Made by Ansh Sharma- Sentimental Analysis by Hugging Face Fine-Tuning BERT , Deployed on wandb AI, Heroku through FASTAPI and Local Server, followed by Testing and Evaluating + Productization of the Model

---

Attached with various Screenshots , and server Analysis.

```
pip install transformers datasets scikit-learn torch matplotlib
```

Show hidden output

Preparing Synthetic Data and Dividing dataset into Training and Testing

```
from datasets import Dataset
from sklearn.model_selection import train_test_split
import pandas as pd

# Example synthetic dataset
data = {
    "ReviewText": [
        "The event was amazing and well-organized!",
        "The amenities were not up to the mark.",
        "Had a decent experience overall.",
        "The campus event was a complete disaster.",
        "I loved every moment of the event!"
    ],
    "SentimentLabel": [2, 0, 1, 0, 2]  # 0: Negative, 1: Neutral, 2: Positive
}

# Create a DataFrame
df = pd.DataFrame(data)

# Split into train, validation, and test sets
train_texts, test_texts, train_labels, test_labels = train_test_split(
    df["ReviewText"], df["SentimentLabel"], test_size=0.2, random_state=42
)

train_texts, val_texts, train_labels, val_labels = train_test_split(
    train_texts, train_labels, test_size=0.1, random_state=42
)

# Convert to Hugging Face Dataset
```

```
train_dataset = Dataset.from_dict({"text": train_texts, "label": train_labels})
val_dataset = Dataset.from_dict({"text": val_texts, "label": val_labels})
test_dataset = Dataset.from_dict({"text": test_texts, "label": test_labels})

```

Training and Tokenize - Hugging Face , Wandb AI-And API Key

```
from transformers import BertTokenizer, BertForSequenceClassification, Trainer, Training
import torch

# Load BERT tokenizer and model
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
model = BertForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=3)

# Tokenize datasets
def tokenize_function(examples):
    return tokenizer(examples["text"], padding="max_length", truncation=True)

train_dataset = train_dataset.map(tokenize_function, batched=True)
val_dataset = val_dataset.map(tokenize_function, batched=True)
test_dataset = test_dataset.map(tokenize_function, batched=True)

# Set format for PyTorch
train_dataset.set_format("torch", columns=["input_ids", "attention_mask", "label"])
val_dataset.set_format("torch", columns=["input_ids", "attention_mask", "label"])
test_dataset.set_format("torch", columns=["input_ids", "attention_mask", "label"])

# Training arguments
training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    weight_decay=0.01,
    save_total_limit=2,
    logging_dir='./logs',
    logging_steps=10,
)

# Define Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
)
import os
os.environ["WANDB_API_KEY"] = "your_40_character_api_key"

```

```
# Train the model
trainer.train()
```

```
Some weights of BertForSequenceClassification were not initialized from the model chec
You should probably TRAIN this model on a down-stream task to be able to use it for pr
```

Map: 100% 3/3 [00:00<00:00, 27.68 examples/s]

Map: 100% 1/1 [00:00<00:00, 13.65 examples/s]

Map: 100% 1/1 [00:00<00:00, 15.50 examples/s]

