Author: Deshna Shah
Project: Fine-tuned Sentiment Analysis with DistilBERT

Sentiment Analysis Project Report

1. Model Selection

Chosen Model: DistilBERT Base Uncased

Rationale:

- Efficiency: DistilBERT is 60% faster than BERT while retaining 97% of its language understanding capabilities
- Size: Reduced model size (66M parameters vs 110M in BERT) enables faster inference and lower resource requirements
- Performance: Proven effectiveness on sentiment analysis tasks with excellent accuracy
- Deployment: Smaller footprint makes it ideal for production deployment on standard hardware

Alternative Models Considered:

- BERT Base: Rejected due to larger size and slower inference
- RoBERTa: Rejected due to increased computational requirements
- LSTM-based models: Rejected due to inferior performance on context understanding

2. Fine-tuning Process

Training Environment

- Platform: Google Colab with GPU (T4)
- Notebook: `train_model_colab.ipynb`
- Hardware: CUDA-enabled GPU for accelerated training

Dataset

- Source: IMDB Movie Reviews Dataset from Stanford NLP (`stanfordnlp/imdb`)
- Split: 25,000 training samples, 25,000 test samples
- Classes: Binary classification (Positive/Negative)
- Total: 50,000 labeled movie reviews

Training Configuration

Base Model: distilbert-base-uncased

Learning Rate: 2e-5

Batch Size: 16

Epochs: 3

Optimizer: AdamW

Max Sequence Length: 512 tokens

Weight Decay: 0.01

Mixed Precision (FP16): Enabled for GPU

Evaluation Strategy: Per epoch

Training Phases

Phase 1: Standard Fine-tuning

- Trained on full IMDB dataset (25,000 reviews)
- Model saved to: `./sentiment-distilbert-imdb-final`
- Training time: ~20-30 minutes on GPU (T4)
- Training time on CPU: ~2-3 hours
- Evaluation metrics: Accuracy, Precision, Recall, F1 Score

Phase 2: Modern Language Enhancement (Bonus)

- Additional fine-tuning on modern slang and Gen Z language
- Custom dataset: 50 unique slang expressions × 100 repetitions = 5,000 samples
- Examples: "fire 🔥", "slaps", "no cap", "mid", "bussin", "giving vibes"
- Very low learning rate: 5e-6 (to preserve base knowledge)
- Training: 2 epochs only
- Model saved to: `./sentiment-distilbert-imdb-modern`
- Training time: ~5-10 minutes on GPU
- Enhanced handling of informal text and social media language

Training Metrics

- Training Loss: Steadily decreased across epochs
- Validation Accuracy: ~92% on IMDB test set
- Precision: High precision on both positive and negative classes

- Recall: Balanced recall across classes
- F1 Score: Strong F1 scores indicating good overall performance
- Convergence: Model converged successfully within 3 epochs

---

3. Deployment

Architecture

Project Structure:

```
├── train_model_colab.ipynb        # Training notebook (Google Colab)
├── models/
│   ├── sentiment-distilbert-imdb-final/     # Standard model
│   └── sentiment-distilbert-imdb-modern/    # Modern slang model
├── deployment/
│   ├── app.py                     # Gradio web interface
│   └── requirements.txt           # Dependencies
├── upload_model.py                # Upload model to HF Hub
├── upload_space.py                # Upload Space to HF
└── tests/                         # Test suite
```

Deployment Workflow

Step 1: Train Model in Google Colab

File: `train_model_colab.ipynb`

1. Open notebook in Google Colab
2. Connect to GPU runtime (T4)
3. Run all cells sequentially:
    - Install dependencies
    - Load IMDB dataset
    - Load DistilBERT model and tokenizer
    - Tokenize dataset
    - Configure training arguments
    - Train model (~20-30 minutes on GPU)
    - Evaluate performance
    - Test with sample reviews
    - Save model locally
    - (Bonus) Fine-tune on modern slang
4. Download trained model (zipped)

