Essay Classification Using BERT

Abstract

Evaluating essays is a subjective and labor-intensive process. This project introduces an automated essay scoring system based on Natural Language Processing (NLP) techniques, particularly using the BERT model. By leveraging pre-trained transformer-based architectures, we aim to make essay evaluation more consistent, fair, and efficient. The system fine-tunes BERT for sequence classification tasks, demonstrating the potential of modern NLP techniques in automating text evaluation.

Introduction

Essay scoring traditionally relies on human evaluators, which can lead to inconsistency and delays. With advancements in NLP, it is now possible to train models capable of accurately and consistently assigning scores to essays. This project focuses on fine-tuning a pre-trained BERT model to automate the scoring process, reducing the subjectivity inherent in manual grading.

Problem Statement

Manual essay scoring is:

- **Time-consuming**: Evaluating a large number of essays takes considerable time.
- **Subjective**: Scores may vary between evaluators.
- **Resource-intensive**: Requires expert evaluators for consistent grading.

The objective is to build an automated essay scoring system that:

1. Assigns scores to essays accurately and consistently.
2. Reduces the time and resources needed for evaluation.
3. Scales to large datasets of essays.

Timeline

WEEK	TASK
1-2	Problem definition, dataset collection.
3-4	Data preprocessing, feature extraction
5-6	Model development, tuning, and evaluation.
7-8	Deployment, testing, and documentation

Dataset

Source

The dataset used in this project is publicly available https://osf.io/9fdrw/ . It consists of essays written by students, with each essay accompanied by scores from two human raters.

Key Features

- **Columns**:
  - **essay_id**: Unique identifier for each essay.
  - **essay_set**: Indicates the topic or prompt for the essay.
  - **essay**: The text of the essay.
  - **rater1_domain1**, **rater2_domain1**: Scores assigned by two human raters.

Preprocessing Steps

1. Cleaning text:
   a. Convert to lowercase.
   b. Remove special characters, stopwords, and excess whitespace.
2. Tokenization using BERT's tokenizer to prepare data for input into the model.
3. Handling labels: Combining scores from the two raters to create a unified target variable.

Code Snippet:

```
df = pd.read_excel('/content/training_set_rel3.xlsx')
columns = ['essay_id', 'essay_set', 'essay', 'rater1_domain1', 'rater2_domain1']
df = pd.DataFrame(df, columns=columns)
```

Methodology

Data Preprocessing

- **Text Cleaning**: Removes noise and standardizes the text.
- **Tokenization**: Converts essays into tokenized sequences using **BertTokenizer**.
- **Padding and Truncation**: Ensures all sequences are of equal length (512 tokens).

Code Snippet:

```
from transformers import BertTokenizer

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
tokens = tokenizer(
   text_list,  # List of essay texts
   max_length=512,
   truncation=True,
   padding="max_length",
   return_tensors="tf"
)
```

Model

The model used is **TFBertForSequenceClassification**, a pre-trained BERT model fine-tuned for essay scoring tasks.

Code Snippet:

```
from transformers import TFBertForSequenceClassification

model = TFBertForSequenceClassification.from_pretrained('bert-base-uncased')
```

Training and Optimization

- **Train-Test Split**: 80% training, 20% testing.
- **Optimizer**: Adam optimizer with a learning rate of **5e-5**.
- **Loss Function**: Cross-entropy loss.
- **Evaluation Metric**: Accuracy.

Code Snippet:

```
optimizer = tf.keras.optimizers.Adam(learning_rate=5e-5)
model.compile(optimizer=optimizer, loss=model.compute_loss, metrics=['accuracy'])
model.fit(train_data, train_labels, validation_data=(test_data, test_labels), epochs=3)
```

Analysis and Conclusion

Insights

1. Transformer-based models like BERT excel at capturing the nuances of textual data.
2. Cleaning and preprocessing are essential for optimal model performance.
3. Fine-tuning a pre-trained model significantly reduces training time and improves accuracy.

Challenges

- Processing long essays requires truncation, potentially losing context.
- Requires significant computational resources, including GPUs.

Future Directions

- Experiment with other transformer architectures (e.g., RoBERTa, GPT).
- Explore explainable AI methods for better interpretability of the scoring process.
- Build a robust deployment pipeline for real-world applications.

Deployment and Demo

The model can be deployed as a web-based interface using frameworks like Flask or Streamlit. Users can upload essays to receive automated scores in real time.