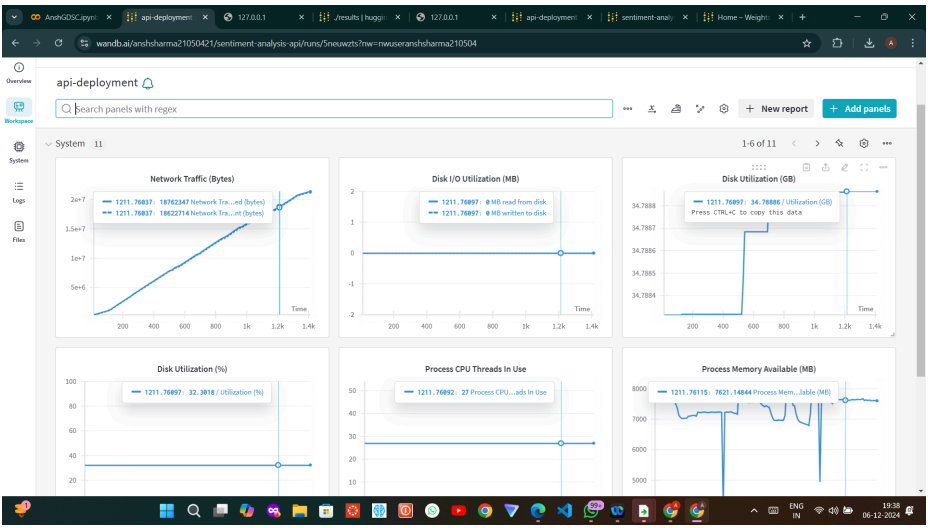
```
/usr/local/lib/python3.10/dist-packages/transformers/training_args.py:1568: FutureWarr
  warnings.warn(
```

Changes to your `wandb` environment variables will be ignored because your `wandb` session has already started. For more information on how to modify your settings with `wandb.init()` arguments, please refer to the W&B docs.

```
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter, or press ctrl+c to quit: ····
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
Tracking run with wandb version 0.18.7
Run data is saved locally in /content/wandb/run-20241206_133753-iynw8jpp
Syncing run ./results to Weights & Biases (docs)
View project at https://wandb.ai/anshsharma21050421/huggingface
View run at https://wandb.ai/anshsharma21050421/huggingface/runs/iynw8jpp
```

[3/3 01:06, Epoch 3/3]

Epoch	Training Loss	Validation Loss
1	No log	0.892518
2	No log	0.919221
3	No log	0.926235



```
# Evaluate the model on the test dataset
results = trainer.evaluate(test_dataset)

# Print evaluation results
print(f"Test Results: {results}")


```

[1/1 : < :]

Evaluation of Model

```
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix

# Get predictions and true labels
predictions = trainer.predict(test_dataset)
preds = torch.argmax(torch.tensor(predictions.predictions), axis=1).numpy()

# Calculate accuracy and F1 score
accuracy = accuracy_score(test_dataset['label'], preds)
f1 = f1_score(test_dataset['label'], preds, average="weighted")

# Print the metrics
print(f"Accuracy: {accuracy}")
print(f"F1 Score: {f1}")

# Confusion Matrix
cm = confusion_matrix(test_dataset['label'], preds)
print("Confusion Matrix:")
print(cm)

```

```
Accuracy: 1.0
F1 Score: 1.0
Confusion Matrix:
[[1]]
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:409: UserWa
```

```
model.save_pretrained("./fine_tuned_bert")
tokenizer.save_pretrained("./fine_tuned_bert")

```

```
('./fine_tuned_bert/tokenizer_config.json',
 './fine_tuned_bert/special_tokens_map.json',
 './fine_tuned_bert/vocab.txt',
 './fine_tuned_bert/added_tokens.json')
```

```
from transformers import BertForSequenceClassification, BertTokenizer

# Load the fine-tuned model and tokenizer
model = BertForSequenceClassification.from_pretrained("./fine_tuned_bert")
```

```
tokenizer = BertTokenizer.from_pretrained("./fine_tuned_bert")

```

Evaluation of Model by using Negative sentences and checking whether mode is predicting correct **sentiment**

```
# Example text
text = "Let's bomb NIT Calicut"

# Tokenize and prepare the input
inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_length

# Make prediction
with torch.no_grad():
    outputs = model(**inputs)

# Get predicted sentiment
predicted_class = torch.argmax(outputs.logits, dim=1).item()

# Map the class index to sentiment
sentiment = {0: "Negative", 1: "Neutral", 2: "Positive"}
print(f"Sentiment: {sentiment[predicted_class]}")

