Installing Dependecies

```
!pip install datasets
```

Show hidden output

Importing Libraries

```
import pandas as pd
import re
import ast
import numpy as np
import torch
from torch.utils.data import Dataset, DataLoader
from tqdm import tqdm
import time
from sklearn.utils import shuffle
from sklearn.metrics import accuracy_score
```

```
# check for gpu
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(device)
```

```
cuda
```

Loading the Dataset

```
from datasets import load_dataset

ds = load_dataset("stanfordnlp/sst2")
```

```
print(ds)
```

```
DatasetDict({
    train: Dataset({
        features: ['idx', 'sentence', 'label'],
        num_rows: 67349
    })
    validation: Dataset({
        features: ['idx', 'sentence', 'label'],
        num_rows: 872
    })
    test: Dataset({
        features: ['idx', 'sentence', 'label'],
        num_rows: 1821
    })
})
```

```
train_df = ds['train'].to_pandas()
test_df = ds['test'].to_pandas()
valid_df = ds['validation'].to_pandas()
```

```
train_df.head()
```

	idx	sentence	label
0	0	hide new secretions from the parental units	0
1	1	contains no wit , only labored gags	0
2	2	that loves its characters and communicates som...	1
3	3	remains utterly satisfied to remain the same t...	0
4	4	[illegible]	0

```
train_df = train_df[:-10000]
```

```
print(train_df.shape)
print(test_df.shape)
print(valid_df.shape)
```

```
(57349, 3)
(1821, 3)
(872, 3)
```

Custom Dataset

```
import torch
from torch.utils.data import Dataset

class MyDataset(Dataset):
    def __init__(self, texts, labels, tokenizer, max_length):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        text = self.texts[idx]
        label = self.labels[idx]

        # Tokenizing text
        encoding = self.tokenizer(
            text,
            padding='max_length',
            truncation=True,
            max_length=self.max_length,
            return_tensors="pt"
        )

        input_ids = encoding["input_ids"].squeeze(0)
        attention_mask = encoding["attention_mask"].squeeze(0)

        return {
            "input_ids": input_ids,
            "attention_mask": attention_mask,
            "labels": torch.tensor(label, dtype=torch.long)
        }
```

```
# Conversion to list for easier processing
train_text = train_df['sentence'].tolist()
train_label = train_df['label'].tolist()
```

```
valid_text = valid_df['sentence'].tolist()
valid_label = valid_df['label'].tolist()
```

```
test_text = test_df['sentence'].tolist()
test_label = test_df['label'].tolist()
```

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from transformers import XLMRobertaTokenizer, XLMRobertaForSequenceClassification
from transformers import TrainingArguments, Trainer
```

```
tokenizer = XLMRobertaTokenizer.from_pretrained("xlm-roberta-base")
```

```
tokenizer_config.json:  100%                                        25.0/25.0  [00:00<00:00,  2.40kB/s]
sentencepiece.bpe.model:  100%                                        5.07M/5.07M  [00:00<00:00,  20.5MB/s]
tokenizer.json:  100%                                        9.10M/9.10M  [00:00<00:00,  17.8MB/s]
```

```
max_length = 48
```

```
train_dataset = MyDataset(train_text, train_label, tokenizer, max_length)
```

```
valid_dataset = MyDataset(valid_text, valid_label, tokenizer, max_length)
```

```
test_dataset = MyDataset(test_text, test_label, tokenizer, max_length)
```

```
train_loader = DataLoader(train_dataset, batch_size=8, shuffle=True)
```

```
for batch in train_loader:
    print(batch)
    break
```

Show hidden output

```
valid_loader = DataLoader(valid_dataset, batch_size=8, shuffle=False)
```

```
test_loader = DataLoader(test_dataset, batch_size=8, shuffle=False)
```

```
from torch.optim import Adam
from transformers import get_scheduler
```

Defining the Model

```
model = AutoModelForSequenceClassification.from_pretrained("xlm-roberta-base", num_labels=2)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)
```

```
Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back to regular HTTP download. For better perfo
WARNING:huggingface_hub.file_download:Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back to r
model.safetensors: 100%                                          1.12G/1.12G [00:04<00:00, 153MB/s]
Some weights of XLMRobertaForSequenceClassification were not initialized from the model checkpoint at xlm-roberta-base and are newly ini
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
XLMRobertaForSequenceClassification(
  (roberta): XLMRobertaModel(
    (embeddings): XLMRobertaEmbeddings(
      (word_embeddings): Embedding(250002, 768, padding_idx=1)
      (position_embeddings): Embedding(514, 768, padding_idx=1)
      (token_type_embeddings): Embedding(1, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): XLMRobertaEncoder(
      (layer): ModuleList(
        (0-11): 12 x XLMRobertaLayer(
          (attention): XLMRobertaAttention(
            (self): XLMRobertaSdpaSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): XLMRobertaSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): XLMRobertaIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): XLMRobertaOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
  )
  (classifier): XLMRobertaClassificationHead(
    (dense): Linear(in_features=768, out_features=768, bias=True)
    (dropout): Dropout(p=0.1, inplace=False)
    (out_proj): Linear(in_features=768, out_features=2, bias=True)
  )
)
```

Setting Parameters

```
optimizer = Adam(model.parameters(), lr=2e-5)
num_epochs = 5
num_training_steps = num_epochs * len(train_loader)
lr_scheduler = get_scheduler("linear", optimizer=optimizer, num_warmup_steps=0, num_training_steps=num_training_steps)
loss_fn = torch.nn.CrossEntropyLoss()
```

Training the model

