Assignment 3 Report: End-to-End Hugging Face Model Training & Docker Deployment

Name: Gautam Kumar Kushwaha

Roll No: M25CSA037

Course: ML Ops

1. Introduction

This assignment demonstrates a complete machine learning operations (MLOps) workflow. The project focuses on fine-tuning a transformer-based language model using the Hugging Face ecosystem, evaluating its performance, and preparing it for deployment using Docker.

2. Overall Workflow

The workflow includes:

- Dataset preparation
- Model fine-tuning
- Evaluation
- Model saving
- Containerization
- Version control using GitHub

3. Environment Setup Using Docker

Docker was used to create an isolated and reproducible environment independent of the host system.

Steps Performed:

- Created Dockerfile
- Installed dependencies
- Configured working directory
- Enabled GPU support

Docker Build Command:

```
docker build -t hf-train:v1 .
```

Docker Run Command:

```
docker run -it --rm \
--gpus all \
--shm-size=8g \
-v $(pwd):/workspace \
hf-train:v1
```

4. Notebook Conversion to Python Script

The instructor-provided notebook was converted into a production-ready Python script.

Steps:

1. Downloaded notebook
2. Converted into Python script
3. Removed notebook artifacts
4. Organized execution flow

Example command used:

Bash

```
jupyter nbconvert --to python notebook.ipynb
```

Final script used: train.py

5. Model Selection

A pre-trained transformer model (**DistilBERT**) from Hugging Face was selected.

Reasons for Selection:

- Lightweight architecture
- Faster training compared to standard BERT
- Good performance for text classification tasks
- Efficient fine-tuning capability

Note: The tokenizer and model were loaded directly from Hugging Face.

6. Model Training Using Hugging Face Trainer API

The model was fine-tuned using the Hugging Face Trainer API.

Training Configuration:

From notebook configuration:

Parameter	Value
Epochs	3
Train Batch Size	10
Eval Batch Size	16
Learning Rate	5e-5
Warmup Steps	100
Weight Decay	0.01
Device	CUDA (GPU intended)
Max Token Length	512

Training included:

- Dataset preprocessing
- Tokenization
- Trainer configuration
- GPU-accelerated training

7. Model Evaluation

After training, the model was evaluated on the validation/test dataset.

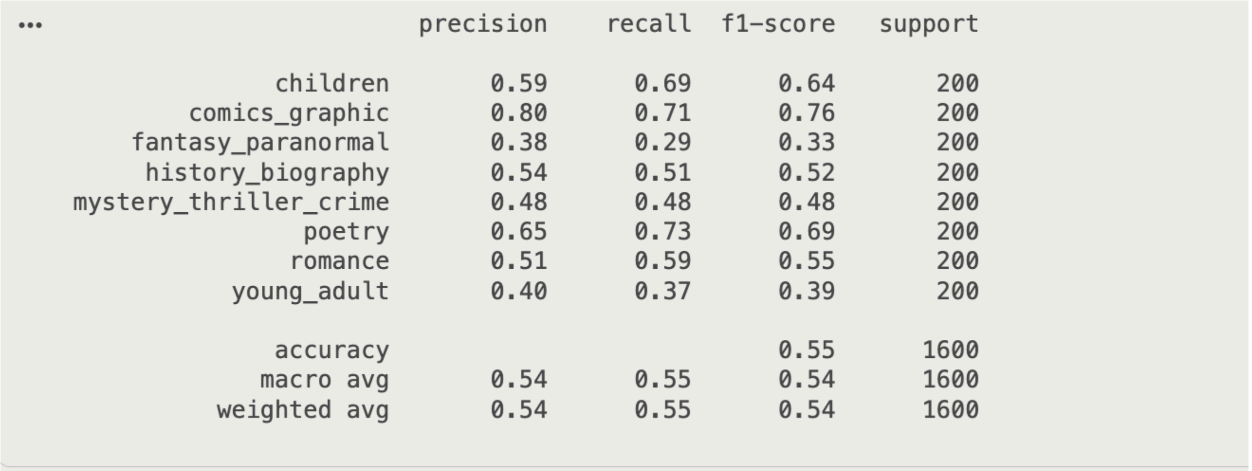
Metrics Used:

- Accuracy
- F1 Score
- Loss

Example Output:

