Sentiment Analysis System for News Paragraphs.

Introduction

This project aims to develop a **Sentiment Analysis System** by scraping news paragraph from **lite.cnn.com**, a lightweight version of CNN's news platform known for its concise reporting. The system will classify news sentiments into **Positive**, **Neutral**, and **Negative** categories, providing valuable insights into media coverage.

Project Workflow

The project follows these key steps:

1. **Data Collection** – Scraping 200+ news paragraphs from **lite.cnn.com**.
2. **Data Annotation** – Assigning sentiment labels to each paragraph.
3. **Model Training** – Developing machine learning models for sentiment classification.
4. **Deployment** – Exposing the trained model via **FastAPI** as a REST API for real-time sentiment analysis.
5. **Optimization & A/B Testing** – Enhancing model efficiency and evaluating performance using A/B testing.

Data Collection & Cleaning

The data collection process involves scraping news paragraphs from CNN's lightweight platform and storing them in CSV files. The key steps include:

- Using **BeautifulSoup** to extract news sentences from articles.
- Storing the extracted data in a structured CSV format.
- Removing unwanted data (first 3 rows and last 7 rows from each dataset).
- Merging all datasets into a single DataFrame for analysis.

To ensure data quality, we checked for missing values and duplicates before proceeding to the next step.

Data Preprocessing & Annotation

To enhance our dataset, we:

- Applied **TextBlob** for sentiment analysis to label sentences as Positive, Neutral, or Negative.
- Balanced the dataset using **resampling** to ensure an equal distribution of sentiment classes.
- Stored the annotated data for model training.

Model Training

We trained and evaluated multiple machine learning models, including:

1. **Naive Bayes (Baseline Model)** – A simple yet effective classifier for text-based sentiment analysis.
2. **Logistic Regression (Optimized Model)** – Improved accuracy with hyperparameter tuning.

Training Steps

- Split the dataset into training and testing sets.
- Converted text data into numerical form using **TF-IDF Vectorization**.
- Tuned hyperparameters using **GridSearchCV**.
- Evaluated models using **classification reports**.
- Saved the best-performing models and vectorizer for deployment.

Model Deployment & A/B Testing

To enable real-time sentiment predictions, we deployed the models using **FastAPI**. Key features include:

- **REST API Endpoint:** Accepts text input and returns the predicted sentiment.
- **A/B Testing Implementation:**
    - Randomly selects between the **Naive Bayes** and **Optimized Model** (Logistic Regression or Random Forest) for prediction.
    - Compares performance over time to determine the best model.

Conclusion

This project successfully implemented a sentiment analysis system for news paragraphs. It leveraged web scraping, machine learning, and API deployment to provide real-time sentiment classification. Future enhancements may include:

- Expanding the dataset to improve model generalization.
- Exploring deep learning techniques for sentiment classification.
- Deploying the system on cloud platforms for scalability.

This system was created as a task for NITHUB internship second phase application.