Sentiment Analysis Of Movie Reviews

A PROJECT REPORT

by

Simran Tyagi (20241011610020 8)
Shefali Yadav (20241011610019 5)
Tanisha Jain (20241011610021 7)
Vishal Chaturvedi (20241011610024 8)

Session:2024-2025 (II Semester)

Under the supervision of

Ms. Komal Salgotra

KIET Group of Institutions, Delhi-NCR, Ghaziabad

DEPARTMENT OF COMPUTER APPLICATIONS
KIET GROUP OF INSTITUTIONS, DELHI-NCR, GHAZIABAD-201206
(2024- 2025)

1. Introduction

In today's digital age, the internet is flooded with user-generated content in the form of reviews, social media posts, and comments. Analyzing this vast amount of textual data manually is nearly impossible, which brings us to the importance of automated sentiment analysis. Sentiment Analysis, also known as opinion mining, is a technique used in Natural Language Processing (NLP) that determines whether a given piece of text expresses a positive, negative, or neutral sentiment. It plays a vital role in understanding public opinion, brand monitoring, market research, and customer service.

This project focuses specifically on performing sentiment analysis on movie reviews using Python programming language within the Google Colab environment. The goal is to build a machine learning model capable of classifying reviews into positive or negative sentiments based on their content. The sentiment analysis process involves several key stages: data loading, text preprocessing, feature extraction, model building, evaluation, and prediction. Each stage contributes significantly to the accuracy and effectiveness of the final model.

In addition to training a classifier, the project includes a functionality to predict the sentiment of custom user-input reviews. For instance, when a user inputs a sentence such as "The movie was outstanding with brilliant performances," the system will output a positive sentiment.We not only achieve a robust sentiment classification model but also gain deep insights into the language patterns used in movie reviews. This foundational knowledge can be extended to other domains such as product reviews, customer feedback, and social media analysis. Overall, this project serves as a comprehensive introduction to the application of machine learning in NLP and provides a valuable learning experience in building real-world sentiment analysis systems.

1.2 Objective

The main goal of this project is to develop an end-to-end sentiment analysis system that:

- Loads and preprocesses raw movie review data
- Extracts meaningful features from text
- Trains a classification model to identify sentiment
- Evaluates model performance using various metrics
- Visualizes data insights and model results
- Provides an interface for custom sentiment prediction

1.3 Scope

This project aims to perform sentiment analysis on movie reviews using Python and machine learning techniques. It covers:

- Cleaning and preprocessing text data from the IMDB movie review dataset
- Converting text into numerical features using TF-IDF vectorization
- Building and evaluating a Logistic Regression model for sentiment classification
- Predicting sentiment (positive or negative) of custom user input
- Visualizing data insights using WordClouds and confusion matrices

The project is limited to English-language, binary classification (positive/negative). It does not include deep learning models, multilingual analysis, or deployment as a web or mobile application—but these are possible future enhancements.

2. Methodology

The methodology followed in this project involves several systematic steps to build a reliable sentiment analysis system. These steps are as follows:

2.1 Data Collection

- Used the IMDB Large Movie Review Dataset containing 50,000 labeled movie reviews (25,000 for training and 25,000 for testing).
- The dataset is balanced with an equal number of positive and negative reviews.

2.2 Data Preprocessing

Preprocessing is crucial to prepare raw text for analysis. The following steps were applied:

- **Lowercasing**: All text converted to lowercase to ensure uniformity
- **HTML Tag Removal**: Removed using regular expressions to eliminate tags like
- **Punctuation Removal**: All special characters and punctuations stripped out
- **Number Removal**: Digits removed to focus on textual content
- **Stopword Removal**: Common English words (like "is", "the") removed using NLTK
- **Tokenization**: Text split into individual words for analysis

Each review is cleaned using a custom function to apply the above steps before feature extraction.

