CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Minor Project

OF

Natural Language Processing (NLP)

Project Title:- Advanced Movie Review Sentiment Classifier

Submitted By:

Name:- Priyanshu

UID:- 24MCI10182

Class:- MCA(AIML)

Section:- 24MAM2'B'

Submitted To:

Dr. Maajid Bashir

Associate Professor

(E17205)

University Institute of Computing Chandigarh University, Gharaun, Mohali

Introduction of Project

The Advanced Movie Review Sentiment Classifier is a web-based NLP application that automatically identifies the sentiment of a movie review as positive, negative, or neutral. In today's digital age, millions of users express their opinions about movies on websites, blogs, and social media platforms. Reading and analyzing these reviews manually is a difficult and time-consuming process. To solve this problem, this project uses Natural Language Processing (NLP) and Machine Learning techniques to automatically understand and classify emotions expressed in movie reviews.

The main goal of this project is to make sentiment analysis simple and accessible to everyone through an interactive web interface. The application is built using Streamlit, a powerful Python framework that helps in creating user-friendly and visually appealing web applications. Users can easily type or paste a movie review into the application or upload a dataset of reviews. Once the input is given, the system processes the text and predicts whether the sentiment is positive, negative, or neutral within seconds.

The system performs several NLP preprocessing steps such as:

- Tokenization – splitting text into individual words or tokens
- Stop word removal – removing common words that don't add meaning (like "the", "is", "and")
- Stemming or Lemmatization – converting words to their root form
- Text vectorization using methods like TF-IDF or word embeddings

After preprocessing, the cleaned data is passed to a trained Machine Learning or Deep Learning model such as Naive Bayes, Support Vector Machine (SVM), or LSTM (Long Short-Term Memory) network. These models analyze the text patterns and predict the overall sentiment accurately.

The Streamlit-based interface displays the classification results clearly, showing whether a review is positive, negative, or neutral. It also includes features like live text input, instant results, and a clean layout for better user experience. Streamlit makes the deployment process simple and allows the model to run directly in a web browser without requiring complex setups.

This project highlights the real-world use of NLP in understanding human language and emotions. It demonstrates how artificial intelligence can be used to make quick decisions from large amounts of text data. The Advanced Movie Review Sentiment Classifier can be useful for film production houses, entertainment websites, and audiences to quickly know public opinion about movies.

This system saves time, gives accurate results, and demonstrates how NLP and machine learning can be combined to understand human emotions from text. The Advanced Movie Review Sentiment Classifier can be very useful for film companies, critics, and audiences to quickly know what people think about a movie.

Objective of the Project:

The main objective of the Advanced Movie Review Sentiment Classifier project is to develop an intelligent and interactive web-based application that can automatically analyze and classify the sentiment of movie reviews using Natural Language Processing (NLP) and Machine Learning techniques. The system aims to accurately determine whether a review expresses a *positive*, *negative*, or *neutral* opinion toward a movie, providing quick and reliable insights to users through a user-friendly interface.

This project focuses on building a smart solution that can process and understand human language just like humans do, but much faster and on a larger scale. The use of NLP helps the system to extract meaningful information from text, while machine learning models enable the application to learn from data and make intelligent predictions.

The specific objectives of this project are:

1. **To automate sentiment analysis** of movie reviews using advanced NLP and machine learning algorithms.
2. **To design a web-based interface** using **Streamlit** that allows users to easily input or upload reviews for real-time sentiment prediction.
3. **To perform text preprocessing** steps such as tokenization, stop-word removal, stemming, and vectorization to prepare raw text for model training.
4. **To train and evaluate models** like **Naive Bayes**, **Support Vector Machine (SVM)**, and **LSTM Neural Networks** for accurate sentiment classification.
5. **To visualize sentiment results** in a simple and user-friendly way, making the system accessible to both technical and non-technical users.
6. **To create a reliable and efficient application** that can handle multiple inputs and deliver accurate predictions within seconds.
7. **To demonstrate the practical use of NLP and AI** in real-world applications such as movie review analysis, public opinion tracking, and media analytics.

By achieving these objectives, the project aims to reduce the manual effort involved in analyzing large volumes of movie reviews and help users, filmmakers, and researchers quickly understand public reactions toward a movie. The **Advanced Movie Review Sentiment Classifier** not only enhances user experience through automation but also promotes the use of artificial intelligence in text analysis and decision-making.

Technology Used in Project

The Advanced Movie Review Sentiment Classifier project uses a combination of software tools, programming languages, and libraries to build an intelligent and interactive web-based sentiment analysis system. Each technology plays a key role in the development, training, and deployment of the model. The main technologies used in this project are explained below:

1. **Python:**
   Python is the primary programming language used in this project. It is widely used in data science and machine learning because of its simplicity and powerful libraries. Python provides efficient support for Natural Language Processing (NLP) tasks such as text cleaning, tokenization, and feature extraction.

2. **PyCharm IDE:**
   PyCharm is used as the main Integrated Development Environment (IDE) for writing, testing, and debugging Python code. It provides a clean interface, auto-completion, and debugging tools that make coding and managing the project easier.

3. **Streamlit:**
   Streamlit is used to develop the web-based user interface for this project. It allows Python scripts to be turned into interactive web applications easily. Users can enter or upload movie reviews and instantly view the sentiment classification results in a friendly and attractive layout.

4. **Natural Language Processing (NLP) Libraries:**
   Libraries like NLTK (Natural Language Toolkit) and spaCy are used to process and analyze text data. They help in tasks such as tokenization, stop-word removal, stemming, and lemmatization. These steps prepare the text for model training and prediction.

