```
#Checking if GPU is running or not

!nvidia-smi
```

```
Sat Aug  5 06:02:48 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   66C    P8    11W /  70W |      0MiB / 15360MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

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

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

```
import os
import sys
import transformers
import tensorflow as tf
from datasets import load_dataset
from transformers import AutoTokenizer
from transformers import TFAutoModelForSeq2SeqLM, DataCollatorForSeq2Seq
from transformers import AdamWeightDecay
from transformers import AutoTokenizer, TFAutoModelForSeq2SeqLM
```

```
model_checkpoint = "Helsinki-NLP/opus-mt-en-hi"
```

Helsinki-NLP/opus-mt-en-hi model source: https://huggingface.co/Helsinki-NLP/opus-mt-en-hi

The Dataset Source: https://huggingface.co/datasets/cfilt/iitb-english-hindi

```
raw_datasets = load_dataset("cfilt/iitb-english-hindi")
```

```
Repo card metadata block was not found. Setting CardData to empty.
WARNING:huggingface_hub.repocard:Repo card metadata block was not found. Setting CardData to empty.
```

```
raw_datasets
```

```
DatasetDict({
    train: Dataset({
        features: ['translation'],
        num_rows: 1659083
    })
    validation: Dataset({
        features: ['translation'],
        num_rows: 520
    })
    test: Dataset({
        features: ['translation'],
        num_rows: 2507
    })
})
```

```
raw_datasets['train'][1]
```

```
{'translation': {'en': 'Accerciser Accessibility Explorer',
  'hi': 'एक्सेर्साइसर पहुंचनीयता अन्वेषक'}}
```

Preprocessing the data

```
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
```

```
/usr/local/lib/python3.10/dist-packages/transformers/models/marian/tokenization_marian.py:194: UserWarning: Recommended: pip instal
  warnings.warn("Recommended: pip install sacremoses.")
```

```
tokenizer("Hello, this is a sentence!")
```

```
{'input_ids': [12110, 2, 90, 23, 19, 8800, 61, 0], 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1]}
```

```
tokenizer("Hello, I am padma!")
```

```
{'input_ids': [12110, 2, 56, 489, 44, 14586, 8038, 61, 0], 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1]}
```

```
tokenizer(["Hello, this is a sentence!", "This is another sentence."])
```

```
{'input_ids': [[12110, 2, 90, 23, 19, 8800, 61, 0], [239, 23, 414, 8800, 3, 0]], 'attention_mask': [[1, 1, 1, 1, 1, 1, 1, 1], [1,
1, 1, 1, 1, 1]]}
```

```
with tokenizer.as_target_tokenizer():
    print(tokenizer(["एक्सेसाइसर पहुंचनीयता अन्वेषक"]))
```

```
{'input_ids': [[26618, 16155, 346, 33383, 0]], 'attention_mask': [[1, 1, 1, 1, 1]]}
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:3635: UserWarning: `as_target_tokenizer` is deprecа
  warnings.warn(
```

```
max_input_length = 128
max_target_length = 128

source_lang = "en"
target_lang = "hi"


def preprocess_function(examples):
    inputs = [ex[source_lang] for ex in examples["translation"]]
    targets = [ex[target_lang] for ex in examples["translation"]]
    model_inputs = tokenizer(inputs, max_length=max_input_length, truncation=True)

    # Setup the tokenizer for targets
    with tokenizer.as_target_tokenizer():
        labels = tokenizer(targets, max_length=max_target_length, truncation=True)

    model_inputs["labels"] = labels["input_ids"]
    return model_inputs
```

```
preprocess_function(raw_datasets["train"][:2])
```

```
{'input_ids': [[3872, 85, 2501, 132, 15441, 36398, 0], [32643, 28541, 36253, 0]], 'attention_mask': [[1, 1, 1, 1, 1, 1, 1], [1, 1,
1, 1]], 'labels': [[63, 2025, 18, 16155, 346, 20311, 24, 2279, 679, 0], [26618, 16155, 346, 33383, 0]]}
```

```
tokenized_datasets = raw_datasets.map(preprocess_function, batched=True)
```

```
model = TFAutoModelForSeq2SeqLM.from_pretrained(model_checkpoint)
```

```
All model checkpoint layers were used when initializing TFMarianMTModel.