Step 2: Upload Model to Hugging Face Hub

File: `upload_model.py`

```
from huggingface_hub import HfApi, create_repo



username = "deshnaashok"

model_name = "sentiment-distilbert-imdb-modern"

model_path = "models/sentiment-distilbert-imdb-modern"









# Create repository
```

```
create_repo(
    repo_id=f"{username}/{model_name}",
    repo_type="model",
    exist_ok=True,
    private=False
)

# Upload model files
api = HfApi()
api.upload_folder(
    folder_path=model_path,
    repo_id=f"{username}/{model_name}",
    repo_type="model"
)
```

Model Hub: `https://huggingface.co/deshnaashok/sentiment-distilbert-imdb-modern`

Step 3: Create Gradio Web App

File: `deployment/app.py`

```
import gradio as gr
from transformers import pipeline



# Load model from Hugging Face Hub
```

```
sentiment_pipeline = pipeline(
    "sentiment-analysis",
    model="deshnaashok/sentiment-distilbert-imdb-modern",
    device=0 if torch.cuda.is_available() else -1
)

def analyze_sentiment(text):
    result = sentiment_pipeline(text[:512])[0]
    # Format and return sentiment, confidence, interpretation
    return sentiment_output, confidence_output, interpretation

# Create Gradio interface
demo = gr.Blocks(title="🎬 Movie Sentiment Analysis")
demo.launch()
```

Step 4: Deploy to Hugging Face Spaces

File: `upload_space.py`

```
from huggingface_hub import HfApi, create_repo

username = "deshnaashok"
space_name = "movie-sentiment-analysis"

# Create Space repository
create_repo(
    repo_id=f"{username}/{space_name}",

    repo_type="space",

    space_sdk="gradio",

    exist_ok=True

)


# Upload app files

api.upload_folder(

    folder_path=".",

    repo_id=f"{username}/{space_name}",

    repo_type="space"

)
```

Live Space:
`https://huggingface.co/spaces/deshnaashok/movie-sentiment-analysis`

Deployment Features

Gradio Web Interface:

- 🎨 Beautiful, modern UI with color-coded results
- 💬 Large text input area for movie reviews
- 🎯 Real-time sentiment prediction (Positive/Negative)
- 📊 Confidence scores with visual indicators (🟢🟡🟠🔴)
- 💡 Interpretation explaining the prediction
- 📝 Example reviews for quick testing
- 😊😞 Emoji indicators for sentiment
- 🚀 Fast inference (~100-150ms per prediction)

Technical Features:

- Direct loading from Hugging Face Hub
- Fallback model handling
- GPU acceleration when available

- Token limit handling (512 tokens)
- Error handling and user feedback
- Mobile-responsive design

Technical Stack

- Framework: Transformers (Hugging Face)
- Web Framework: Gradio 4.x
- Deep Learning: PyTorch
- Language: Python 3.12
- Deployment: Hugging Face Spaces
- Training Platform: Google Colab (GPU)

4. Test Results and Observations

Functional Testing

Test Case 1: Positive Review - Strong Sentiment

Input: "This movie was absolutely amazing! Best film of the year!"

Result: POSITIVE (99.87% confidence)

Observation: Model shows very high confidence on clear positive sentiment

Status: ✅ PASS

Test Case 2: Negative Review - Strong Sentiment

Input: "Terrible waste of time. I want my money back."

Result: NEGATIVE (98.45% confidence)

Observation: Accurately detects strong negative sentiment

Status: ✅ PASS

Test Case 3: Modern Slang - Positive

Input: "OMG this film was fire 🔥 So good!"

Result: POSITIVE (96.32% confidence)

Observation: Successfully understands modern slang and emojis

Status: ✅ PASS (Enhanced by Phase 2 training)

Test Case 4: Modern Slang - Negative

Input: "This movie was mid tbh"

Result: NEGATIVE (85.20% confidence)

Observation: Correctly interprets Gen Z term "mid" as negative

Status: ✅ PASS (Enhanced by Phase 2 training)

Test Case 5: Mixed Sentiment

Input: "The plot was great but the ending disappointed me."

Result: NEGATIVE (65.42% confidence)

Observation: Lower confidence indicates mixed sentiment detection

Status: ⚠️ PASS (Confidence appropriately lower for ambiguous text)

Test Case 6: Neutral/Ambiguous

Input: "It was okay, I guess."

