Text Classification Transformer

Tim Hanlon

April 22, 2025

Project Overview

Goal:
Classify Red Sox baseball commentary as **positive** or **negative** using a fine-tuned BERT model.

Technologies Used:

- Hugging Face Transformers
- Hugging Face Datasets
- PyTorch
- BERT (bert-base-uncased)

Dataset

Source: `redsox_commentary.csv`
Features:

- `timestamp`: Date and time of the comment
- `comment`: Text of the fan commentary
- `sentiment`: Sentiment label (positive/negative)

Label Distribution:

- Negative: 55 comments
- Positive: 45 comments

Workflow Overview

- Load and encode the data
- Tokenize text using BERT tokenizer
- Convert to Hugging Face Dataset
- Train/test split
- Load and fine-tune BERT
- Evaluate and save model

Data Processing

Read CSV using pandas

Map `sentiment` to numeric labels:

- Positive → 1
- Negative → 0

Check distribution using `value_counts()`

- label_map = {'negative': 0, 'positive': 1}
- df['label'] = df['sentiment'].map(label_map)

Tokenization

- Loaded BERT tokenizer (`bert-base-uncased`)
- Applied padding and truncation
- Used `map()` for batch tokenization with Hugging Face Datasets
- def tokenize(batch):
-     return tokenizer(batch['comment'], padding=True, truncation=True)

Dataset Conversion

- Converted `DataFrame` → Hugging Face `Dataset`
- Removed unnecessary columns
- Set format for PyTorch tensors
- dataset.set_format(type='torch', columns=['input_ids', 'token_type_ids', 'attention_mask', 'label'])

Training

80% Training, 20% Testing

Total samples: 100

Training: 80

Testing: 20

```
split_dataset = dataset.train_test_split(test_size=0.2)
```

Model Setup

- Loaded `BertForSequenceClassification`
- Configured for 2 sentiment classes
- Initialized Trainer with model, datasets, and training arguments
- model = BertForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=2)

Training Arguments

Key Settings:

- Learning Rate: `2e-5`
- Batch Size: 8
- Epochs: 3
- Weight Decay: 0.01
- Logging: Enabled

Model Training

Used `Trainer` API to handle training

3 Epochs completed in ~30 seconds

Progress printed during training

```
trainer.train()
```

Evaluation and Results

Evaluation Loss: ~0.25

Runtime: ~0.5 sec

Test Accuracy: Good performance on small dataset

```
results = trainer.evaluate(test_dataset)
```

Saving Model

Saved trained model locally for reuse

trainer.save_model("./sentiment_model")

How to Use:

- Load saved model
- Tokenize new comments
- Use `model(**inputs)` to predict sentiment

Conclusion

Fine-tuned BERT for Red Sox sentiment classification

Solid performance with small dataset

Easily extendable to larger datasets or different domains

Questions

- Thank you for past 6 months everyone!
- Any questions?