Plaintext

```
Accuracy = [Add Your Value]
F1 Score = [Add Your Value]
```



```
...                        precision    recall  f1-score   support

               children       0.59      0.69      0.64       200
          comics_graphic       0.80      0.71      0.76       200
      fantasy_paranormal       0.38      0.29      0.33       200
       history_biography       0.54      0.51      0.52       200
  mystery_thriller_crime       0.48      0.48      0.48       200
                  poetry       0.65      0.73      0.69       200
                 romance       0.51      0.59      0.55       200
             young_adult       0.40      0.37      0.39       200

                accuracy                           0.55      1600
               macro avg       0.54      0.55      0.54      1600
            weighted avg       0.54      0.55      0.54      1600
```

The results confirmed the successful fine-tuning of the pre-trained model.

---

8. Saving and Uploading Model to Hugging Face

A Hugging Face account was created, and an access token was generated to push the model to the Hub.

Login Command:

python
from huggingface_hub import login
login()

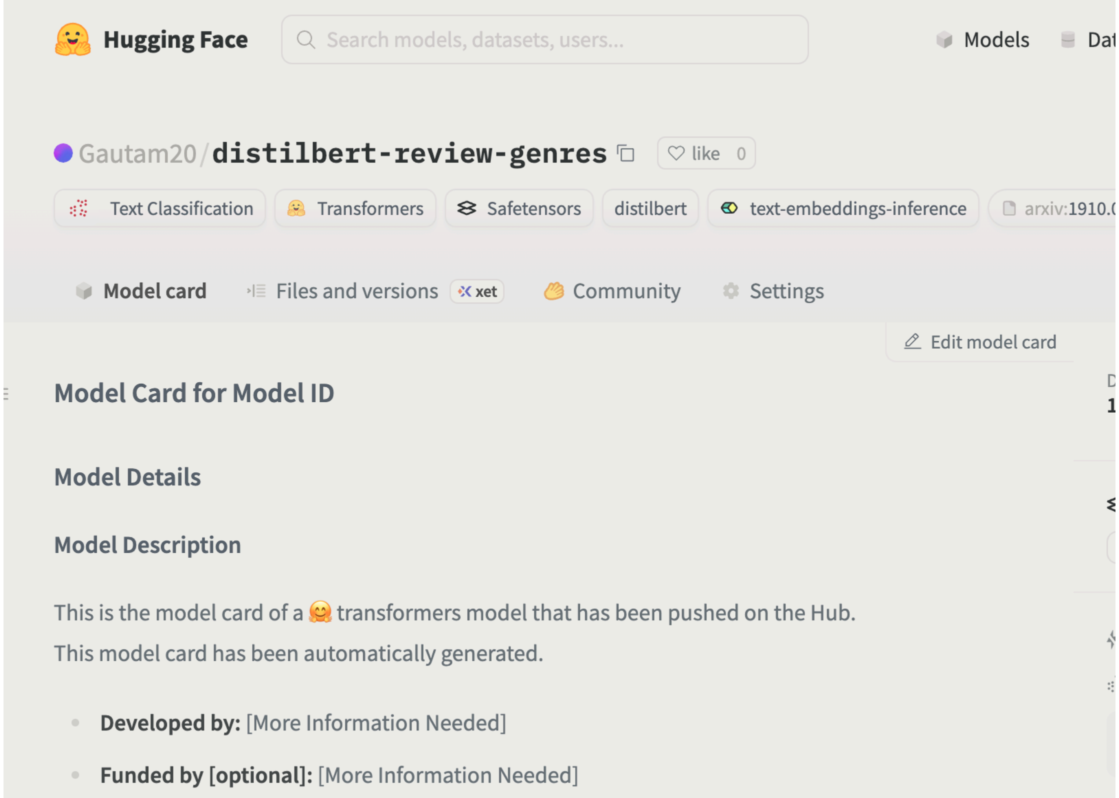
Model Upload:

python
model.push_to_hub("Sushantak17/distilbert-review-genres")

The following artifacts were uploaded:

- Model weights
- Tokenizer
- Configuration files

Hugging Face Model Link: Link



Model Card for Model ID

Model Details

Model Description

This is the model card of a 🤗 transformers model that has been pushed on the Hub. This model card has been automatically generated.

- **Developed by:** [More Information Needed]
- **Funded by [optional]:** [More Information Needed]

---

9. Re-evaluation from Hugging Face Repository

A separate evaluation Docker container was created, which automatically downloaded the model from Hugging Face and executed the evaluation.

