```
#Checking if GPU is running or not

!nvidia-smi
```

```
Fri Sep  1 05:24:46 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   75C    P8    14W /  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
```

```
# Importing the Required Libraries
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"
```

```
# Loading 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.
```

```
# Dataset Info
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'}}
```

```
raw_datasets['test'][1]
```

```
{'translation': {'en': "As America's road planners struggle to find the cash to mend a crumbling highway system, many are beginning to
see a solution in a little black box that fits neatly by the dashboard of your car.",
  'hi': 'जबकि अमेरिका के सड़क योजनाकार, ध्वस्त होते हुए हाईवे सिस्टम को सुधारने के लिए धन की कमी से जूझ रहे हैं, वहीं बहुत-से लोग इसका समाधान छोटे से ब्लैक
बॉक्स में देख रहे हैं, जो आपकी कार के डैशबोर्ड पर सफ़ाई से फिट हो जाता है।'}}
```

Preprocessing

```
# Initializing the Tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)

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

```
tokenizer(["I had about a 30 minute demo just using this new headset"])

     {'input_ids': [[56, 154, 195, 19, 1671, 7336, 35914, 469, 1192, 90, 336, 1876, 8907, 0]], 'attention_mask': [[1, 1, 1, 1, 1, 1, 1, 1,
     1, 1, 1, 1, 1, 1]]}
```

```
with tokenizer.as_target_tokenizer():
    print(tokenizer(["मझुे सि र्फ ३० minute का demo मि ला था इस नयेheadset का इस्तमे ाल करने के लिए"]))

     {'input_ids': [[4095, 14034, 3625, 44, 174, 18943, 3353, 14921, 6785, 3383, 39169, 24, 3947, 363, 818, 3245, 1754, 82, 89, 4075, 2326, 4
     /usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:3660: UserWarning: `as_target_tokenizer` is deprecated a
       warnings.warn(
```

```
# Tokenzing the English and Hindi words
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"][:1])

     {'input_ids': [[3872, 85, 2501, 132, 15441, 36398, 0]], 'attention_mask': [[1, 1, 1, 1, 1, 1, 1]], 'labels': [[63, 2025, 18, 16155,
     346, 20311, 24, 2279, 679, 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 with
```

```
# Initializing the Hyper Parameter
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)
```

```
tokenized_datasets['train']
```

```
Dataset({
    features: ['translation', 'input_ids', 'attention_mask', 'labels'],
    num_rows: 1659083
})
```

```
# Spliting the Dataset to Train and Test data
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,
)


# Declaring the optimizer for improving the accuracy and reduce the loss
optimizer = AdamWeightDecay(learning_rate=learning_rate, weight_decay_rate=weight_decay)
model.compile(optimizer=optimizer)
```

Training the Model

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

```
Epoch 1/10
156/156 [==============================] - 94s 375ms/step - loss: 3.7551 - val_loss: 3.9527
Epoch 2/10
156/156 [==============================] - 49s 312ms/step - loss: 3.3230 - val_loss: 3.8702
Epoch 3/10
156/156 [==============================] - 51s 325ms/step - loss: 3.0182 - val_loss: 3.8317
Epoch 4/10
156/156 [==============================] - 50s 323ms/step - loss: 2.7760 - val_loss: 3.8207
Epoch 5/10
156/156 [==============================] - 50s 320ms/step - loss: 2.5713 - val_loss: 3.8223
Epoch 6/10
156/156 [==============================] - 49s 314ms/step - loss: 2.3807 - val_loss: 3.8225
Epoch 7/10
156/156 [==============================] - 49s 316ms/step - loss: 2.2086 - val_loss: 3.8284
Epoch 8/10
156/156 [==============================] - 50s 320ms/step - loss: 2.0565 - val_loss: 3.8544
Epoch 9/10
156/156 [==============================] - 51s 328ms/step - loss: 1.9127 - val_loss: 3.8658
Epoch 10/10
156/156 [==============================] - 51s 324ms/step - loss: 1.7789 - val_loss: 3.8789
<keras.callbacks.History at 0x7ee0130dabc0>
```

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

Model Testing

```
from nltk.translate.bleu_score import sentence_bleu
```

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

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

```
raw_datasets['train']['translation'][1]['en']
```

```
'Accerciser Accessibility Explorer'
```

```
pred_input=raw_datasets['train']['translation'][1]['en']
tokenized = tokenizer([pred_input], return_tensors='np')
out = model.generate(**tokenized, max_length=128)
print(out)
```

```
tf.Tensor([[61949 26618 16155   346 33383     0 61949]], shape=(1, 7), dtype=int32)
```

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

```
# Input value
raw_datasets['train']['translation'][1]['en']
```

```
'Accerciser Accessibility Explorer'
```

```
# Orginal values
raw_datasets['train']['translation'][1]['hi']
```

```
'एक्सेर्साइसर पहुंचनीयता अन्वेषक'
```

```
# Predicted Value
prediction
```

```
'एक्सेर्साइसर पहुंचनीयता अन्वेषक'
```

Testing with New Inputs

```
# Input 1
input_text  = "Definitely share your feedback in the comment section."

tokenized = tokenizer([input_text], return_tensors='np')
out = model.generate(**tokenized, max_length=128)
with tokenizer.as_target_tokenizer():
    print(tokenizer.decode(out[0], skip_special_tokens=True))
```

```
टिप्पणी खंड में निश्चित रूप से अपनी प्रतिक्रिया साझा करें।
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:3660: UserWarning: `as_target_tokenizer` is deprecated a
  warnings.warn(
```

```
# Input 2
input_text  = "So even if it's a big video, I will clearly mention all the products."

tokenized = tokenizer([input_text], return_tensors='np')
out = model.generate(**tokenized, max_length=128)
with tokenizer.as_target_tokenizer():
    print(tokenizer.decode(out[0], skip_special_tokens=True))
```

```
तो यह एक बड़ा वीडियो है, तो भी मैं सभी उत्पादों का स्पष्ट रूप से उल्लेख करेंगे।
```

```
# Input 3
input_text  = "I was waiting for my bag."

tokenized = tokenizer([input_text], return_tensors='np')
out = model.generate(**tokenized, max_length=128)
with tokenizer.as_target_tokenizer():
    print(tokenizer.decode(out[0], skip_special_tokens=True))
```

```
मैं अपने बैग के लिए प्रतीक्षा कर रहा था।
```

✓ 5s completed at 11:49 AM