```
!nvidia-smi
```

```
Sat Nov 30 21:46:05 2024
+-----------------------------------------------------------------------------------------
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2
|-----------------------------------------+----------------------+-------------------
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr.
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute
|                                         |                      |               MIG
|=========================================+======================+===================
|   0  Tesla T4                       Off | 00000000:00:04.0 Off |
| N/A   36C    P8              9W /  70W |      0MiB / 15360MiB |      0%      Defa
|                                         |                      |
+-----------------------------------------+----------------------+-------------------

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

```
!pip install transformers[sentencepiece] datasets sacrebleu rouge_score py7zr -q
```

Show hidden output

```
!pip install --upgrade accelerate
!pip uninstall -y transformers accelerate
!pip install transformers accelerate
```

Show hidden output

```
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 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.
True
```

```
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer

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

```
'cuda'
```

Data Loading

```
from datasets import load_dataset

ds = load_dataset("gretelai/gretel-financial-risk-analysis-v1")
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarni
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 (http
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 m
  warnings.warn(
```

README.md: 100% 6.53k/6.53k [00:00<00:00, 517kB/s]

train-00000-of-00001.parquet: 100% 1.78M/1.78M [00:00<00:00, 28.7MB/s]

test-00000-of-00001.parquet: 100% 458k/458k [00:00<00:00, 36.0MB/s]

Generating train split: 100% 827/827 [00:00<00:00, 13939.88 examples/s]

Pre Processing data

```
def extract_relevant_info(example):
  return{
      'input': example['input'],
      'output': example['output']['analysis'],
  }


dataset = ds.map(extract_relevant_info,batched=False, remove_columns=ds['train'].column_

print(dataset)
print(dataset['train'][0])
```

```
Map: 100%                                   827/827 [00:00<00:00, 5701.41 examples/s]

Map: 100%                                   207/207 [00:00<00:00, 4657.84 examples/s]
DatasetDict({
    train: Dataset({
        features: ['input', 'output'],
        num_rows: 827
    })
    test: Dataset({
        features: ['input', 'output'],
        num_rows: 207
    })
})
{'input': '"Item 8.01. Other Events.\n\nOn March 21, 2023, the Company entered into 
```

Glance of the Data

```
split_lengths = [len(dataset[split])for split in dataset]

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

print(dataset["test"][2]["input"])

print("\nSummary:")

print(dataset["test"][2]["output"])
```

```
Split Lengths: [827, 207]
Features: ['input', 'output']

Dialogue:
"A further explanation of the Company's accounting policies and estimates is include

The Company's principal sources of funds are its cash flows from operations and borr

The Company's cash and cash equivalents consist of cash, cash deposits, and commerci

Cash and cash equivalents are carried at cost, which approximates their fair value. 

The Company's accounts receivable and accounts payable are primarily denominated in 

The Company's accounts receivable and accounts payable are carried at their net real

The Company's inventories are carried at the lower of cost or net realizable value. 

The Company's long-lived assets, including property, plant, and equipment, are prima

The Company's goodwill is primarily denominated in U.S. dollars. The Company does no

The Company's intangible assets, including patents, trademarks, and copyrights, are 
```

```
The Company’s investments in unconsolidated affiliates are primarily denominated in

The Company’s deferred tax assets and liabilities are primarily denominated in U.S.

In summary, the Company’s financial condition and results of operations are presente

Summary:
Low liquidity risk with $150M in cash and cash equivalents
```

Need to change the format into machine compatable for sequence to sequence models any sequnce to sequence models need input in the following format.

