Sentiment Analysis Web App

This repository contains a **Sentiment Analysis Web App** built using **DistilBERT**, open-source datasets, and custom methodologies. The app is designed to predict sentiments (Positive, Neutral, Negative) for a given text input. Below, we describe the development process, challenges faced, datasets used, and the final implementation.

Table of Contents

Project Overview

The goal of this project was to build a web app for sentiment analysis. Initially, we experimented with a custom dataset that was **unreliable** due to:

- Repetitive sentences
- Inconsistent annotations
- Limited coverage of sentiment variability

Despite these challenges, we explored the dataset through **visualization** and **preprocessing** steps and attempted to fine-tune **DistilBERT** for sentiment classification. However, due to poor dataset quality, the training process yielded suboptimal results.

To overcome these issues, we switched to an open-source Twitter Sentiment Dataset, which provided high-quality, pre-annotated sentiment data with better class balance. This dataset enabled us to experiment with and implement more robust methodologies and achieve improved model performance.

Datasets

Initial Custom Dataset

Our initial attempts used a custom dataset which proved unreliable for effective training due to:

- Repetitive content
- Inconsistent sentiment annotations
- Insufficient variability in language and expressions

Open-Source Twitter Sentiment Dataset

We transitioned to using an **open-source Twitter Sentiment Dataset** from Kaggle:

- High-quality, pre-annotated sentiment data
- Balanced representation of Positive, Neutral, and Negative sentiments
- Greater diversity in language and expressions
  This dataset formed the backbone of our final models and experiments.

Methodologies

We experimented with the following approaches:

1. **Data Cleaning & Preprocessing:**
   - Removed stopwords, URLs, special characters, and repetitive phrases.
   - Performed tokenization and lemmatization using NLTK to clean and standardize input text.
2. **Model Exploration:**
   - Initially attempted to fine-tune **DistilBERT** for sentiment classification on our custom dataset.
   - Evaluated models using metrics like *accuracy*, *F1 score*, and *confusion matrix*.
   - Due to poor dataset quality, these efforts were re-oriented towards better quality data.
3. **Model Training on Twitter Sentiment Dataset:**
   - Fine-tuned **DistilBERT** using Hugging Face Transformers on the reliable Twitter dataset.
   - Explored simpler architectures such as **BiLSTM with an Attention Layer** as a comparison.
   - Evaluated and compared model performances to ensure robust sentiment predictions.
4. **Web App Integration:**
   - Saved the trained models (`.h5` for BiLSTM+Attention and `.pkl` for tokenizer) for seamless use in the web app.
   - Built an interactive **Streamlit-based app** to allow users to input text and view sentiment predictions accompanied by random emoji animations.

Key Challenges

1. **Dataset Quality:**
   - The initial custom dataset was unreliable, leading to poor training outcomes.
   - Transitioning to an open-source Twitter dataset significantly improved model reliability and performance.
2. **Model Performance:**
   - Fine-tuning **DistilBERT** required considerable computational resources and careful hyperparameter tuning.
   - Simpler architectures like **BiLSTM with an Attention Layer** provided competitive results with fewer resources and faster iteration cycles.

3. **Integration Issues:**
    - Ensuring compatibility between custom layers (e.g., `AttentionLayer`), saved model files (`.h5`), and tokenizer files (`.pkl`) was challenging.
    - Careful configuration and testing were necessary to integrate these components into the web app smoothly.

Final Implementation

The final implementation includes:

1. **Training Notebooks:**
    - `finallynotebook.ipynb` contains the code for training both DistilBERT and BiLSTM models on the Twitter Sentiment Dataset.
    - Detailed EDA, preprocessing, model training, evaluation, and comparison steps are documented in the notebook.
2. **Model Saving:**
    - Trained models are saved for deployment:
        - `bilstm_att_model.h5` (BiLSTM with Attention)
        - `tokenizer.pkl` (Tokenizer for preprocessing)
3. **Web App:**
    - A **Streamlit-based interactive web app** where users can input text and receive sentiment analysis.
    - Features include:
        - Random floating emoji animations for engaging UI feedback.
        - Real-time sentiment prediction labeled as Positive, Neutral, or Negative.

How to Run

1. **Clone the Repository:**

```
git clone https://github.com/Mahos-H/LetUsTalk
cd sentiment-analysis-web-app
```