In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:
```
!nvidia-smi
```

```
Wed Dec  6 08:23:14 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.105.17   Driver Version: 525.105.17   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   58C    P8    10W /  70W |      0MiB / 15360MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

In [ ]:
```
from transformers import pipeline, set_seed
from datasets import load_dataset, load_from_disk
import matplotlib.pyplot as plt
from datasets import load_dataset
import pandas as pd
from datasets import load_dataset, load_metric

from transformers import AutoModelForSeq2SeqLM, AutoTokenizer

import nltk
from nltk.tokenize import sent_tokenize

from tqdm import tqdm
import torch

nltk.download("punkt")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

Out[ ]:
```
True
```

In [ ]:
```
from transformers import AutoModelForSeq2SeqLM, AutoTokenize

device = "cuda" if torch.cuda.is_available() else "cpu"
device
```

Out[ ]:
```
'cuda'
```

In [ ]:
```
model_ckpt = "google/pegasus-cnn_dailymail"

tokenizer = AutoTokenizer.from_pretrained(model_ckpt)

model_pegasus = AutoModelForSeq2SeqLM.from_pretrained(model_ck
pt).to(device)
```

```
Some weights of PegasusForConditionalGeneration were not ini
alized from the model checkpoint at google/pegasus-cnn_dailyma
il and are newly initialized: ['model.decoder.embed_positions.
weight', 'model.encoder.embed_positions.weight']
You should probably TRAIN this model on a down-stream task to
be able to use it for predictions and inference.
```

In [ ]:
```
!unzip /content/drive/MyDrive/Classroom/pegasus-samsum-model
ip
```

```
Archive:  /content/drive/MyDrive/Classroom/pegasus-samsum-mo
l.zip
  inflating: config.json
  inflating: model.safetensors
  inflating: generation_config.json
```

In [ ]:
```
dataset_samsum = load_from_disk('samsum_dataset')
dataset_samsum
```

Out[ ]:
```
DatasetDict({
    train: Dataset({
        features: ['id', 'dialogue', 'summary'],
        num_rows: 14732
    })
    test: Dataset({
        features: ['id', 'dialogue', 'summary'],
        num_rows: 819
    })
    validation: Dataset({
        features: ['id', 'dialogue', 'summary'],
        num_rows: 818
    })
})
```

In [ ]:
```
split_lengths = [len(dataset_samsum[split]) for split in dat
et_samsum]

print(f"Split lengths: {split_lengths}")
print(f"Features: {dataset_samsum['train'].column_names}")
print("\nDialogue:")

print(dataset_samsum["test"][1]["dialogue"])

print("\nSummary:")

print(dataset_samsum["test"][1]["summary"])
```

```
Split lengths: [14732, 819, 818]
Features: ['id', 'dialogue', 'summary']

Dialogue:
Eric: MACHINE!
Rob: That's so gr8!
Eric: I know! And shows how Americans see Russian ;)
Rob: And it's really funny!
Eric: I know! I especially like the train part!
Rob: Hahaha! No one talks to the machine like that!
Eric: Is this his only stand-up?
Rob: Idk. I'll check.
Eric: Sure.
Rob: Turns out no! There are some of his stand-ups on youtube.
Eric: Gr8! I'll watch them now!
Rob: Me too!
Eric: MACHINE!
Rob: MACHINE!
Eric: TTYL?
Rob: Sure :)

Summary:
Eric and Rob are going to watch a stand-up on youtube.
```

In [ ]:
```
def convert_examples_to_features(example_batch):
    input_encodings = tokenizer(example_batch['dialogue'] , ma
x_length = 1024, truncation = True )

    with tokenizer.as_target_tokenizer():
        target_encodings = tokenizer(example_batch['summary'],
max_length = 128, truncation = True )

    return {
        'input_ids' : input_encodings['input_ids'],
        'attention_mask': input_encodings['attention_mask'],
        'labels': target_encodings['input_ids']
    }
```

In [ ]:
```
dataset_samsum_pt = dataset_samsum.map(convert_examples_to_f
tures, batched = True)
```

```
/usr/local/lib/python3.10/dist-packages/transformers/tokeniz
ion_utils_base.py:3856: UserWarning: `as_target_tokenizer` is
deprecated and will be removed in v5 of Transformers. You can
tokenize your labels by using the argument `text_target` of th
e regular `__call__` method (either in the same call as your i
nput texts if you use the same keyword arguments, or in a sepa
rate call.
  warnings.warn(
```