All the layers of TFMarianMTModel were initialized from the model checkpoint at Helsinki-NLP/opus-mt-en-hi.
If your task is similar to the task the model of the checkpoint was trained on, you can already use TFMarianMTModel for predictions
```

```
batch_size = 16
learning_rate = 2e-5
weight_decay = 0.01
num_train_epochs = 10
```

```
data_collator = DataCollatorForSeq2Seq(tokenizer, model=model, return_tensors="tf")
```

```
generation_data_collator = DataCollatorForSeq2Seq(tokenizer, model=model, return_tensors="tf", pad_to_multiple_of=128)
```

```
train_dataset = model.prepare_tf_dataset(
    tokenized_datasets["test"],
    batch_size=batch_size,
    shuffle=True,
    collate_fn=data_collator,
)
```

```
validation_dataset = model.prepare_tf_dataset(
    tokenized_datasets["validation"],
```

```
    batch_size=batch_size,
    shuffle=False,
    collate_fn=data_collator,
)
```

```
generation_dataset = model.prepare_tf_dataset(
    tokenized_datasets["validation"],
    batch_size=8,
    shuffle=False,
    collate_fn=generation_data_collator,
)
```

```
optimizer = AdamWeightDecay(learning_rate=learning_rate, weight_decay_rate=weight_decay)
model.compile(optimizer=optimizer)
```

```
model.fit(train_dataset, validation_data=validation_dataset, epochs=10)
```

```
Epoch 1/10
156/156 [==============================] - 98s 359ms/step - loss: 3.7686 - val_loss: 3.9454
Epoch 2/10
156/156 [==============================] - 50s 321ms/step - loss: 3.3201 - val_loss: 3.8700
Epoch 3/10
156/156 [==============================] - 48s 309ms/step - loss: 3.0186 - val_loss: 3.8354
Epoch 4/10
156/156 [==============================] - 50s 321ms/step - loss: 2.7796 - val_loss: 3.8120
Epoch 5/10
156/156 [==============================] - 48s 310ms/step - loss: 2.5603 - val_loss: 3.8180
Epoch 6/10
156/156 [==============================] - 48s 307ms/step - loss: 2.3789 - val_loss: 3.8225
Epoch 7/10
156/156 [==============================] - 50s 322ms/step - loss: 2.2036 - val_loss: 3.8327
Epoch 8/10
156/156 [==============================] - 50s 322ms/step - loss: 2.0554 - val_loss: 3.8375
Epoch 9/10
156/156 [==============================] - 47s 302ms/step - loss: 1.9046 - val_loss: 3.8419
Epoch 10/10
156/156 [==============================] - 50s 319ms/step - loss: 1.7708 - val_loss: 3.8748
<keras.callbacks.History at 0x7df0b076ff10>
```

```
model.save_pretrained("tf_model/")
```

Model Testing

```
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
model = TFAutoModelForSeq2SeqLM.from_pretrained("tf_model/")
```

```
All model checkpoint layers were used when initializing TFMarianMTModel.

All the layers of TFMarianMTModel were initialized from the model checkpoint at tf_model/.
If your task is similar to the task the model of the checkpoint was trained on, you can already use TFMarianMTModel for predictions
```

```
input_text  = "My name padma, I am a datascientist"

tokenized = tokenizer([input_text], return_tensors='np')
out = model.generate(**tokenized, max_length=128)
print(out)
```

```
tf.Tensor([[61949   500   179 21183  4807     2   104    38  4977 10972   254     0]], shape=(1, 12), dtype=int32)
```

```
with tokenizer.as_target_tokenizer():
    print(tokenizer.decode(out[0], skip_special_tokens=True))
```

```
मेरा नाम अल्मा, मैं एक डेटिस्ट हूँ
```

✓ 0s completed at 11:45 AM