News Topic Classifier using BERT + AG News Dataset

This project demonstrates how to fine-tune a pre-trained transformer model (BERT) on the AG News dataset to classify news headlines into four categories: World, Sports, Business, and Sci/Tech. The implementation uses Hugging Face Transformers, Datasets, and Gradio for deployment in Google Colab.

Steps involved: 1. Install required libraries (Transformers, Datasets, Torch, Scikit-learn, Gradio). 2. Load and preprocess the AG News dataset using a BERT tokenizer. 3. Fine-tune the `bert-base-uncased` model with Hugging Face Trainer API. 4. Evaluate the model using Accuracy and F1-score. 5. Deploy the model using Gradio for live interaction in Google Colab.

Google Colab Ready Code:

```
# ==============================
# News Topic Classifier using BERT + AG News Dataset
# Google Colab Ready Code
# ==============================

!pip install -q transformers datasets torch scikit-learn gradio

import os
import numpy as np
import torch
from datasets import load_dataset
from transformers import (
    BertTokenizer,
    BertForSequenceClassification,
    Trainer,
    TrainingArguments,
    DataCollatorWithPadding,
    set_seed,
)
from sklearn.metrics import accuracy_score, f1_score
import gradio as gr

set_seed(42)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

dataset = load_dataset("ag_news")

model_name = "bert-base-uncased"
tokenizer = BertTokenizer.from_pretrained(model_name)

def tokenize(batch):
    return tokenizer(batch["text"], padding="max_length", truncation=True, max_length=128)

tokenized = dataset.map(tokenize, batched=True, remove_columns=["text"])
tokenized = tokenized.rename_column("label", "labels")
tokenized.set_format(type="torch")

num_labels = 4
model = BertForSequenceClassification.from_pretrained(model_name, num_labels=num_labels).to(device)

def compute_metrics(pred):
    labels = pred.label_ids
    preds = np.argmax(pred.predictions, axis=1)
    acc = accuracy_score(labels, preds)
    f1 = f1_score(labels, preds, average="weighted")
    return {"accuracy": acc, "f1": f1}

training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
```

```
    metric_for_best_model="f1",
    greater_is_better=True,
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=2,
    weight_decay=0.01,
    warmup_ratio=0.1,
    logging_dir="./logs",
    logging_steps=50,
    report_to="none",
    fp16=torch.cuda.is_available(),
)

data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized["train"],
    eval_dataset=tokenized["test"],
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics,
)

trainer.train()
eval_metrics = trainer.evaluate()
print("Evaluation:", eval_metrics)

os.makedirs("./checkpoint", exist_ok=True)
trainer.save_model("./checkpoint")
tokenizer.save_pretrained("./checkpoint")

id2label = {0: "World", 1: "Sports", 2: "Business", 3: "Sci/Tech"}

def predict(text):
    model.eval()
    with torch.no_grad():
        inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_length=128)
        outputs = model(**inputs)
        probs = torch.softmax(outputs.logits, dim=-1).squeeze().cpu().numpy()
        return {id2label[i]: float(probs[i]) for i in range(num_labels)}

demo = gr.Interface(
    fn=predict,
    inputs=gr.Textbox(lines=3, placeholder="Enter a news headline or short article..."),
    outputs=gr.Label(num_top_classes=4),
    title="News Topic Classifier (BERT • AG News)",
    description="Fine-tuned bert-base-uncased on AG News. Returns class probabilities.",
)

demo.launch(share=True)
```