Applied Artificial Intelligence

Argentina: Web Scraping and Sentiment Analysis with Streamlit App

Submitted By: Venkatesh Mahindra: [70572200035]
Submitted To: Prof. Rajesh Prabhakar

SVKM'S NMIMS HYDERABAD

Project Overview

1. Introduction to Sentiment Analysis

Sentiment Analysis, also known as opinion mining, is a subfield of Natural Language Processing (NLP) that focuses on identifying and categorizing opinions or sentiments expressed in textual data. The goal is to determine whether the expressed sentiment is **positive**, **negative**, or **neutral**. It has widespread applications in areas like customer feedback analysis, product review mining, social media monitoring, and political trend analysis.

In this project, we aim to perform sentiment analysis on textual content scraped from a Wikipedia article of a selected country (*Argentina*, in this case). The idea is to understand the emotional polarity embedded within encyclopedic content, convert it into machine-readable features, apply classification techniques, and deploy it via a simple web interface.

---

2. Data Acquisition through Web Scraping

Web scraping is the process of extracting data from websites. We utilized the requests library to make HTTP requests and fetch the HTML content of the Wikipedia article. Using BeautifulSoup, the relevant text was parsed and cleaned by targeting <p> tags, which typically contain the main body of the content.

This resulted in a raw text corpus containing a mixture of informative historical, political, and geographical content from the Wikipedia page.

---

3. Text Preprocessing

Text preprocessing is a crucial step in any NLP pipeline as raw text data is often noisy and inconsistent. Several preprocessing techniques were applied:

- **Removing citations and references**: e.g., [1], [2], etc.
- **Removing special characters and digits**: Ensured only alphabetic text was retained.
- **Converting text to lowercase**: Ensured uniformity for processing.
- **Tokenization**: Using NLTK's sent_tokenize() and word_tokenize() to break down the text into sentences and words respectively.
- **Stopword Removal**: Common words that don't contribute to sentiment (e.g., "the", "is", "and") were removed using NLTK's stopword corpus.

---

4. Sentiment Analysis using TextBlob

TextBlob is a simple NLP library that provides sentiment polarity scores for a given text:

- Polarity ranges from -1 (very negative) to +1 (very positive).
- Based on the polarity score, each sentence was categorized as:
    - **Positive** if score > 0
    - **Negative** if score < 0
    - **Neutral** if score = 0

Only positive and negative sentences were retained for training purposes to keep the classification binary (2-class).

---

5. Feature Extraction using TF-IDF

To make textual data usable for machine learning algorithms, we used **TF-IDF Vectorization**:

- **Term Frequency (TF)**: Measures how frequently a word occurs in a document.
- **Inverse Document Frequency (IDF)**: Measures how important a word is, based on how frequently it appears across documents.
- The result is a sparse matrix where each sentence is represented by numerical features based on the importance of words.

This numerical representation served as the input to our ML models.

---

6. Handling Class Imbalance with SMOTE

Often in real-world datasets, one class (e.g., positive) is more frequent than another (e.g., negative), leading to biased models. To counter this, we used **SMOTE (Synthetic Minority Over-sampling Technique)**:

- SMOTE generates synthetic samples of the minority class.
- Helps balance the dataset, allowing classifiers to learn both classes equally.

---

7. Machine Learning Model Training

Multiple classification models were trained and evaluated:

🔹 **Logistic Regression**

A linear model for binary classification. It works well with high-dimensional sparse data like TF-IDF matrices.

🔹 **Decision Tree**

A tree-based model that splits data based on feature importance. Offers interpretability and handles both linear and non-linear data.

🔹 **Random Forest**

An ensemble of decision trees. Reduces overfitting and improves generalization.

🔹 **Gradient Boosting**

Another ensemble technique that builds models sequentially and corrects previous errors. Known for high performance on structured data.

🔹 **Naïve Bayes**

Based on Bayes' Theorem and assumes feature independence. It is fast and works well with text data.

🔹 **K-Nearest Neighbors (KNN)**

A distance-based classifier that assigns labels based on the majority label of the nearest neighbors.

---

8. Model Evaluation

Each model was evaluated using:

- **Accuracy**: Proportion of correct predictions.
- **Precision**: Correctness of positive predictions.
- **Recall**: Model's ability to detect positive instances.
- **F1-Score**: Harmonic mean of precision and recall.

The classification report helped compare performance across models and finalize the best-performing one.