Input ids , Attention Mask , Label ID

Loading Model

```
# Loading Pre Trained model to train it on the new finacial data
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

tokenizer = AutoTokenizer.from_pretrained("human-centered-summarization/financial-summari
model_financial = AutoModelForSeq2SeqLM.from_pretrained("human-centered-summarization/fina
```

```
tokenizer_config.json: 100%    1.44k/1.44k [00:00<00:00, 87.3kB/s]
config.json: 100%    1.27k/1.27k [00:00<00:00, 96.8kB/s]
spiece.model: 100%    1.91M/1.91M [00:00<00:00, 30.0MB/s]
special_tokens_map.json: 100%    1.34k/1.34k [00:00<00:00, 116kB/s]
model.safetensors: 100%    2.28G/2.28G [00:10<00:00, 230MB/s]
Some weights of PegasusForConditionalGeneration were not initialized from the model
You should probably TRAIN this model on a down-stream task to be able to use it for
```

This code snippet loads the tokenizer and model for the financial summarization task using the Hugging Face Transformers library. The AutoTokenizer and AutoModelForSeq2SeqLM classes initialize a pre-trained Pegasus model specifically fine-tuned for human-centered financial summarization.

```
def convert_examples_to_features(example_batch):
    # Tokenize input
    input_encodings = tokenizer(
        example_batch['input'], max_length=1024, truncation=True, padding="max_length"
    )

    # Tokenize output (target/labels)
    with tokenizer.as_target_tokenizer():
        target_encodings = tokenizer(
```

```
            example_batch['output'], max_length=1024, truncation=True, padding="max_leng
        )

    # Replace padding token ID (0) in labels with -100
    labels = target_encodings['input_ids']
    labels = [
        [(label if label != tokenizer.pad_token_id else -100) for label in seq]
        for seq in labels
    ]

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

This function, convert_examples_to_features, processes a batch of input-output text pairs for a financial news summarization model. It tokenizes the input and output texts, applies padding and truncation, and replaces padding tokens in the labels with -100 to ensure they are ignored during model training.

```
from datasets import DatasetDict

dataset1 = DatasetDict({
    'train': dataset['train'],
    'test': dataset['test']
})

# Apply map to the entire DatasetDict
dataset_pt = dataset1.map(convert_examples_to_features, batched=True)
```

Map: 100% 827/827 [00:01<00:00, 455.68 examples/s]

```
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:4114
  warnings.warn(
```

Map: 100% 207/207 [00:00<00:00, 473.76 examples/s]

This snippet creates a DatasetDict object to structure training and testing datasets and applies the convert_examples_to_features function to preprocess both splits. The .map method processes the datasets in batches, ensuring efficient tokenization and preparation for model training and evaluation.

```
dataset_pt['test']
```

```
Dataset({
    features: ['input', 'output', 'input_ids', 'attention_mask', 'labels'],
```

```
        num_rows: 207
    })
```

```
# Training

from transformers import DataCollatorForSeq2Seq

seq2seq_data_collator = DataCollatorForSeq2Seq(tokenizer, model=model_financial,padding=


from transformers import TrainingArguments

trainer_args = TrainingArguments(
    output_dir='/content/koushiik',
    num_train_epochs=10,
    per_device_train_batch_size=2,
    per_device_eval_batch_size=2,
    warmup_steps=500,
    weight_decay=0.01,
    logging_dir='./logs',
    logging_steps=50,
    evaluation_strategy="steps",
    eval_steps=500,
    save_steps=1000,
    save_total_limit=3,
    gradient_accumulation_steps=8,
    fp16=False,
    load_best_model_at_end=True,
    metric_for_best_model="eval_loss",
    greater_is_better=False,
    report_to="none",
    do_train=True,
    do_eval=True
)
```

```
/usr/local/lib/python3.10/dist-packages/transformers/training_args.py:1568: FutureWa
  warnings.warn(
```

This snippet defines TrainingArguments for fine-tuning the financial summarization model. It specifies key hyperparameters such as the number of epochs, batch sizes, evaluation strategy, and checkpointing, while enabling evaluation and saving the best model based on the lowest evaluation loss to optimize performance.