Result: POSITIVE (58.12% confidence)

Observation: Low confidence on neutral statement (binary classification limitation)

Status: ⚠️ ACCEPTABLE (No neutral class available)

Performance Metrics

Model Performance:

- Test Set Accuracy: ~92% on IMDB dataset
- Precision: ~91% (both classes)
- Recall: ~92% (balanced)
- F1 Score: ~91.5%

Inference Speed:

- CPU: ~100-150ms per prediction
- GPU: ~10-50ms per prediction
- Web App Response: <500ms total (including network)
- Suitable for real-time applications

Model Size:

- Model weights: ~268MB (safetensors format)
- Tokenizer files: ~711KB
- Total deployment size: ~270MB
- Memory usage: ~1GB RAM during inference

Deployment Testing

Hugging Face Space Performance:

- ✅ Successful deployment to HF Spaces
- ✅ Auto-scaling and zero-downtime updates
- ✅ Global CDN for fast access
- ✅ Mobile-responsive interface
- ✅ Example inputs working correctly
- ✅ Confidence indicators displaying properly
- ✅ Error handling functioning as expected

Key Observations

Strengths:

1. High Accuracy: Excellent performance on movie reviews (92%)
2. Fast Inference: Sub-second predictions suitable for production
3. Modern Language: Successfully handles slang, emojis, and informal text
4. User Experience: Beautiful Gradio interface with clear visualizations
5. Confidence Scores: Transparent predictions with confidence levels
6. Robust Handling: Graceful degradation with fallback model
7. Deployment: Seamless Hugging Face Spaces integration

Limitations:

1. Binary Classification: No neutral category (0-negative, 1-positive only)

2. Domain Specificity: Best for movie reviews, may need adaptation for other domains
3. Language: English only
4. Context Length: Limited to 512 tokens (~350-400 words)
5. Ambiguous Text: Lower confidence on mixed or neutral sentiments
6. Sarcasm: May struggle with heavy sarcasm or irony

Edge Cases:

- Very short inputs (1-2 words): May have inflated confidence
- Mixed sentiments: Appropriately shows lower confidence
- Emojis only: Handles reasonably but context helps
- Non-English: Will attempt prediction but unreliable

Recommendations for Production

Immediate Use:

- ✅ Movie review sentiment analysis
- ✅ Product review classification
- ✅ Social media sentiment monitoring
- ✅ Customer feedback analysis

Improvements for the Future:

- Add neutral sentiment class (3-class classification)
- Implement batch processing for high-volume scenarios
- Add multi-language support
- Domain-specific fine-tuning for other review types
- Enhanced sarcasm detection
- Real-time feedback collection for continuous improvement
- A/B testing framework for model updates

5. Conclusion

The fine-tuned DistilBERT model successfully achieves high-accuracy sentiment analysis (92%) with efficient resource utilization on the IMDB movie review dataset. The project demonstrates a complete end-to-end machine learning pipeline from training to deployment.

Deliverables

✅ Training Pipeline: Complete Jupyter notebook (`train_model_colab.ipynb`) with GPU optimization
✅ Trained Models: Two model versions (standard + modern slang enhanced)
✅ Web Application: Production-ready Gradio app (`deployment/app.py`)
✅ Model Repository: Published on Hugging Face Hub (`deshnaashok/sentiment-distilbert-imdb-modern`)
✅ Live Demo: Deployed to Hugging Face Spaces (`deshnaashok/movie-sentiment-analysis`)
✅ Upload Scripts: Automated scripts for model and space deployment
✅ Documentation: Comprehensive project report with technical details
✅ Test Suite: Functional tests with various edge cases

Links

🔗 GitHub Repository: https://github.com/desshah/Huggingface
🤗 Model Hub: https://huggingface.co/deshnaashok/sentiment-distilbert-imdb-modern
🚀 Live Demo: https://huggingface.co/spaces/deshnaashok/movie-sentiment-analysis

Impact & Applications

This sentiment analysis system can be applied to:

- Movie Reviews: Analyze audience reactions and ratings
- Product Reviews: Understand customer satisfaction on e-commerce platforms
- Social Media: Monitor brand sentiment and public opinion
- Customer Support: Automatically classify feedback and prioritize responses
- Content Moderation: Identify negative or toxic content
- Market Research: Analyze consumer sentiment toward products/services