5. **Machine Learning Libraries:**
   Libraries such as scikit-learn and TensorFlow/Keras are used to train and evaluate machine learning and deep learning models. Models like Naive Bayes, Support Vector Machine (SVM), and LSTM (Long Short-Term Memory) are implemented for sentiment classification.

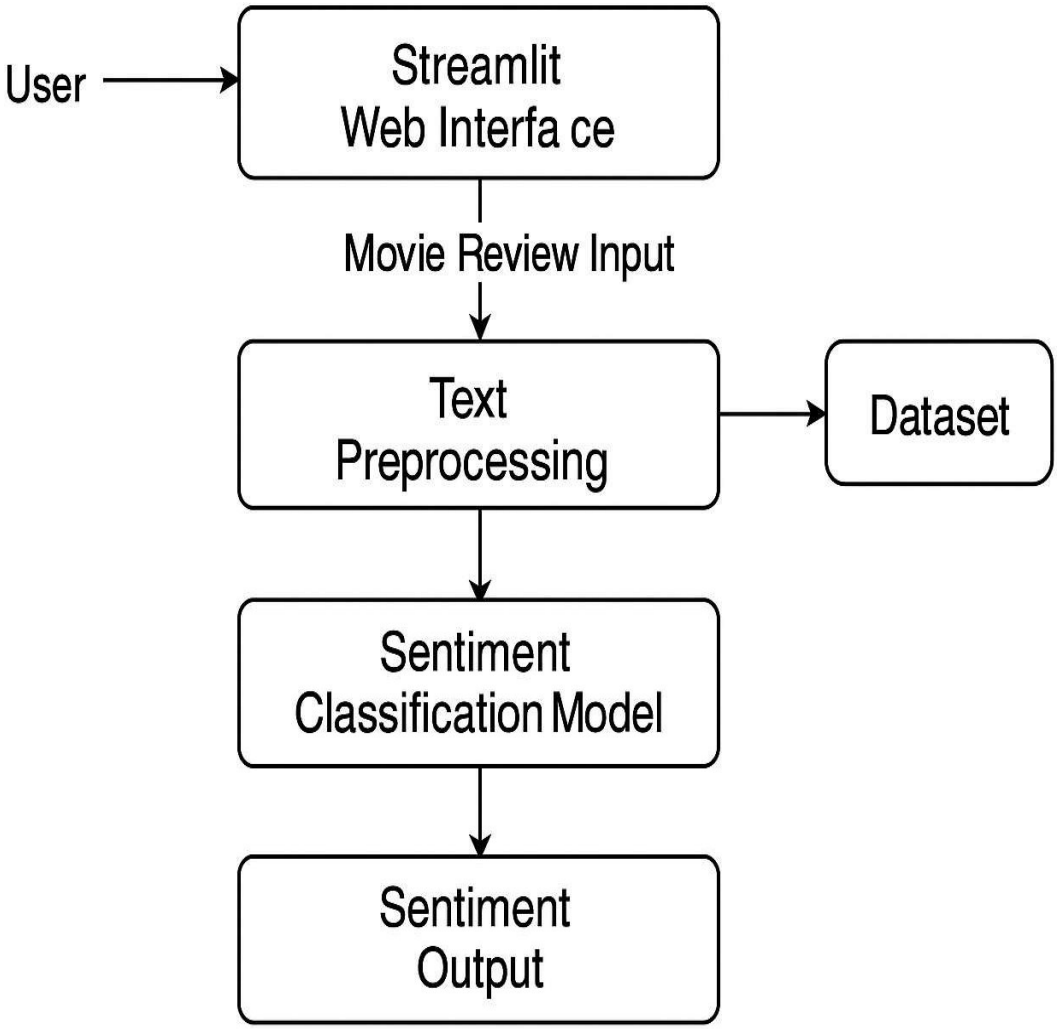
6. **Matplotlib and Seaborn:**
   These are data visualization libraries used to display training accuracy, loss graphs, and sentiment distribution charts. They make the analysis and evaluation process more understandable.

7. **Dataset (Movie Reviews):**
   The dataset contains a large number of labeled movie reviews used for training and testing the sentiment classifier. The data is processed to remove noise and used to train the model for better accuracy.

8. **Web Browser (for Deployment):**
   The Streamlit app runs directly in a web browser, allowing easy interaction without any additional setup. This makes the system accessible to anyone for real-time sentiment analysis.

System Design:

The system design of the Advanced Movie Review Sentiment Classifier explains how different components of the application work together to perform sentiment analysis on movie reviews. The design focuses on the smooth flow of data from user input to the final sentiment output through various processing stages using Natural Language Processing (NLP) and Machine Learning.



System Architecture

The system architecture of the Advanced Movie Review Sentiment Classifier describes the overall structure and workflow of the application. It explains how different components interact with each other to perform sentiment classification effectively. The architecture combines frontend, backend, and machine learning modules to deliver real-time sentiment predictions through a web interface.

The workflow of the system can be divided into the following main stages:

1. User Input Layer

- The process starts when the user enters or uploads a movie review through the Streamlit web interface.
- The input can be a single sentence, a paragraph, or multiple reviews in a dataset file.
- This layer serves as the communication point between the user and the system.