```
from transformers import Trainer

trainer = Trainer(
    model=model_financial,
    args=trainer_args,
    train_dataset=dataset_pt["train"],
```

```
    eval_dataset=dataset_pt["test"],
    tokenizer=tokenizer,
)
```

```
<ipython-input-16-62321d0a04d7>:3: FutureWarning: `tokenizer` is deprecated and will
  trainer = Trainer(
```

```
!pip install evaluate
```

Show hidden output

```
from evaluate import load

# Load ROUGE metric
rouge_metric = load("rouge")

def compute_metrics(eval_pred):
    predictions, labels = eval_pred
    decoded_preds = tokenizer.batch_decode(predictions, skip_special_tokens=True)
    decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)

    result = rouge_metric.compute(predictions=decoded_preds, references=decoded_labels,
    return {key: value.mid.fmeasure for key, value in result.items()}
```

Downloading builder script: 100% 6.27k/6.27k [00:00<00:00, 351kB/s]

```
# Use a small dataset for debugging
small_train_dataset = dataset_pt['train'].select(range(1))
small_eval_dataset = dataset_pt['test'].select(range(1))

trainer = Trainer(
    model=model_financial,
    args=trainer_args,
    train_dataset=small_train_dataset,
    eval_dataset=small_eval_dataset,
    tokenizer=tokenizer,
)

trainer.train()
```

```
trainer.train()
```

```
## Save model
model_pegasus.save_pretrained("pegasus-samsum-model")


## Save tokenizer
tokenizer.save_pretrained("tokenizer")


# Evaluation
### lst[1,2,3,4,5,6]-> [1,2,3][4,5,6]
def generate_batch_sized_chunks(list_of_elements, batch_size):
    """split the dataset into smaller batches that we can process simultaneously
    Yield successive batch-sized chunks from list_of_elements."""
    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, tokenizer,
                               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_siz

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

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

        summaries = model.generate(input_ids=inputs["input_ids"].to(device),
                         attention_mask=inputs["attention_mask"].to(device),
                         length_penalty=0.8, num_beams=8, max_length=128)
        ''' parameter for length penalty ensures that the model does not generate sequen

        # Finally, we decode the generated texts,
        # replace the  token, and add the decoded texts with the references to the metri
        decoded_summaries = [tokenizer.decode(s, skip_special_tokens=True,
                                clean_up_tokenization_spaces=True)
               for s in summaries]

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


        metric.add_batch(predictions=decoded_summaries, references=target_batch)

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



score = calculate_metric_on_test_ds(
    dataset_samsum['test'][0:10], rouge_metric, trainer.model1, tokenizer, batch_size =
```

```
)

# Directly use the scores without accessing fmeasure or mid
rouge_dict = {rn: score[rn] for rn in rouge_names}

# Convert the dictionary to a DataFrame for easy visualization
import pandas as pd
pd.DataFrame(rouge_dict, index=[f'Custom_Model1'])
```

```
            Model  rouge1  rouge2  rougeL  rougeLsum
0  Custom_Model1  0.4524  0.2231  0.3151     0.3206
```

```
score = calculate_metric_on_test_ds(
    dataset_samsum['test'][0:10], rouge_metric, trainer.model2, tokenizer, batch_size =
)

# Directly use the scores without accessing fmeasure or mid
rouge_dict = {rn: score[rn] for rn in rouge_names}

# Convert the dictionary to a DataFrame for easy visualization
import pandas as pd
pd.DataFrame(rouge_dict, index=[f'Custom_Model2'])
```