---

9. Visualization

To gain insights from the data, the following visualizations were created:

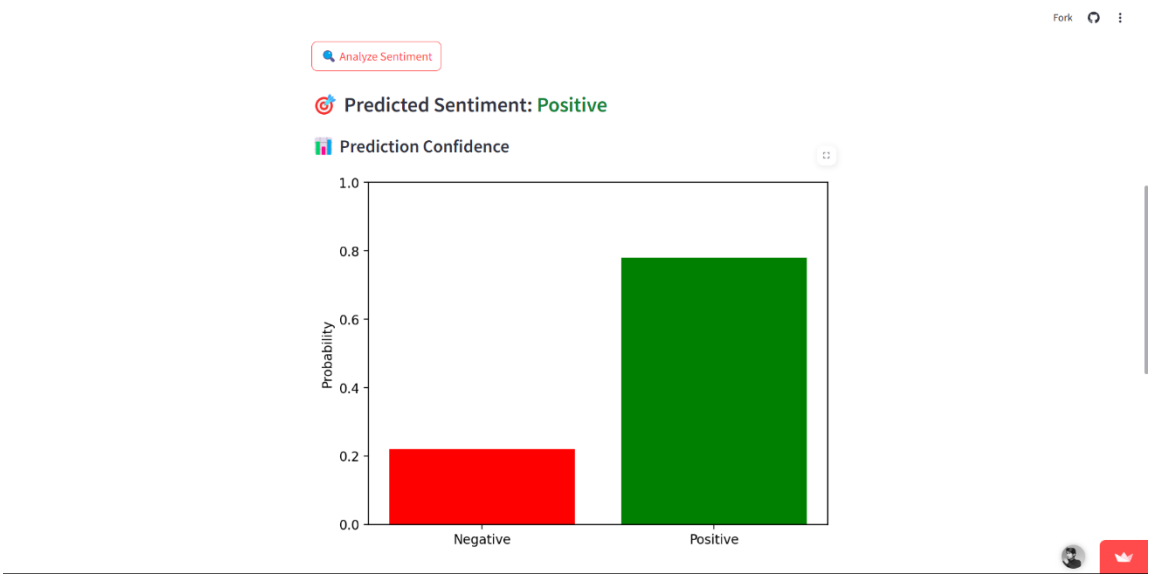
🔸 **WordCloud**

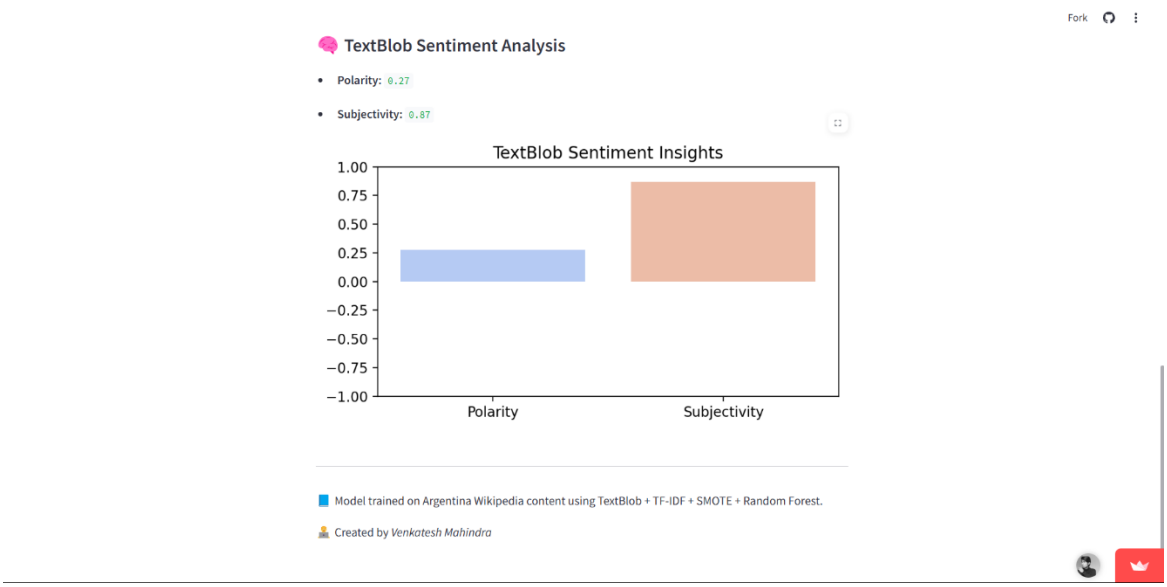
- A visual representation of the most frequent words in the dataset.
- Larger words indicate higher frequency.

🔸 **Bar Chart of Top Words**

- Displayed the 15 most common words post stopword removal.

- Helped identify themes or recurring topics in the content.





---

10. Model Saving and Deployment

After selecting the best-performing model (e.g., Random Forest), it was saved using **joblib** for deployment. The following components were also stored:

- Trained classifier model (random_forest_sentiment_model.pkl)
- TF-IDF vectorizer object (tfidf_vectorizer.pkl)

The application was built using **Streamlit**, a Python library for building interactive web apps. The app allows users to:

- Input a sentence
- Get real-time sentiment prediction
- Visualize WordCloud and frequent words

The app was deployed using **LocalTunnel**, making it accessible via a public URL for demonstration and testing.

---

11. Conclusion

This project demonstrated a full pipeline for sentiment analysis — from web scraping and preprocessing to classification and deployment. It highlighted the power of combining NLP with machine learning to extract meaningful insights from unstructured text. The deployed application provides a simple and interactive way for users to analyze sentiment, visualize text, and explore NLP techniques.

12.Links of Project

https://argentinawikiwebscraping-p8vhbvpmwqwfdnsojtarrp.streamlit.app/

https://github.com/venkatesh-mahindra/ArgentinaWikiwebscraping