```

```
Sentiment: Negative
```

FAST API

```
pip install fastapi uvicorn transformers torch wandb
```

```
Collecting fastapi
  Downloading fastapi-0.115.6-py3-none-any.whl.metadata (27 kB)
Collecting uvicorn
  Downloading uvicorn-0.32.1-py3-none-any.whl.metadata (6.6 kB)
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.
Requirement already satisfied: wandb in /usr/local/lib/python3.10/dist-packages (0.
Collecting starlette<0.42.0,>=0.40.0 (from fastapi)
  Downloading starlette-0.41.3-py3-none-any.whl.metadata (6.0 kB)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,!=2.1.0,<3.0.0
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.1
Requirement already satisfied: click>=7.0 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: h11>=0.8 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in /usr/local/lib/pytho
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: tokenizers<0.21,>=0.20 in /usr/local/lib/python3.10/
```

```
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist
Requirement already satisfied: docker-pycreds>=0.4.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: gitpython!=3.1.29,>=1.0.0 in /usr/local/lib/python3.
Requirement already satisfied: platformdirs in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: protobuf!=4.21.0,!=5.28.0,<6,>=3.19.0 in /usr/local/
Requirement already satisfied: psutil>=5.0.0 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: sentry-sdk>=2.0.0 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: setproctitle in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: six>=1.4.0 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.10/
Requirement already satisfied: pydantic-core==2.27.1 in /usr/local/lib/python3.10/d
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.1
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist
Requirement already satisfied: anyio<5,>=3.4.0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.10/dist-pa
Downloading fastapi-0.115.6-py3-none-any.whl (94 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 94.8/94.8 kB 3.4 MB/s eta 0:00:00
Downloading uvicorn-0.32.1-py3-none-any.whl (63 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 63.8/63.8 kB 3.7 MB/s eta 0:00:00
Downloading starlette-0.41.3-py3-none-any.whl (73 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 73.2/73.2 kB 4.4 MB/s eta 0:00:00
Installing collected packages: uvicorn, starlette, fastapi
Successfully installed fastapi-0.115.6 starlette-0.41.3 uvicorn-0.32.1
```

```python
from fastapi import FastAPI
from pydantic import BaseModel
from transformers import BertForSequenceClassification, BertTokenizer
import torch
import wandb

# Initialize the FastAPI app
app = FastAPI()

# Load the fine-tuned model and tokenizer (replace with your saved model path)
model = BertForSequenceClassification.from_pretrained("./fine_tuned_bert")
tokenizer = BertTokenizer.from_pretrained("./fine_tuned_bert")

# Set up wandb (optional for experiment tracking)
wandb.init(project="sentiment-analysis-api", name="api-deployment")

# Define the input format using Pydantic
class Review(BaseModel):
    text: str

# Define the sentiment prediction endpoint
```

```
@app.post("/predict/")
def predict_sentiment(review: Review):
    # Tokenize input text
    inputs = tokenizer(review.text, return_tensors="pt", truncation=True, padding=True,

    # Make prediction
    with torch.no_grad():
        outputs = model(**inputs)

    # Get the predicted sentiment (choose the max logit)
    predicted_class = torch.argmax(outputs.logits, dim=1).item()

    # Map class index to sentiment label
    sentiment = {0: "Negative", 1: "Neutral", 2: "Positive"}
    predicted_sentiment = sentiment[predicted_class]

    # Log the prediction to wandb (optional)
    wandb.log({"review": review.text, "predicted_sentiment": predicted_sentiment})

    return {"review": review.text, "sentiment": predicted_sentiment}

```

Changes to your `wandb` environment variables will be ignored because your `wandb` session has already started. For more information on how to modify your settings with `wandb.init()` arguments, please refer to the W&B docs.
Finishing last run (ID:iynw8jpp) before initializing another...

