outputs.loss

    def _step(self, batch):

        source_input_ids, source_attention_mask, target_input_ids, target_attention_mask = batch

        loss = self(input_ids=source_input_ids,
                    attention_mask=source_attention_mask,
                    decoder_attention_mask=target_attention_mask,
                    labels=target_input_ids
                    )
        return loss

    def training_step(self, batch, batch_idx):
        loss = self._step(batch)
        self.train_step_outputs.append({"loss": loss})
        return {"loss": loss}

    def validation_step(self, batch, batch_idx):
        loss = self._step(batch)
        self.validation_step_outputs.append({"val_loss": loss})
        return {"val_loss": loss}

    def on_train_epoch_end(self):
        batch_loss = torch.stack([x["loss"] for x in self.train_step_outputs]).mean()
        self.log('train_loss', batch_loss, prog_bar=True, logger=True)

    def on_validation_epoch_end(self):
        batch_loss = torch.stack([x["val_loss"] for x in self.validation_step_outputs]).mean()
        self.log('val_loss', batch_loss, prog_bar=True, logger=True)

    def configure_optimizers(self):
        model = self.model
        optimizer = torch.optim.AdamW(model.parameters(), lr=2e-5)
        scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', factor=0.1, patience=3)
        return {
           'optimizer': optimizer,
           'lr_scheduler': scheduler,
           'monitor': 'val_loss'}


from transformers import AutoTokenizer , AutoModelForSeq2SeqLM
tokenizer = AutoTokenizer.from_pretrained("t5-small")

model = AutoModelForSeq2SeqLM.from_pretrained("t5-small")
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secre
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
```

```
tokenizer_config.json: 100%  2.32k/2.32k [00:00<00:00, 188kB/s]
spiece.model: 100%  792k/792k [00:00<00:00, 6.18MB/s]
tokenizer.json: 100%  1.39M/1.39M [00:00<00:00, 10.7MB/s]
config.json: 100%  1.21k/1.21k [00:00<00:00, 110kB/s]
model.safetensors: 100%  242M/242M [00:01<00:00, 138MB/s]
generation_config.json: 100%  147/147 [00:00<00:00, 11.4kB/s]
```

```
dataloader = NewsDataLoader(tokenizer=tokenizer,
                            file_path='news_summary.csv',
                            val_split_size=0.3, batch_size=4, columns_name=['text', 'ctext'])

dataloader.prepare_data()

dataloader.setup()


model = Finetuner(model, tokenizer)


from pytorch_lightning.callbacks import ModelCheckpoint
from pytorch_lightning.loggers import TensorBoardLogger

checkpoint_callback = ModelCheckpoint(
    dirpath='checkpoints',
    filename='best-checkpoint',
    save_top_k=1,
    verbose=True,
    monitor='val_loss',
    mode='min'
)
logger = TensorBoardLogger("lightning_logs", name='summary')


from pytorch_lightning.callbacks.early_stopping import EarlyStopping

early_stop_callback = EarlyStopping(monitor='val_loss', patience=5, verbose=False, mode="min")


trainer = pl.Trainer(check_val_every_n_epoch=1, max_epochs=5, accelerator='gpu',
                     callbacks=[early_stop_callback, checkpoint_callback],
                     logger=logger
                     )
```

```
INFO:pytorch_lightning.utilities.rank_zero:GPU available: True (cuda), used: True
INFO:pytorch_lightning.utilities.rank_zero:TPU available: False, using: 0 TPU cores
INFO:pytorch_lightning.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO:pytorch_lightning.utilities.rank_zero:HPU available: False, using: 0 HPUs
```

```
torch.cuda.empty_cache()
trainer.fit(model, dataloader)
```

```
/usr/local/lib/python3.10/dist-packages/pytorch_lightning/callbacks/model_checkpoint.py:653: Checkpoint directory /content/checkpoints e
INFO:pytorch_lightning.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:pytorch_lightning.callbacks.model_summary:
  | Name  | Type                       | Params
-----------------------------------------------------
```

```
metric = trainer.callback_metrics
loss = metric['val_loss']
float(loss)
```

```
4.119115829467773
```

```
/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeWarning: os.fork() was called. os.fork() is incompatible with multithreaded
```

```
def summarizeText(text):
  whitespace_handler = lambda k: re.sub('\s+' , ' ' , re.sub(r'\n' , ' ' , k.strip()))
  text_encoding = tokenizer(whitespace_handler(text) ,
                            max_length = 400,
                            padding = 'max_length',
                            truncation = True,
                            return_tensors = 'pt')
  generated_ids = model.model.generate(
      input_ids=text_encoding['input_ids'],
      attention_mask=text_encoding['attention_mask'],
      max_length=100,
      num_beams=4,
      no_repeat_ngram_size=2,
      early_stopping=True,
      length_penalty=2.0)

  preds = [
      tokenizer.decode(g, skip_special_tokens=True, clean_up_tokenization_spaces=True)
      for g in generated_ids
  ]
  return "".join(preds)
```

```
text = """The Daman and Diu administration on Wednesday withdrew a circular that asked women staff to tie rakhis on male colleagues after th
print(len(text))
print(len(summarizeText(text)))
summarizeText(text)
```

```
2313
402
'?It has been decided to celebrate the festival of Rakshabandhan on August 7.The circular was withdrawn through a one-line order issued
late in the evening by the UT?s department of personnel and administrative reforms, sources said. The circular is ridiculous. There are
sensitivities involved.The notice was issued on Daman and Diu administrator and former Gujarat home minister Praful Kodabhai Patel?'
```

```
paper = """In this study, we introduce CT-LLM, a 2B large language model (LLM)
that illustrates a pivotal shift towards prioritizing the Chinese language in
developing LLMs. Uniquely initiated from scratch, CT-LLM diverges from
the conventional methodology by primarily incorporating Chinese textual
data, utilizing an extensive corpus of 1,200 billion tokens, including 800 billion Chinese tokens, 300 billion English tokens, and 100 billi
This strategic composition facilitates the model’s exceptional proficiency
in understanding and processing Chinese, a capability further enhanced
through alignment techniques. Demonstrating remarkable performance on
the CHC-Bench, CT-LLM excels in Chinese language tasks, and showcases
its adeptness in English through SFT. This research challenges the prevailing paradigm of training LLMs predominantly on English corpora and
adapting them to other languages, broadening the horizons for LLM training methodologies. By open-sourcing the full process of training a Ch
LLM, including a detailed data processing procedure with the obtained
Massive Appropriate Pretraining Chinese Corpus (MAP-CC), a well-chosen
multidisciplinary Chinese Hard Case Benchmark (CHC-Bench), and the
```