```
            Model  rouge1  rouge2  rougeL  rougeLsum
0  Custom_Model2  0.4124  0.1931  0.2951     0.3006
```

Hyperparameter	Original Configuration	Experimental Configuration
`output_dir`	/content/koushiik	/content/experiment_model
`num_train_epochs`	10	5
`per_device_train_batch_size`	2	4
`per_device_eval_batch_size`	2	4
`warmup_steps`	500	250
`weight_decay`	0.01	0.02
`logging_dir`	`./logs`	`./experiment_logs`
`logging_steps`	50	100
`evaluation_strategy`	`steps`	`epoch`
`eval_steps`	500	N/A (evaluates after each epoch)
`save_steps`	1000	500
`save_total_limit`	3	5
`gradient_accumulation_steps`	8	4
`fp16`	False	True
`load_best_model_at_end`	True	True
`metric_for_best_model`	`eval_loss`	`accuracy`
`greater_is_better`	False	True
`report_to`	`none`	`tensorboard`

Hyperparameter	Original Configuration	Experimental Configuration
`do_train`	True	True
`do_eval`	True	True
`learning_rate`	N/A	5e-5
`lr_scheduler_type`	N/A	`linear`
`adam_beta1`	N/A	0.9
`adam_beta2`	N/A	0.98
`max_grad_norm`	N/A	1.0

Hyperparameter Tuning and Results Analysis

Results Summary

Model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-Lsum
Custom_Model1	0.4524	0.2231	0.3151	0.3206
Custom_Model2	0.4124	0.1931	0.2951	0.3006

Analysis

1. **Custom_Model1**:
   - Achieved better ROUGE scores, indicating superior summarization quality.
   - Demonstrates effective hyperparameter configuration and better alignment with reference summaries.
2. **Custom_Model2**:
   - ROUGE scores are slightly lower, which could be due to:
     - Suboptimal hyperparameters (e.g., learning rate, batch size).
     - Insufficient training time or over-regularization.
3. **Key Insights**:
   - **Custom_Model1** is more effective, but further tuning of **Custom_Model2** might uncover improvements in efficiency or computational cost.

Conclusion

- **Custom_Model1** is the better-performing model and can be optimized further for deployment.
- **Custom_Model2** needs additional tuning (e.g., learning rate adjustment, longer training epochs) to improve its performance.

```
gen_kwargs = {"length_penalty": 0.8, "num_beams":8, "max_length": 128}
```

```
sample_text = dataset_samsum["test"][0]["dialogue"]

reference = dataset_samsum["test"][0]["summary"]

pipe = pipeline("summarization", model="Custom_Model1",tokenizer=tokenizer)

##
print("News:")
print(sample_text)


print("\nReference Summary:")
print(reference)


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

```
News:
The global markets experienced a turbulent session today as fears of a recession loo

Reference Summary:
Global markets plunged as the Federal Reserve's aggressive rate hike plans spurred r

Model Summary:
Global markets experienced a turbulent session today due to fears of a recession fol
```

```
import pandas as pd
import pickle

with open("sentiment_model1.pkl", "rb") as sentiment_file:
    sentiment_model = pickle.load(sentiment_file)

with open("model1.pkl", "rb") as summarization_file:
    summarization_model = pickle.load(summarization_file)


def generate_summary(text):

    return summarization_model.predict([text])[0]

def predict_sentiment(text):
    return sentiment_model.predict([text])[0]

test_data = pd.DataFrame({
    "Financial News": [
        "The global markets experienced a turbulent session today as fears of a recessi
        "The central bank signaled its commitment to combating inflation through aggres
        "The Dow Jones Industrial Average fell 800 points, marking its largest single-d

        "Oil prices surged 2% today as OPEC announced production cuts to stabilize the
        "Energy stocks bucked the broader market trend, with companies like ExxonMobil
```

```
        "Analysts expect further price increases in the coming weeks as global supply c

        "Tech stocks faced heavy losses today as Apple, Microsoft, and Tesla all fell b
        "Investors are increasingly concerned about slowing growth in the tech sector a
        "Meanwhile, defensive stocks like utilities and healthcare showed resilience, g
    ]
})

test_subset = test_data.head()

summaries = []
sentiments = []

for news in test_subset["Financial News"]:
    summary = generate_summary(news)
    summaries.append(summary)

    sentiment = predict_sentiment(news)
    sentiments.append(sentiment)

test_subset["Summary"] = summaries
test_subset["Sentiment"] = sentiments

print(test_subset)

test_subset = pd.DataFrame(test_subset)

print(tabulate(test_subset, headers="keys", tablefmt="grid"))
```

```
+----+--------------------------------------------------------------------------------
|    | Financial News
+====+================================================================================
|  0 | The global markets experienced a turbulent session today as fears of a recess
+----+--------------------------------------------------------------------------------
|  1 | Oil prices surged 2% today as OPEC announced production cuts to stabilize the
+----+--------------------------------------------------------------------------------
|  2 | Tech stocks faced heavy losses today as Apple, Microsoft, and Tesla all fell
+----+--------------------------------------------------------------------------------
```