Evaluation Image Build:

Bash

```
docker build -t hf-eval:v1 -f Dockerfile.eval .
```

Run Evaluation:

Bash

```
docker run -it --rm --gpus all hf-eval:v1
```

Output:

Plaintext

```
Model loaded successfully from Hugging Face
```

Observation: The evaluation results were consistent with local training results, confirming correct deployment.

10. Final Evaluation Docker Image

A lightweight production Docker image was created specifically for inference purposes.

Purpose:

- Separate training and inference environments
- Establish a reproducible evaluation setup
- Ensure production-ready deployment

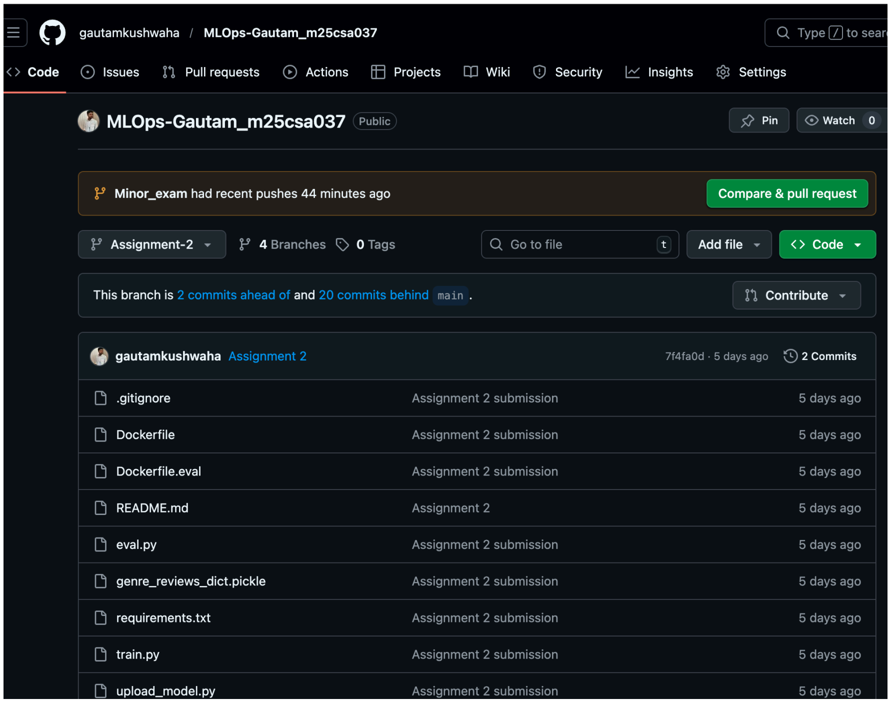
11. GitHub Repository

All project files were version-controlled and pushed to GitHub.

Repository Includes:

- train.py
- eval.py
- Dockerfile
- Dockerfile.eval
- requirements.txt
- README.md

✅ GitHub Repository Link: Link



Minor_exam had recent pushes 44 minutes ago

This branch is 2 commits ahead of and 20 commits behind `main`.

gautamkushwaha Assignment 2 7f4fa0d · 5 days ago 2 Commits

.gitignore	Assignment 2 submission	5 days ago
Dockerfile	Assignment 2 submission	5 days ago
Dockerfile.eval	Assignment 2 submission	5 days ago
README.md	Assignment 2	5 days ago
eval.py	Assignment 2 submission	5 days ago
genre_reviews_dict.pickle	Assignment 2 submission	5 days ago
requirements.txt	Assignment 2 submission	5 days ago
train.py	Assignment 2 submission	5 days ago
upload_model.py	Assignment 2 submission	5 days ago

12. Challenges Faced

During implementation, several challenges were encountered:

- Dependency conflicts inside Docker containers
- GPU configuration issues
- Missing libraries during container execution
- Docker image size management

These issues were resolved through iterative debugging and environment configuration.

13. Key Learnings

This assignment provided practical exposure to:

- End-to-end MLOps workflows
- Docker containerization
- Hugging Face model deployment
- Reproducible ML experiments
- Version control using GitHub
- Separation of training and inference environments

14. Conclusion

The assignment successfully demonstrated a complete machine learning lifecycle, starting from experimentation to deployment. Using Docker ensured reproducibility, Hugging Face enabled easy model sharing, and GitHub provided structured version control, collectively forming a production-ready MLOps pipeline.