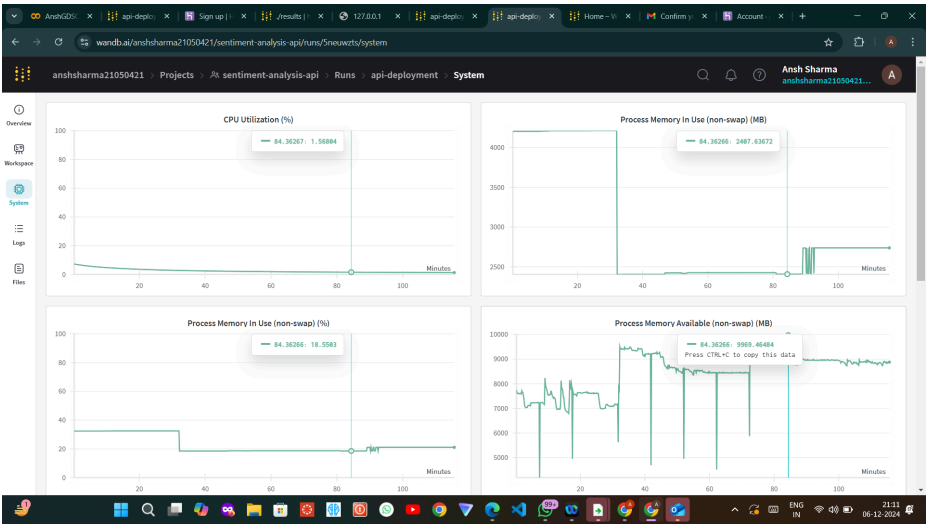
Run history:

eval/loss	▁▂▅▅
eval/runtime	▁▅▁▅
eval/samples_per_second	▅▁▅▅
eval/steps_per_second	▅▁▅▅
test/loss	▁▁
test/runtime	▅▁
test/samples_per_second	▁▅
test/steps_per_second	▁▅
train/epoch	▁▅▅▅
train/global_step	▁▅▅▅▅▅

Run summary:

eval/loss	1.02003
eval/runtime	3.4203
eval/samples_per_second	0.292
eval/steps_per_second	0.292
test/loss	1.02003
test/runtime	2.8443
test/samples_per_second	0.352
test/steps_per_second	0.352
total_flos	2368020759552.0
train/epoch	3
train/global_step	3
train_loss	1.08066
train_runtime	212.1883
train_samples_per_second	0.042
train_steps_per_second	0.014

View run **./results** at: https://wandb.ai/anshsharma21050421/huggingface/runs/iynw8jpp
View project at: https://wandb.ai/anshsharma21050421/huggingface
Synced 5 W&B file(s), 0 media file(s), 0 artifact file(s) and 0 other file(s)
Find logs at: `./wandb/run-20241206_133753-iynw8jpp/logs`
Successfully finished last run (ID:iynw8jpp). Initializing new run:
Tracking run with wandb version 0.18.7
Run data is saved locally in `/content/wandb/run-20241206_134511-5neuwzts`
Syncing run **api-deployment** to Weights & Biases (docs)




```
!curl https://cli-assets.heroku.com/install.sh | sh
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1980  100  1980    0     0   6701      0 --:--:-- --:--:-- --:--:--  6711
Installing CLI from https://cli-assets.heroku.com/channels/stable/heroku-linux-x64.tar
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 51.4M  100 51.4M    0     0  3716k      0  0:00:14  0:00:14 --:--:-- 3279k
v16.20.2
heroku installed to /usr/local/bin/heroku
 › Warning: Our terms of service have changed:
 › https://dashboard.heroku.com/terms-of-service
heroku/9.5.1 linux-x64 node-v16.20.2
```

```
!heroku login -i
```

```
heroku: Enter your login credentials
Email: anshsharma210504@gmail.com
Password: ********************************************************
```

```
!pip freeze > requirements.txt
```

```
from fastapi import FastAPI
from pydantic import BaseModel
from transformers import BertForSequenceClassification, BertTokenizer
import torch
import wandb
```

```
app = FastAPI()

model = BertForSequenceClassification.from_pretrained("./fine_tuned_bert")
tokenizer = BertTokenizer.from_pretrained("./fine_tuned_bert")

class Review(BaseModel):
    text: str

@app.post("/predict/")
def predict_sentiment(review: Review):
    inputs = tokenizer(review.text, return_tensors="pt", truncation=True, padding=True,
    with torch.no_grad():
        outputs = model(**inputs)
    predicted_class = torch.argmax(outputs.logits, dim=1).item()
    sentiment = {0: "Negative", 1: "Neutral", 2: "Positive"}
    predicted_sentiment = sentiment[predicted_class]
    wandb.log({"review": review.text, "predicted_sentiment": predicted_sentiment})
    return {"review": review.text, "sentiment": predicted_sentiment}

