Intern Report

Submitted By Ankit Lal

Introduction

The Maritime News Sentiment Analysis project aims to determine the sentiment of maritime news articles using machine learning. Sentiment analysis is a valuable tool for understanding public opinion, market trends, and potential impacts on the maritime industry.

Dataset

The dataset for this project comprises maritime news articles. The articles were scraped from a maritime news website using Selenium, a popular web scraping tool. The dataset is stored in a CSV file with a column named Content, which contains the text of the news articles.

Web Scraping with Selenium

Finding the Website Structure

Before scraping, it was essential to understand the structure of the target website. The target website was analyzed to locate the HTML elements containing the news articles. This was done using browser developer tools to inspect the webpage structure.

Key steps involved:

1. **Identify the URL**: The URL of the webpage containing the news articles was identified.
2. **Inspect Elements**: The elements containing the article text were inspected using browser developer tools (e.g., Chrome DevTools).
3. **Determine Patterns**: Patterns in the HTML structure were determined to accurately locate and extract the required data.

Scraping the Data

Selenium was used to automate the web browser and scrape the news articles. The following Python script was employed:

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service as ChromeService
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
```

```
import pandas as pd

# Set up Selenium WebDriver
chromedriver_path = '/usr/bin/chromedriver'
options = Options()
options.add_argument('--headless')  # Run in headless mode
service = ChromeService(executable_path=chromedriver_path,
log_path='chromedriver.log')

driver = webdriver.Chrome(service=service, options=options)

# URL of the maritime news website
url = 'https://www.marinetraffic.com/en/maritime-news'
driver.get(url)

# Locate and extract news articles
articles = driver.find_elements(By.CLASS_NAME, 'article-class')  # Replace
with actual class name
data = []
for article in articles:
    title = article.find_element(By.CLASS_NAME, 'title-class').text  # Replace
with actual class name
    content = article.find_element(By.CLASS_NAME, 'content-class').text  #
Replace with actual class name
    data.append({'Title': title, 'Content': content})

driver.quit()

# Save data to CSV
df = pd.DataFrame(data)
df.to_csv('maritime_news.csv', index=False)
```

Model

Model Selection

For sentiment analysis, the distilbert-base-uncased-finetuned-sst-2-english model from Hugging Face was selected. This model is a smaller, faster, and cheaper version of BERT (Bidirectional Encoder Representations from Transformers) and has been fine-tuned on the Stanford Sentiment Treebank (SST-2) dataset for sentiment classification.

Tokenization

Tokenization is the process of converting text into tokens that the model can understand. The selected model uses a tokenizer that splits the input text into tokens and converts them

into numerical IDs. The tokenizer ensures that the input text is in a format compatible with the model's architecture.

```
from transformers import DistilBertTokenizerFast

# Initialize the tokenizer
tokenizer = DistilBertTokenizerFast.from_pretrained('distilbert-base-uncased-finetuned-sst-2-english')

# Example of tokenization
sample_text = "Maritime transport plays a crucial role in global trade."
tokens = tokenizer.tokenize(sample_text)
input_ids = tokenizer.convert_tokens_to_ids(tokens)

print(tokens)
print(input_ids)
```

Sentiment Analysis

The sentiment analysis was performed using the pre-trained **distilbert-base-uncased-finetuned-sst-2-english** model. The pipeline function from the Hugging Face library was used to simplify the process.

```
from transformers import pipeline
import pandas as pd

# Load the CSV file
df = pd.read_csv('maritime_news.csv')

# Initialize the sentiment analysis pipeline
sentiment_pipeline = pipeline('sentiment-analysis', model='distilbert-base-uncased-finetuned-sst-2-english')

# Function to get sentiment for each article
def get_sentiment(text):
    result = sentiment_pipeline(text[:512])  # Limiting text to 512 tokens
    return result[0]['label']

# Apply sentiment analysis to the 'Content' column
df['Sentiment'] = df['Content'].apply(get_sentiment)

# Save results to a new CSV file
df.to_csv('maritime_news_with_sentiment.csv', index=False)

print(f'Sentiment analysis results saved to maritime_news_with_sentiment.csv')
```

Results

The output CSV file (maritime_news_with_sentiment.csv) contains the original news articles along with a new column named Sentiment, indicating whether the sentiment of each article is positive or negative.

Conclusion

The Maritime News Sentiment Analysis project successfully demonstrated how to scrape maritime news articles, process them using a pre-trained language model, and classify their sentiment. The project highlights the effectiveness of using pre-trained models for natural language processing tasks and the importance of understanding the website's structure for accurate data extraction.

References

1. Hugging Face Transformers Library: https://huggingface.co/transformers/
2. Selenium Documentation: https://www.selenium.dev/documentation/
3. DistilBERT Model: https://arxiv.org/abs/1910.01108
4. Stanford Sentiment Treebank (SST-2): https://nlp.stanford.edu/sentiment/index.html

Github Repo link

https://github.com/ankitlal-iitr/Workfile.git