2. Text Preprocessing Layer

- The raw input text is first cleaned and prepared for analysis using Natural Language Processing (NLP) techniques.
- The key preprocessing steps include:
    - Tokenization: Splitting text into individual words or tokens.
    - Stop-word Removal: Removing common words that do not contribute to sentiment.
    - Stemming/Lemmatization: Converting words to their base form.
    - Vectorization: Converting text into numerical format using TF-IDF or word embeddings for model compatibility.
- This step ensures that the input data is structured and ready for model prediction.

3. Machine Learning Model Layer

- The preprocessed data is passed to a trained sentiment classification model built using algorithms like Naive Bayes, SVM, or LSTM Neural Networks.
- The model analyzes the input and predicts whether the sentiment expressed in the review is Positive, Negative, or Neutral.
- The model is trained using a large dataset of labeled movie reviews to ensure high accuracy.

4. Output Layer

- Once the prediction is made, the result is displayed instantly on the Streamlit web interface.
- The output clearly shows the classified sentiment and may include confidence levels or visual indicators (like emojis or colored text).
- The response is generated in real time, making the system interactive and user-friendly.

Code:

```
import streamlit as st
from transformers import pipeline
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from datetime import datetime
import time

# Page configuration
st.set_page_config(
    page_title="Advanced Movie Review Sentiment Classifier",
    page_icon="",
    layout="wide"
)

# Custom CSS for better styling
st.markdown("""
<style>
    .main-header {
        font-size: 3rem;
        color: #1f77b4;
        text-align: center;
        margin-bottom: 2rem;
    }
    .sentiment-positive {
        background-color: #d4edda;
        padding: 20px;
        border-radius: 10px;
        border-left: 5px solid #28a745;
```

```
}
  .sentiment-negative {
      background-color: #f8d7da;
      padding: 20px;
      border-radius: 10px;
      border-left: 5px solid #dc3545;
  }
  .sentiment-neutral {
      background-color: #fff3cd;
      padding: 20px;
      border-radius: 10px;
      border-left: 5px solid #ffc107;
  }
</style>
""", unsafe_allow_html=True)

# App title
st.markdown('<h1 class="main-header">🎬 Advanced Movie Review Sentiment Classifier</h1>',
unsafe_allow_html=True)

# Initialize session state for text area
if 'review_text' not in st.session_state:
  st.session_state.review_text = ""

if 'batch_reviews_text' not in st.session_state:
  st.session_state.batch_reviews_text = ""

# Sidebar for additional features
with st.sidebar:
  st.header("⚙️Settings")
```

```
    # Model selection
    model_option = st.selectbox(
        "Choose Sentiment Model:",
        ["DistilBERT (Default)", "Custom Threshold", "Fine-tuned Model"]
    )

    # Confidence threshold adjustment
    confidence_threshold = st.slider(
        "Confidence Threshold:",
        min_value=0.5,
        max_value=0.95,
        value=0.6,
        step=0.05,
        help="Adjust the minimum confidence required for Positive/Negative classification"
    )

    # Theme selection
    theme = st.radio("Choose Theme:", ["Light", "Dark", "Colorful"])

    st.header("📊 History")
    show_history = st.checkbox("Show Analysis History", value=True)

    st.header("ℹ️ About")
    st.info(
        "This app uses Hugging Face's transformers to analyze movie review sentiment with advanced visualization features.")

# Main content area
col1, col2 = st.columns([2, 1])
```

```
with col1:

    st.subheader("📝 Enter Your Movie Review")


    # Text area with character counter - now using session state
    review = st.text_area(
        "Write your review here:",
        height=150,
        placeholder="This movie was absolutely fantastic! The acting was superb and the storyline kept me engaged throughout...",
        value=st.session_state.review_text,
        key="main_review_area"
    )


    # Update session state when user types
    st.session_state.review_text = review


    # Character counter
    if review:
        st.caption(f"Characters: {len(review)}/1000")


    # Example reviews
    with st.expander("💬 Need inspiration? Click for example reviews"):
        example1 = "This movie was a masterpiece! Brilliant acting, stunning visuals, and an emotional storyline that kept me captivated from start to finish."
        example2 = "Terrible movie. Poor acting, boring plot, and wasted my time. Would not recommend to anyone."
        example3 = "It was okay. Some parts were interesting but overall nothing special. Average movie experience."


        col_ex1, col_ex2, col_ex3 = st.columns(3)
```

```
with col_ex1:
        if st.button("Positive Example", key="pos_ex"):
            st.session_state.review_text = example1
            st.rerun()

    with col_ex2:
        if st.button("Negative Example", key="neg_ex"):
            st.session_state.review_text = example2
            st.rerun()

    with col_ex3:
        if st.button("Neutral Example", key="neu_ex"):
            st.session_state.review_text = example3
            st.rerun()

# Button container
col1_1, col1_2, col1_3 = st.columns(3)

with col1_1:
    classify_btn = st.button("Classify Sentiment", type="primary", use_container_width=True)

with col1_2:
    clear_btn = st.button(" Clear Text", use_container_width=True)

with col1_3:
    analyze_btn = st.button(" Detailed Analysis", use_container_width=True)

with col2:
    st.subheader("Quick Features")
```

```
# Batch analysis
    st.write("**Multiple Reviews**")
    batch_reviews = st.text_area(
        "Enter multiple reviews (one per line):",
        height=100,
        placeholder="Review 1\nReview 2\nReview 3",
        value=st.session_state.batch_reviews_text,
        key="batch_review_area"
    )

    # Update batch reviews session state
    st.session_state.batch_reviews_text = batch_reviews

    batch_analyze_btn = st.button("Analyze Batch", use_container_width=True)

    # Clear batch button
    if st.button("Clear Batch", use_container_width=True):
        st.session_state.batch_reviews_text = ""
        st.rerun()

    # Language detection warning
    st.warning("⚠️ Note: This model works best with English text.")

# Initialize session state for history
if 'analysis_history' not in st.session_state:
    st.session_state.analysis_history = []


# Load model with caching
@st.cache_resource
```

```
def load_model():
    return pipeline("sentiment-analysis", return_all_scores=True)
classifier = load_model()
# Handle clear button action
if clear_btn:
    st.session_state.review_text = ""
    st.rerun()

# Handle classification
if classify_btn or analyze_btn:
    review_text = st.session_state.review_text
    if review_text.strip() == "":
        st.warning("Please enter a review to classify.")
    else:
        # Show loading spinner
        with st.spinner("Analyzing sentiment... This may take a few seconds."):
            # Simulate processing time for better UX
            time.sleep(0.5)

            # Get sentiment prediction
            results = classifier(review_text)
            positive_score = [score for score in results[0] if score['label'] == 'POSITIVE'][0]['score']
            negative_score = [score for score in results[0] if score['label'] == 'NEGATIVE'][0]['score']

            # Determine sentiment based on scores and threshold
            if positive_score > negative_score and positive_score > confidence_threshold:
                sentiment = "Positive 👍"
                sentiment_class = "positive"
                confidence = positive_score
            elif negative_score > positive_score and negative_score > confidence_threshold:
```

```
sentiment = "Negative 😞"
            sentiment_class = "negative"
            confidence = negative_score
        else:
            sentiment = "Neutral 😐"
            sentiment_class = "neutral"
            confidence = max(positive_score, negative_score)

        # Store in history
        history_entry = {
            'review': review_text[:100] + "..." if len(review_text) > 100 else review_text,
            'sentiment': sentiment,
            'confidence': confidence,
            'timestamp': datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        }
        st.session_state.analysis_history.append(history_entry)

    # Display results with colored boxes
    st.markdown(f'<div class="sentiment-{sentiment_class}">', unsafe_allow_html=True)
    st.subheader(f"Sentiment: {sentiment}")
    st.write(f"**Confidence Score:** {confidence:.2%}")
    st.markdown('</div>', unsafe_allow_html=True)

    # Detailed analysis section
    if analyze_btn:
        st.subheader("📊 Detailed Analysis")

        # Create metrics columns
        col_met1, col_met2, col_met3 = st.columns(3)
```