In [ ]:
```
dataset_samsum_pt["train"]
```

Out[ ]:
```
Dataset({
    features: ['id', 'dialogue', 'summary', 'input_ids', 'atte
ntion_mask', 'labels'],
    num_rows: 14732
})
```

In [ ]:
```
# Training
from transformers import DataCollatorForSeq2Seq

seq2seq_data_collator = DataCollatorForSeq2Seq(tokenizer, mode
l=model_pegasus)
```

In [ ]:
```
from transformers import TrainingArguments, Trainer

trainer_args = TrainingArguments(
    output_dir='pegasus-samsum', num_train_epochs=1, warmup_st
eps=500,
    per_device_train_batch_size=1, per_device_eval_batch_size=
1,
    weight_decay=0.01, logging_steps=10,
    evaluation_strategy='steps', eval_steps=500, save_steps=1e
6,
    gradient_accumulation_steps=16
)
```

In [ ]:
```
trainer = Trainer(model=model_pegasus, args=trainer_args,
                  tokenizer=tokenizer, data_collator=seq2seq_d
ata_collator,
                  train_dataset=dataset_samsum_pt["train"],
                  eval_dataset=dataset_samsum_pt["validatio
n"])
```

In [ ]:
```
trainer.train()
```

```
You're using a PegasusTokenizerFast tokenizer. Please note tl
t with a fast tokenizer, using the `__call__` method is faster
than using a method to encode the text followed by a call to t
he `pad` method to get a padded encoding.
```

[510/510 25:18, Epoch 9/10]

Step	Training Loss	Validation Loss
500	1.251600	1.556377

Out[ ]:
```
TrainOutput(global_step=510, training_loss=1.864230918884277
metrics={'train_runtime': 1524.2532, 'train_samples_per_secon
d': 5.373, 'train_steps_per_second': 0.335, 'total_flos': 3131
112806301696.0, 'train_loss': 1.8642309188842774, 'epoch': 9.9
6})
```

In [ ]:

```
# Evaluation
def generate_batch_sized_chunks(list_of_elements, batch_size):
    """split the dataset into smaller batches that we can proc
ess simultaneously
    Yield successive batch-sized chunks from list_of_element
s."""
    for i in range(0, len(list_of_elements), batch_size):
        yield list_of_elements[i : i + batch_size]


def calculate_metric_on_test_ds(dataset, metric, model, tokeni
zer,
                                batch_size=16, device=device,
                                column_text="article",
                                column_summary="highlights"):
    article_batches = list(generate_batch_sized_chunks(dataset
[column_text], batch_size))
    target_batches = list(generate_batch_sized_chunks(dataset
[column_summary], batch_size))

    for article_batch, target_batch in tqdm(
        zip(article_batches, target_batches), total=len(articl
e_batches)):

        inputs = tokenizer(article_batch, max_length=1024,  tr
uncation=True,
                           padding="max_length", return_tensors
="pt")

        summaries = model.generate(input_ids=inputs["input_id
s"].to(device),
                           attention_mask=inputs["attention_mas
k"].to(device),
                           length_penalty=0.8, num_beams=8, max_
length=128)
        ''' parameter for length penalty ensures that the mode
l does not generate sequences that are too long. '''

        # Finally, we decode the generated texts,
        # replace the  token, and add the decoded texts with t
he references to the metric.
        decoded_summaries = [tokenizer.decode(s, skip_special_
tokens=True,
                                    clean_up_tokenization_spaces=T
rue)
                for s in summaries]

        decoded_summaries = [d.replace("", " ") for d in decod
ed_summaries]


        metric.add_batch(predictions=decoded_summaries, refere
nces=target_batch)

    #  Finally compute and return the ROUGE scores.
    score = metric.compute()
    return score
```

In [ ]:
```
rouge_names = ["rouge1", "rouge2", "rougeL", "rougeLsum"]
rouge_metric = load_metric('rouge')
```

```
<ipython-input-18-5a43aadd1b0e>:2: FutureWarning: load_metri
is deprecated and will be removed in the next major version of
datasets. Use 'evaluate.load' instead, from the new library 🤗
Evaluate: https://huggingface.co/docs/evaluate
  rouge_metric = load_metric('rouge')
```

