Sentiment Analysis Chatbot with Flask and SQLite

This project is a **sentiment analysis chatbot** built using **Flask**, **SQLite**, and machine learning techniques based on **Naive Bayes**. The chatbot processes user comments, determines their sentiment (positive, negative, or neutral), and responds accordingly. Additionally, the project includes features for managing a database of analyzed comments, allowing updates to sentiments and optional model retraining.

Key Features

1. **Sentiment Analysis**
   - The machine learning model uses **Multinomial Naive Bayes** and a **Count Vectorizer** to analyze text.
   - User comments are preprocessed (converted to lowercase, stopwords removed, and lemmatized) and classified into three categories: **positive**, **negative**, and **neutral**.
2. **SQLite Database Management**
   - Analyzed comments and their sentiment predictions are stored in a table called SentimentData.
   - Each record in the database includes:
     - A unique **ID**
     - The **comment** text
     - The **sentiment prediction**
     - An optional **user comment**
   - Sentiments for stored comments can be updated directly via the database.
3. **Model Retraining**
   - If necessary, the model can be retrained using existing data in the database to improve performance over time.
4. **REST API**
   - The project includes several API endpoints for interaction:
     - POST /chat: Analyzes a comment and returns a prediction along with a chatbot response.
     - GET /frasi: Retrieves all analyzed comments from the database.

        - POST /modifica_sentenzo: Updates the sentiment for a specific comment and optionally retrains the model.

Main Components

Dataset

- A dataset of Italian comments was created, categorized as **positive**, **negative**, and **neutral**.
- 1000 random samples were generated to simulate a wide range of data.

Text Preprocessing

- Text is preprocessed with:
    - Lowercasing.
    - Removal of Italian stopwords.
    - Lemmatization using **WordNetLemmatizer**.

Machine Learning Model

- The model was developed using **scikit-learn**.
- A **Count Vectorizer** was used to transform text into numerical format suitable for the model.

SQLite Database

- The SentimentData table was designed to store all analyzed data.
- Functions are included to:
    - Insert new records.
    - Retrieve all data.
    - Update the sentiment for specific comments.

Frontend

- A basic **HTML template** (index.html) is provided to interact with the chatbot. Users can input comments and receive responses directly from the interface.