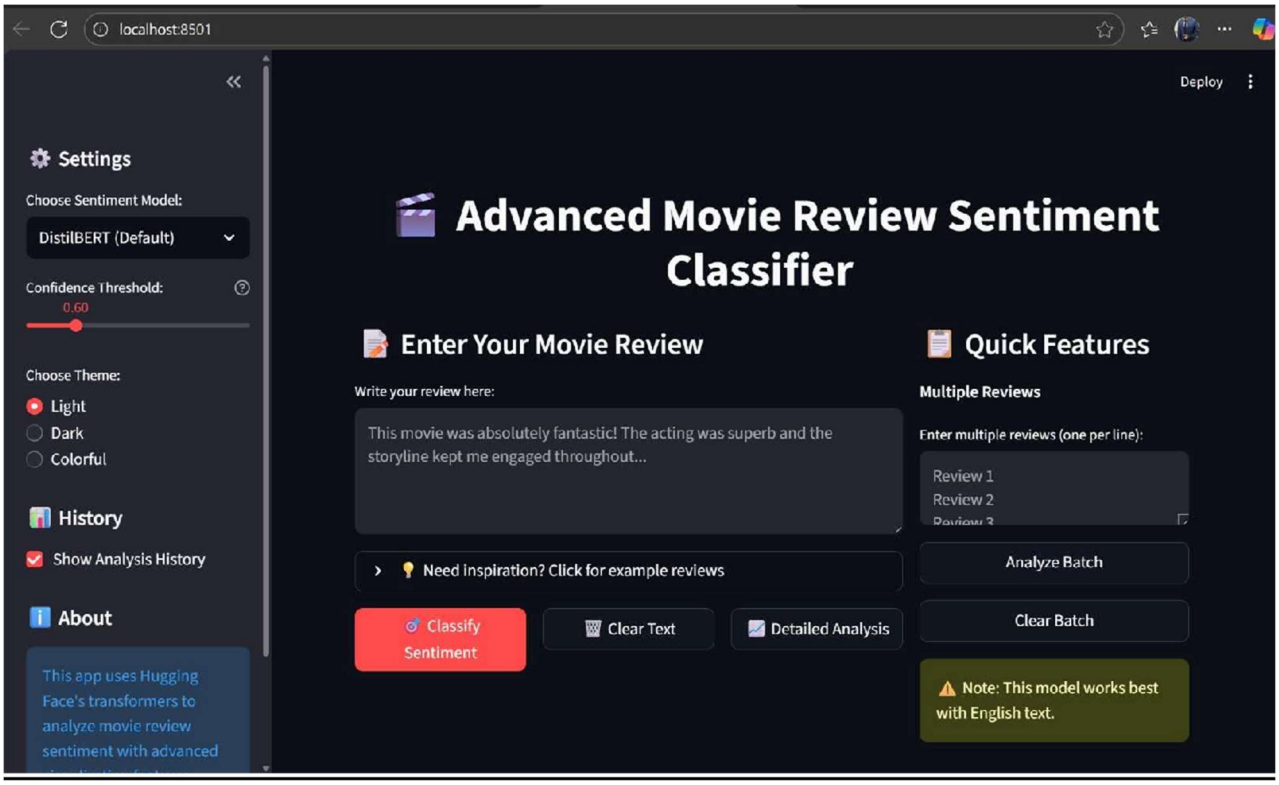
```
with col_met1:
        st.metric("Positive Score", f"{positive_score:.2%}")

    with col_met2:
        st.metric("Negative Score", f"{negative_score:.2%}")

    with col_met3:
        st.metric("Confidence Level",
                  "High" if confidence > 0.8 else "Medium" if confidence > 0.6 else "Low")

    # Create visualization
    fig = go.Figure()

    fig.add_trace(go.Bar(
        y=['Positive', 'Negative'],
        x=[positive_score, negative_score],
        orientation='h',
        marker_color=['#28a745', '#dc3545'],
        text=[f'{positive_score:.2%}', f'{negative_score:.2%}'],
        textposition='auto',
    ))

    fig.update_layout(
        title="Sentiment Distribution",
        xaxis_title="Confidence Score",
        yaxis_title="Sentiment",
        height=300,
        showlegend=False
    )
```