In [ ]:
```
score = calculate_metric_on_test_ds(
    dataset_samsum['test'][0:10], rouge_metric, trainer.model,
tokenizer, batch_size = 2, column_text = 'dialogue', column_su
mmary= 'summary'
)

rouge_dict = dict((rn, score[rn].mid.fmeasure ) for rn in roug
e_names )

pd.DataFrame(rouge_dict, index = [f'pegasus'] )
```

```
100%|██████████| 5/5 [00:09<00:00,  1.96s/it]
```

Out[ ]:

	rouge1	rouge2	rougeL	rougeLsum
pegasus	0.02201	0.0	0.021917	0.021938

In [ ]:
```
## Save model
model_pegasus.save_pretrained("pegasus-samsum-model")
```

In [ ]:
```
## Save tokenizer
tokenizer.save_pretrained("tokenizer")
```

Out[ ]:
```
('tokenizer/tokenizer_config.json',
 'tokenizer/special_tokens_map.json',
 'tokenizer/spiece.model',
 'tokenizer/added_tokens.json',
 'tokenizer/tokenizer.json')
```

In [ ]:
```
import shutil
shutil.unpack_archive("/content/drive/MyDrive/Classroom/pegasu
s-samsum-model.zip", "/content/pegasus-samsum-model")
shutil.unpack_archive("/content/drive/MyDrive/Classroom/tokeni
zer.zip", "/content/tokenizer")
```

In [ ]:
```
#Load
tokenizer = AutoTokenizer.from_pretrained("/content/tokenize
r")
```

In [ ]:

```
#Prediction
gen_kwargs = {"length_penalty": 0.8, "num_beams":8, "max_lengt
h": 128}

sample_text = """Exploring the depths of the ocean is like ven
turing into an alien world
teeming with mesmerizing life forms and breathtaking landscape
s.
The mysterious abyss holds a wealth of secrets,
with its vast blue expanse and hidden treasures waiting to be
discovered.
As you descend into the fathomless depths,
the water pressure intensifies, creating an otherworldly envir
onment.
Schools of vibrant fish glide effortlessly,
their scales shimmering under the refracted sunlight,
while graceful sea turtles navigate through the currents with
graceful ease.
Delicate coral reefs, resembling underwater gardens,
provide shelter for a kaleidoscope of marine species,
showcasing nature's intricate beauty.
Amidst the quiet serenity, divers might chance upon encounters
with
majestic creatures like the gentle giants of the sea,
the awe-inspiring whales, or the sleek and agile dolphins,
reminding us of the sheer majesty and awe-inspiring wonders
that lie beneath the surface of our planet's oceans.
"""

pipe = pipeline("summarization", model="/content/pegasus-samsu
m-model",tokenizer=tokenizer)

##
print("Dialogue:")
print(sample_text)

print("\nModel Summary:")
print(pipe(sample_text, **gen_kwargs)[0]["summary_text"])
```

Dialogue:
Exploring the depths of the ocean is like venturing into an al
ien world
teeming with mesmerizing life forms and breathtaking landscape
s.
The mysterious abyss holds a wealth of secrets,
with its vast blue expanse and hidden treasures waiting to be
discovered.
As you descend into the fathomless depths,
the water pressure intensifies, creating an otherworldly envir
onment.
Schools of vibrant fish glide effortlessly,
their scales shimmering under the refracted sunlight,
while graceful sea turtles navigate through the currents with
graceful ease.
Delicate coral reefs, resembling underwater gardens,
provide shelter for a kaleidoscope of marine species,
showcasing nature's intricate beauty.
Amidst the quiet serenity, divers might chance upon encounters
with
majestic creatures like the gentle giants of the sea,
the awe-inspiring whales, or the sleek and agile dolphins,
reminding us of the sheer majesty and awe-inspiring wonders
that lie beneath the surface of our planet's oceans.

Model Summary:
The depths of the ocean are teeming with breathtaking life for
ms and breathtaking landscapes. Delicate coral reefs, resembli
ng underwater gardens, provide shelter for a kaleidoscope of m
arine species.<n>Amid the quiet serenity, divers might chance
upon encounters with majestic creatures like the gentle giants
of the sea, the awe-inspiring whales, or the sleek and agile d
olphins.