```
# Training Loop
for epoch in range(num_epochs):
    model.train()
    total_loss = 0
    start_time = time.time()

    for batch in tqdm(train_loader, desc=f"Epoch {epoch + 1}/{num_epochs}", unit="batch"):
        optimizer.zero_grad()

        input_ids = batch["input_ids"].to(device)
        attention_mask = batch["attention_mask"].to(device)
        labels = batch["labels"].to(device)
        outputs = model(input_ids=input_ids, attention_mask=attention_mask, labels=labels)
        loss = outputs.loss
        total_loss += loss.item()

        loss.backward()
        optimizer.step()
        lr_scheduler.step()

    end_time = time.time()
    epoch_time = end_time - start_time

    avg_train_loss = total_loss / len(train_loader)
    print(f"Epoch {epoch+1}: Train Loss = {avg_train_loss:.4f}, Time: {epoch_time:.2f} seconds")

    model.eval()
    correct = 0
    total = 0
    with torch.no_grad():
        for batch in tqdm(valid_loader, desc="Validation", unit="batch"):
            input_ids = batch["input_ids"].to(device)
            attention_mask = batch["attention_mask"].to(device)
            labels = batch["labels"].to(device)

            outputs = model(input_ids=input_ids, attention_mask=attention_mask)
            predictions = torch.argmax(outputs.logits, dim=-1)

            correct += (predictions == labels).sum().item()
            total += labels.size(0)

    accuracy = correct / total
    print(f"Epoch {epoch+1}: Validation Accuracy = {accuracy:.4f}")
```

```
Epoch 1/3: 100%|██████████| 7169/7169 [17:00<00:00,  7.02batch/s]
Epoch 1: Train Loss = 0.3737, Time: 1020.94 seconds
Validation: 100%|██████████| 109/109 [00:02<00:00, 41.57batch/s]
Epoch 1: Validation Accuracy = 0.9163
Epoch 2/3: 100%|██████████| 7169/7169 [16:59<00:00,  7.03batch/s]
Epoch 2: Train Loss = 0.1755, Time: 1019.50 seconds
Validation: 100%|██████████| 109/109 [00:02<00:00, 40.63batch/s]
Epoch 2: Validation Accuracy = 0.9186
Epoch 3/3: 100%|██████████| 7169/7169 [16:59<00:00,  7.03batch/s]
Epoch 3: Train Loss = 0.1082, Time: 1019.74 seconds
Validation: 100%|██████████| 109/109 [00:02<00:00, 40.86batch/s]Epoch 3: Validation Accuracy = 0.9278
```

Preparing Hindi Dataset

```
with open("neg_train.txt", "r", encoding="utf-8") as f:
    sentences = [line.strip() for line in f if line.strip()]

neg_label = [0] * len(sentences)

df_neg = pd.DataFrame({
    "sentence": sentences,
    "label": neg_label
})
```

```
df_neg.head()
```

	sentence	label
0	बुंदेलखंड के किसानों को अबकी बार बड़ी उम्मीदें...	0
1	उन्होंने कहा कि चूंकि पूंजी की जरूरत बहुत बड़ी...	0
2	पश्चिम बंगाल में शासन द्वारा ऐसी शिकायतें लगात...	0
3	शहर में नेटवर्किंग कारोबार का मकड़जाल लगातार फ...	0

```
with open("pos_train.txt", "r", encoding="utf-8") as f:
    sentences = [line.strip() for line in f if line.strip()]
```

```
pos_label = [1] * len(sentences)

df_pos = pd.DataFrame({
    "sentence": sentences,
    "label": pos_label
})
```

```
df_pos.head()
```

	sentence	label
0	मुलाकात के बाद दोनों तरफ के जवानों ने वॉलीबॉल ...	1
1	दोनों में दोस्ताना माहौल लग रहा है इसलिए इन्हे...	1
2	लिटिल एंजिल स्कूल में दिलचस्प मैत्रीपूर्ण क्रि...	1
3	मैत्रीपूर्ण मैच में अखिलेश कुमार को मैन आफ द म...	1

```
hindi_df = pd.concat([df_pos, df_neg], ignore_index=True)
hindi_df = shuffle(hindi_df, random_state=42)
```

```
hindi_df.head()
```

	sentence	label
2032	तापमान में आ रही गिरावट का असर जनजीवन पर दिख र...	0
906	यह मेरी जिंदगी का सबसे आकर्षक समय है और अगर मे...	1
1128	लंद हौसलों के साथ कुछ भी करना संभव है	1
2004	नीतीश राजनीति में कठिन दौर से गुजर रहे है।	0

```
hindi_df.shape
```

```
(2387, 2)
```

```
hindi_texts = hindi_df['sentence'].tolist()
hindi_labels = hindi_df['label'].tolist()
```

```
hindi_dataset = MyDataset(hindi_texts, hindi_labels, tokenizer, max_length)
```

```
hindi_loader = DataLoader(hindi_dataset, batch_size=8, shuffle=False)
```

Making Prediction on Hindi Dataset

```
all_predictions = []
all_labels = []

model.eval()
with torch.no_grad():
    for batch in tqdm(hindi_loader, desc="Predicting on Hindi Dataset"):
        input_ids = batch["input_ids"].to(device)
        attention_mask = batch["attention_mask"].to(device)
        labels = batch["labels"].to(device)

        outputs = model(input_ids=input_ids, attention_mask=attention_mask)
        predictions = torch.argmax(outputs.logits, dim=-1)

        all_predictions.extend(predictions.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())
```

```
accuracy = accuracy_score(all_labels, all_predictions)
```