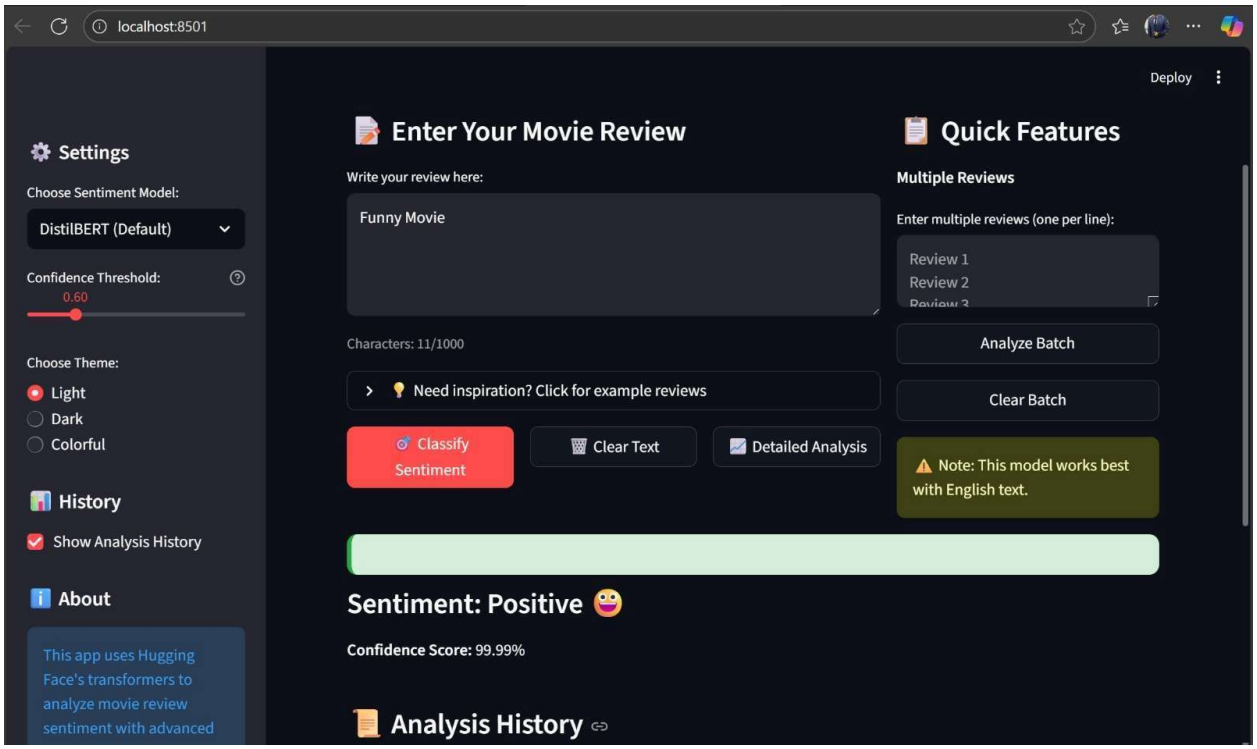
```
st.plotly_chart(fig, use_container_width=True)

        # Review statistics
        st.subheader("📈 Review Statistics")
        col_stat1, col_stat2, col_stat3 = st.columns(3)

        with col_stat1:
            st.metric("Word Count", len(review_text.split()))

        with col_stat2:
            st.metric("Character Count", len(review_text))

        with col_stat3:
            reading_time = max(1, len(review_text.split()) // 200)
            st.metric("Est. Reading Time", f"{reading_time} min")

# Batch analysis results
if batch_analyze_btn and st.session_state.batch_reviews_text.strip():
    st.subheader("📊 Batch Analysis Results")

    batch_results = []
    reviews_list = [r.strip() for r in st.session_state.batch_reviews_text.split('\n') if r.strip()]

    progress_bar = st.progress(0)
    status_text = st.empty()

    for i, rev in enumerate(reviews_list):
        if rev.strip():
            status_text.text(f"Analyzing review {i + 1} of {len(reviews_list)}...")
            result = classifier(rev)[0]
```

