Sentiment Analysis with BERT: A Comprehensive Report

Author: Uru Onyemaobi

Introduction

This report demonstrates the use of the BERT model for sentiment analysis. My goal was to analyze reviews and determine their sentiment, whether positive, neutral, or negative. The steps involved installing dependencies, loading the model, encoding text, and calculating sentiment scores.

1. Install and Import Dependencies

To begin, we need to install the necessary dependencies. This includes libraries such as PyTorch, Transformers, Requests, BeautifulSoup, Pandas, and NumPy.

Python

```
!pip install torch torchvision torchaudio --index-url
https://download.pytorch.org/whl/cu121 #pytorch
!pip install transformers requests beautifulsoup4 pandas numpy
```

Next, we import the required modules.

Python

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch
import requests
from bs4 import BeautifulSoup
import re
import numpy as np
import pandas as pd
```

2. Instantiate the Model

We instantiate the tokenizer and the model from the Hugging Face library. Here, we use the `nlptown/bert-base-multilingual-uncased-sentiment` model, which is pre-trained for sentiment analysis.

```python
tokenizer =
AutoTokenizer.from_pretrained('nlptown/bert-base-multilingual-uncased-sent
iment')
model =
AutoModelForSequenceClassification.from_pretrained('nlptown/bert-base-mult
ilingual-uncased-sentiment')
```

3. Encode and Calculate Sentiment

A. Encode the Text

We pass a string or prompt to our tokenizer and encode it to obtain token tensors.

```python
uruTokens = tokenizer.encode("A sample text for sentiment analysis",
return_tensors='pt')
```

B. Tokenize and Classify

We tokenize the input text and pass it through the model to get the classification.

```python
result = model(uruTokens)
sentiment = torch.argmax(result.logits) + 1 # Adding 1 to match the
sentiment scale
```

C. Sample Sentiment Analysis

```python
sample_text = "The product was excellent and I loved it!"
sample_tokens = tokenizer.encode(sample_text, return_tensors='pt')
sample_result = model(sample_tokens)
sample_sentiment = torch.argmax(sample_result.logits) + 1
```

4. Collect Reviews

We use the Requests and BeautifulSoup libraries to scrape reviews from a website. In this example, reviews are extracted from a Yelp page.

```python
r = requests.get('https://www.yelp.com/biz/southern-suya-atlanta')
soup = BeautifulSoup(r.text, 'html.parser')
regex = re.compile('.*comment.*')
results = soup.find_all('p', {'class': regex})
reviews = [result.text for result in results]
```

5. Load Reviews into a DataFrame and Score Them

We load the collected reviews into a Pandas DataFrame for further processing and sentiment scoring.

Python

```python
df = pd.DataFrame(np.array(reviews), columns=['Review'])
```

We define a function to compute the sentiment score for any review.

Python

```python
def sentiment_score(anyreview):
  urutokens = tokenizer.encode(anyreview, return_tensors="pt")
  result = model(urutokens)
  return int(torch.argmax(result.logits)) + 1
```

We apply this function to each review in the DataFrame to compute their sentiments.

Python

```python
df['sentiment'] = df['Review'].apply(lambda x: sentiment_score(x[:512]))
```

Conclusion

This report outlines the steps to perform sentiment analysis using the BERT model. The core reason for taking on this project was the need to get a better feel for how customers were reacting to service at African restaurants in the Atlanta area. The process involved installing dependencies, loading a pre-trained model, encoding text, and computing sentiment scores. The results are stored in a DataFrame, showcasing the sentiment of each review.

By following these steps, it’s been possible to provide valuable insights into customer opinions and feedback.