```

Pushing GIT Repository and *Heroku* Deployement

```
!git init
```

```
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /content/.git/
```

```
!git config --global user.email "ansh_b230825mt@nitc.ac.in"
!git config --global user.name "Anshsharmacse"
```

```
!git add .config/ fine_tuned_bert/ logs/ requirements.txt results/ sample_data/ wandb/

```

```
.config/
fine_tuned_bert
logs
requirements.txt
results
sample_data
```

```
!git commit -m "Initial commit"

```

```
[master (root-commit) 13426ee] Initial commit
 56 files changed, 84591 insertions(+)
 create mode 100644 .config/.last_opt_in_prompt.yaml
 create mode 100644 .config/.last_survey_prompt.yaml
 create mode 100644 .config/.last_update_check.json
 create mode 100644 .config/active_config
 create mode 100644 .config/config_sentinel
 create mode 100644 .config/configurations/config_default
 create mode 100644 .config/default_configs.db
 create mode 100644 .config/gce
 create mode 100644 .config/hidden_gcloud_config_universe_descriptor_data_cache_con
 create mode 100644 .config/logs/2024.12.04/14.22.43.568001.log
 create mode 100644 .config/logs/2024.12.04/14.23.06.536794.log
 create mode 100644 .config/logs/2024.12.04/14.23.17.977201.log
 create mode 100644 .config/logs/2024.12.04/14.23.19.373946.log
 create mode 100644 .config/logs/2024.12.04/14.23.31.005371.log
 create mode 100644 .config/logs/2024.12.04/14.23.31.672215.log
 create mode 100644 fine_tuned_bert/config.json
 create mode 100644 fine_tuned_bert/model.safetensors
 create mode 100644 fine_tuned_bert/special_tokens_map.json
 create mode 100644 fine_tuned_bert/tokenizer_config.json
 create mode 100644 fine_tuned_bert/vocab.txt
 create mode 100644 logs/events.out.tfevents.1733491855.223463f26f82.1619.0
 create mode 100644 logs/events.out.tfevents.1733492156.223463f26f82.1619.1
 create mode 100644 logs/events.out.tfevents.1733492454.223463f26f82.1619.2
 create mode 100644 requirements.txt
 create mode 100644 results/checkpoint-3/config.json
 create mode 100644 results/checkpoint-3/model.safetensors
 create mode 100644 results/checkpoint-3/optimizer.pt
 create mode 100644 results/checkpoint-3/rng_state.pth
 create mode 100644 results/checkpoint-3/scheduler.pt
 create mode 100644 results/checkpoint-3/trainer_state.json
 create mode 100644 results/checkpoint-3/training_args.bin
 create mode 100755 sample_data/README.md
 create mode 100755 sample_data/anscombe.json
 create mode 100644 sample_data/california_housing_test.csv
 create mode 100644 sample_data/california_housing_train.csv
 create mode 100644 sample_data/mnist_test.csv
 create mode 100644 sample_data/mnist_train_small.csv
 create mode 120000 wandb/debug-internal.log
 create mode 120000 wandb/debug.log
 create mode 120000 wandb/latest-run
 create mode 100644 wandb/run-20241206_133753-iynw8jpp/files/config.yaml
 create mode 100644 wandb/run-20241206_133753-iynw8jpp/files/output.log
 create mode 100644 wandb/run-20241206_133753-iynw8jpp/files/requirements.txt
 create mode 100644 wandb/run-20241206_133753-iynw8jpp/files/wandb-metadata.json
 create mode 100644 wandb/run-20241206_133753-iynw8jpp/files/wandb-summary.json
 create mode 120000 wandb/run-20241206_133753-iynw8jpp/logs/debug-core.log
 create mode 100644 wandb/run-20241206_133753-iynw8jpp/logs/debug-internal.log
 create mode 100644 wandb/run-20241206_133753-iynw8jpp/logs/debug.log
 create mode 100644 wandb/run-20241206_133753-iynw8jpp/run-iynw8jpp.wandb
 create mode 100644 wandb/run-20241206_134511-5neuwzts/files/output.log
 create mode 100644 wandb/run-20241206_134511-5neuwzts/files/requirements.txt
 create mode 100644 wandb/run-20241206_134511-5neuwzts/files/wandb-metadata.json
 create mode 120000 wandb/run-20241206_134511-5neuwzts/logs/debug-core.log
 create mode 100644 wandb/run-20241206_134511-5neuwzts/logs/debug-internal.log
 create mode 100644 wandb/run-20241206_134511-5neuwzts/logs/debug.log
```

```
!heroku create <Ansh Sentimental Analysis>
```

```
/bin/bash: -c: line 1: syntax error near unexpected token `newline'
/bin/bash: -c: line 1: `heroku create <Ansh Sentimental Analysis>'
```

```
!heroku apps
```

```
heroku: Press any key to open up the browser to login or q to exit:
```

```
!heroku create Anshsharmacse
```

```
Creating ⬢ Anshsharmacse...
Creating ⬢ Anshsharmacse... ⣾
Creating ⬢ Anshsharmacse... ?
heroku: Press any key to open up the browser to login or q to exit: Creating ⬢ Anshsh
```

```
!git remote add heroku https://git.heroku.com/Anshsharmacse.git
```

```
error: remote heroku already exists.
```

```
!git remote -v