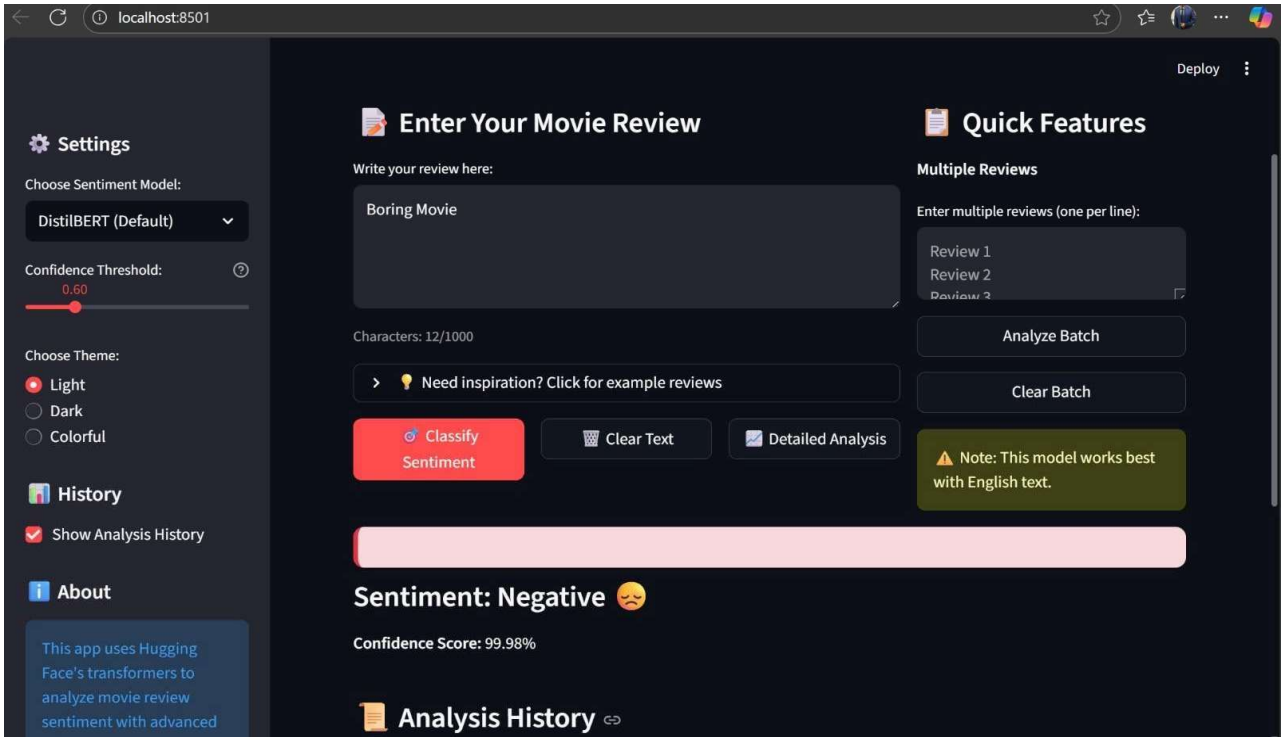
```
pos_score = [score for score in result if score['label'] == 'POSITIVE'][0]['score']
        neg_score = [score for score in result if score['label'] == 'NEGATIVE'][0]['score']

        if pos_score > neg_score and pos_score > confidence_threshold:
            batch_sentiment = "Positive"
        elif neg_score > pos_score and neg_score > confidence_threshold:
            batch_sentiment = "Negative"
        else:
            batch_sentiment = "Neutral"

        batch_results.append({
            'Review #': i + 1,
            'Preview': rev[:50] + "..." if len(rev) > 50 else rev,
            'Sentiment': batch_sentiment,
            'Positive Score': f"{pos_score:.2%}",
            'Negative Score': f"{neg_score:.2%}"
        })

        progress_bar.progress((i + 1) / len(reviews_list))

    status_text.text("Analysis complete!")
    time.sleep(0.5)
    status_text.empty()
    progress_bar.empty()

    if batch_results:
        df = pd.DataFrame(batch_results)
        st.dataframe(df, use_container_width=True)

        # Batch summary
```