```

```
heroku  https://git.heroku.com/your-app-name.git (fetch)
heroku  https://git.heroku.com/your-app-name.git (push)
```

```
!heroku login -i
```

```
heroku: Enter your login credentials
Email: anshsharma210504@gmail.com
Password: ****************************************
```

Final Evaluation

Steps

1. Write the Review Text in the review box
2. Click on Predict Sentiment

Note-->Refresh the cell on every new sentiment , to obtain the accurate result

```
# Import libraries
from transformers import BertForSequenceClassification, BertTokenizer
import torch
import ipywidgets as widgets
```

```
from IPython.display import display

# Load your fine-tuned model and tokenizer (replace with your model path if necessary)
model = BertForSequenceClassification.from_pretrained("./fine_tuned_bert")
tokenizer = BertTokenizer.from_pretrained("./fine_tuned_bert")

# Create a text box for user input
input_text = widgets.Textarea(
    value='',
    placeholder='Type your review here...',
    description='Review Text:',
    disabled=False,
    layout=widgets.Layout(width='50%', height='100px')
)

# Define a function to predict sentiment
def predict_sentiment(text):
    # Tokenize input text
    inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_le

    # Make prediction
    with torch.no_grad():
        outputs = model(**inputs)

    # Get predicted sentiment (choose the max logit)
    predicted_class = torch.argmax(outputs.logits, dim=1).item()

    # Map class index to sentiment label
    sentiment = {0: "Negative", 1: "Neutral", 2: "Positive"}
    predicted_sentiment = sentiment[predicted_class]

    # Display the result
    print(f"Predicted Sentiment: {predicted_sentiment}")

# Create a button to trigger the prediction
predict_button = widgets.Button(description="Predict Sentiment")

# Link the button to the prediction function
predict_button.on_click(lambda x: predict_sentiment(input_text.value))

# Display the input field and the button
display(input_text, predict_button)

```

Review Text: GDSC is the best club , Ansh Sharma will certainely be selected in GDSC Club.

Predict Sentiment

```
Predicted Sentiment: Positive
```