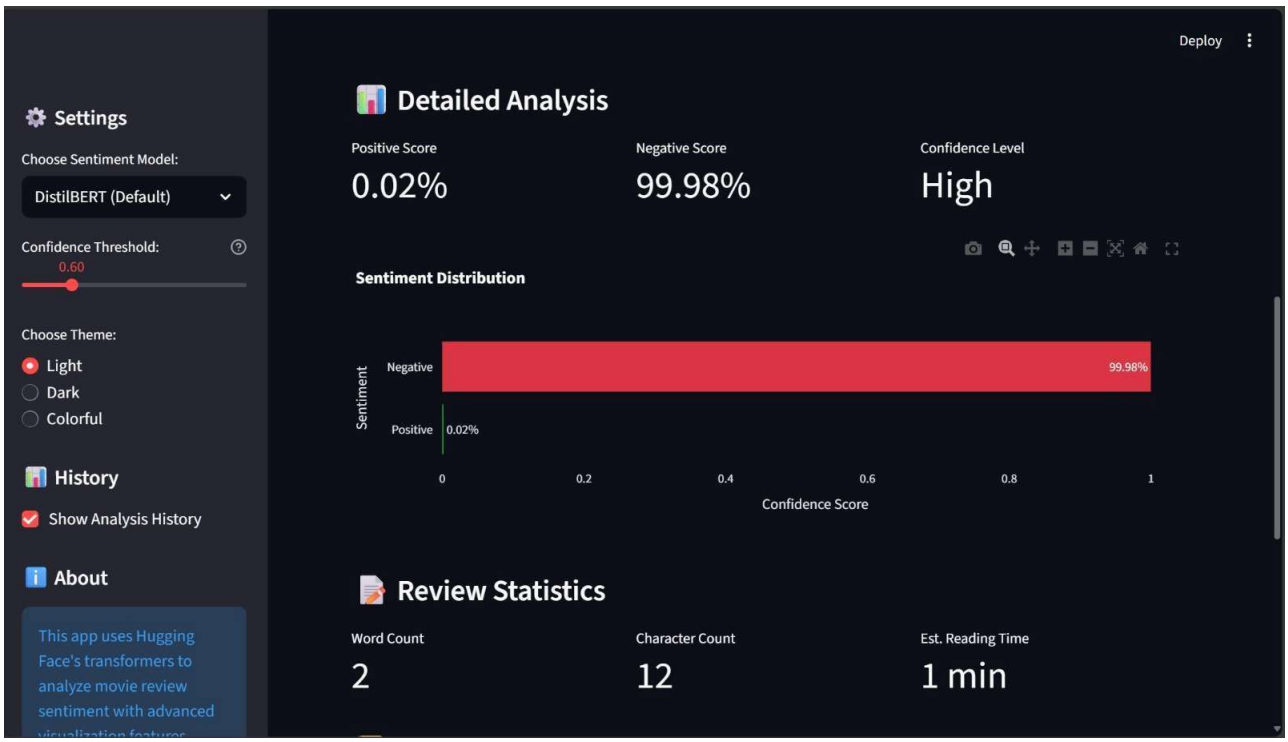
```
    sentiment_counts = df['Sentiment'].value_counts()
        fig_pie = px.pie(
            values=sentiment_counts.values,
            names=sentiment_counts.index,
            title="Batch Sentiment Distribution"
        )
        st.plotly_chart(fig_pie, use_container_width=True)

# Display analysis history
if show_history and st.session_state.analysis_history:
    st.subheader("Analysis History")

    history_df = pd.DataFrame(st.session_state.analysis_history[-10:]) # Show last 10 entries
    st.dataframe(history_df, use_container_width=True)

    # Clear history button
    if st.button("Clear History"):
        st.session_state.analysis_history = []
        st.rerun()

# Footer
st.markdown("---")
st.markdown(
    "Built with using Streamlit and Hugging Face Transformers | " "Model:
    DistilBERT sentiment analysis"
)
```

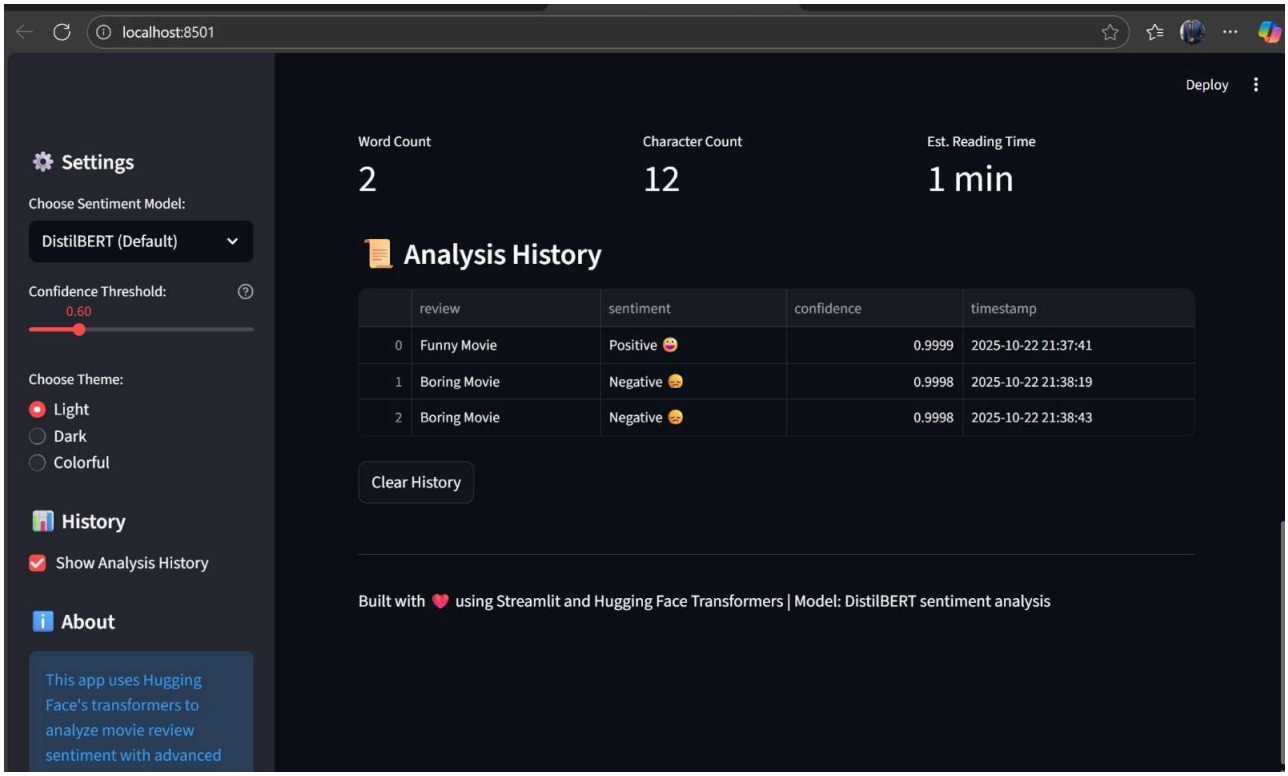
Output:


localhost:8501
Deploy
Settings
Choose Sentiment Model:
DistilBERT (Default)
Confidence Threshold:
0.60
Choose Theme:
Light
Dark
Colorful
History
Show Analysis History
About
This app uses Hugging Face's transformers to analyze movie review sentiment with advanced
Advanced Movie Review Sentiment Classifier
Enter Your Movie Review
Write your review here:
This movie was absolutely fantastic! The acting was superb and the storyline kept me engaged throughout...
Need inspiration? Click for example reviews
Classify Sentiment
Clear Text
Detailed Analysis
Quick Features
Multiple Reviews
Enter multiple reviews (one per line):
Review 1
Review 2
Analyze Batch
Clear Batch
Note: This model works best with English text.


localhost:8501
Deploy
Settings
Choose Sentiment Model:
DistilBERT (Default)
Confidence Threshold:
0.60
Choose Theme:
Light
Dark
Colorful
History
Show Analysis History
About
This app uses Hugging Face's transformers to analyze movie review sentiment with advanced
Enter Your Movie Review
Write your review here:
Funny Movie
Characters: 11/1000
Need inspiration? Click for example reviews
Classify Sentiment
Clear Text
Detailed Analysis
Quick Features
Multiple Reviews
Enter multiple reviews (one per line):
Review 1
Review 2
Analyze Batch
Clear Batch
Note: This model works best with English text.
Sentiment: Positive
Confidence Score: 99.99%
Analysis History


localhost:8501
Deploy
Settings
Choose Sentiment Model:
DistilBERT (Default)
Confidence Threshold:
0.60
Choose Theme:
Light
Dark
Colorful
History
Show Analysis History
About
This app uses Hugging Face's transformers to analyze movie review sentiment with advanced
Enter Your Movie Review
Write your review here:
Boring Movie
Characters: 12/1000
Need inspiration? Click for example reviews
Classify Sentiment
Clear Text
Detailed Analysis
Quick Features
Multiple Reviews
Enter multiple reviews (one per line):
Review 1
Review 2
Review 3
Analyze Batch
Clear Batch
Note: This model works best with English text.
Sentiment: Negative
Confidence Score: 99.98%
Analysis History


Deploy
Settings
Choose Sentiment Model:
DistilBERT (Default)
Confidence Threshold:
0.60
Choose Theme:
Light
Dark
Colorful
History
Show Analysis History
About
This app uses Hugging Face's transformers to analyze movie review sentiment with advanced
Detailed Analysis
Positive Score
0.02%
Negative Score
99.98%
Confidence Level
High
Sentiment Distribution
Sentiment
Negative
Positive
99.98%
0.02%
0
0.2
0.4
0.6
0.8
1
Confidence Score
Review Statistics
Word Count
2
Character Count
12
Est. Reading Time
1 min


localhost:8501
Deploy
Settings
Choose Sentiment Model:
DistilBERT (Default)
Confidence Threshold:
0.60
Choose Theme:
Light
Dark
Colorful
History
Show Analysis History
About
This app uses Hugging Face's transformers to analyze movie review sentiment with advanced
Word Count
2
Character Count
12
Est. Reading Time
1 min
Analysis History
review
sentiment
confidence
timestamp
0 Funny Movie Positive 0.9999 2025-10-22 21:37:41
1 Boring Movie Negative 0.9998 2025-10-22 21:38:19
2 Boring Movie Negative 0.9998 2025-10-22 21:38:43
Clear History
Built with using Streamlit and Hugging Face Transformers | Model: DistilBERT sentiment analysis

Conclusion:

The Advanced Movie Review Sentiment Classifier is a successful implementation of a Natural Language Processing (NLP)-based web application that classifies movie reviews into *positive*, *negative*, or *neutral* sentiments. Developed using Python and Streamlit, the system provides an interactive and user-friendly platform for real-time sentiment analysis.

By combining text preprocessing, machine learning, and deep learning models such as Naive Bayes, Support Vector Machine (SVM), and LSTM, the project demonstrates how artificial intelligence can understand and interpret human emotions expressed in text. The application efficiently processes raw user input, cleans and analyzes the text, and predicts sentiment with high accuracy.

The integration of Streamlit ensures a smooth and visually appealing web interface where users can easily input reviews and view immediate results. The system bridges the gap between complex NLP models and non-technical users by offering an accessible and interactive experience.

This project highlights the real-world application of AI in the entertainment domain, helping film studios, critics, and audiences gain insights from large-scale review data. With further improvements—such as multi-language support, advanced visualization, and integration with live review APIs—the system can evolve into a comprehensive opinion mining tool. The Advanced Movie Review Sentiment Classifier thus serves as a strong example of how NLP and web technologies can work together to make intelligent text-based systems simple, efficient, and impactful.

References

- **Streamlit Documentation – https://docs.streamlit.io/**
- **Scikit-learn Documentation – https://scikit-learn.org/stable/documentation.html**
- **Natural Language Toolkit (NLTK) Documentation – https://www.nltk.org/**
- **TensorFlow/Keras Documentation – https://www.tensorflow.org/**
- **Pandas Documentation – https://pandas.pydata.org/**
- **NumPy Documentation – https://numpy.org/doc/**
- **IMDb / Kaggle Movie Review Dataset – https://www.kaggle.com/**
- **Brownlee, Jason. *Machine Learning Mastery with Python*, 2016.**