2.3 Feature Extraction

Since machine learning models cannot interpret raw text, it must be transformed into numerical features. We used **TF-IDF Vectorization**:

- **TF-IDF (Term Frequency-Inverse Document Frequency)** captures how important a word is in a document relative to the whole corpus
- Only the top 5000 most important features are retained
- Resulting vectorized data is used to train the model

2.4 Model Building

We used **Logistic Regression**, a commonly used and interpretable algorithm for binary classification tasks. Steps:

- Split the dataset into **training (80%)** and **testing (20%)**
- Fit the model on training data using the vectorized features
- Predict sentiments on unseen test data

Logistic Regression is suitable here due to its efficiency and effectiveness for text classification problems.

2.5 Model Evaluation

After model training, we evaluate performance using:

- **Accuracy Score**: The percentage of correctly predicted reviews
- **Classification Report**:
    - **Precision**: Accuracy of positive predictions
    - **Recall**: Coverage of actual positive instances
    - **F1-score**: Harmonic mean of precision and recall

2.6 Visualization

Visualization helps in understanding the data distribution and content:

- **Sentiment Distribution**:
    - A bar chart is generated using Seaborn to show the count of positive and negative reviews
    - This helps verify that the dataset is balanced
- **WordClouds**:
    - Created for both positive and negative reviews

- Highlights the most frequent and important words
- Helps interpret key vocabulary associated with each sentiment

2.7 User Interaction

- Built a function that takes custom user input (a movie review) and predicts the sentiment in real-time using the trained model.

3. Tools and Technologies Used

- **Programming Language**: Python
- **Platform**: Google Colab (cloud-based Jupyter Notebook)
- **Libraries and Packages**:
  - **Pandas, NumPy**: For data loading, manipulation, and numerical computations
  - **Matplotlib, Seaborn**: For generating plots and visualizations
  - **WordCloud**: To visualize frequently occurring words
  - **NLTK (Natural Language Toolkit)**: For text preprocessing (stopwords, tokenization)
  - **Scikit-learn**: For feature extraction, model building, evaluation

4. Results

- Achieved high accuracy (typically above 85%) on the test set
- Visualizations like WordClouds provided meaningful insights into frequent sentiment-related words
- Logistic Regression proved to be a strong baseline model for binary sentiment classification

5. Conclusion

This project successfully demonstrates the application of Natural Language Processing (NLP) and machine learning techniques to analyze sentiments in movie reviews. By using the IMDB dataset, we were able to build an end-to-end sentiment classification model that can automatically determine whether a given movie review expresses a positive or negative sentiment.

We followed a systematic pipeline that included data collection, preprocessing, feature extraction with TF-IDF, model training using Logistic Regression, and evaluation through standard metrics such as accuracy, precision, recall, and F1-score. The use of WordClouds and visual tools like confusion matrices enhanced the interpretability of the results and gave us better insight into the dataset.

The model achieved high accuracy, proving that even simple algorithms like Logistic Regression, when paired with effective preprocessing, can perform well on textual sentiment classification tasks. Additionally, the implementation of a custom prediction function made the project interactive and practical, simulating real-world usage.This project not only helped in building technical skills in Python, NLP, and machine learning but also highlighted the importance of data cleaning, feature selection, and proper model evaluation in achieving meaningful results.

6. Future Work

To further enhance this project, the following improvements are recommended:

- Incorporate **deep learning models** like LSTM, GRU, or pre-trained transformers (e.g., BERT)
- Add a **web-based user interface** for real-time sentiment predictions
- Extend classification to include **neutral and mixed sentiments**
- Analyze more complex datasets like tweets or product reviews

7. References

1. IMDB Movie Review Dataset
Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011).
https://ai.stanford.edu/~amaas/data/sentiment/

2. Scikit-learn: Machine Learning in Python
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011).
https://scikit-learn.org/stable/

3. NLTK – Natural Language Toolkit
Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python.
https://www.nltk.org/

4.Matplotlib: Visualization with Python
Hunter, J. D. (2007).
https://matplotlib.org/

5. WordCloud Python Library
Müller, A. (2015).
https://github.com/amueller/word_cloud

6. Google Colab
Google Research.
https://colab.research.google.com/

Code:

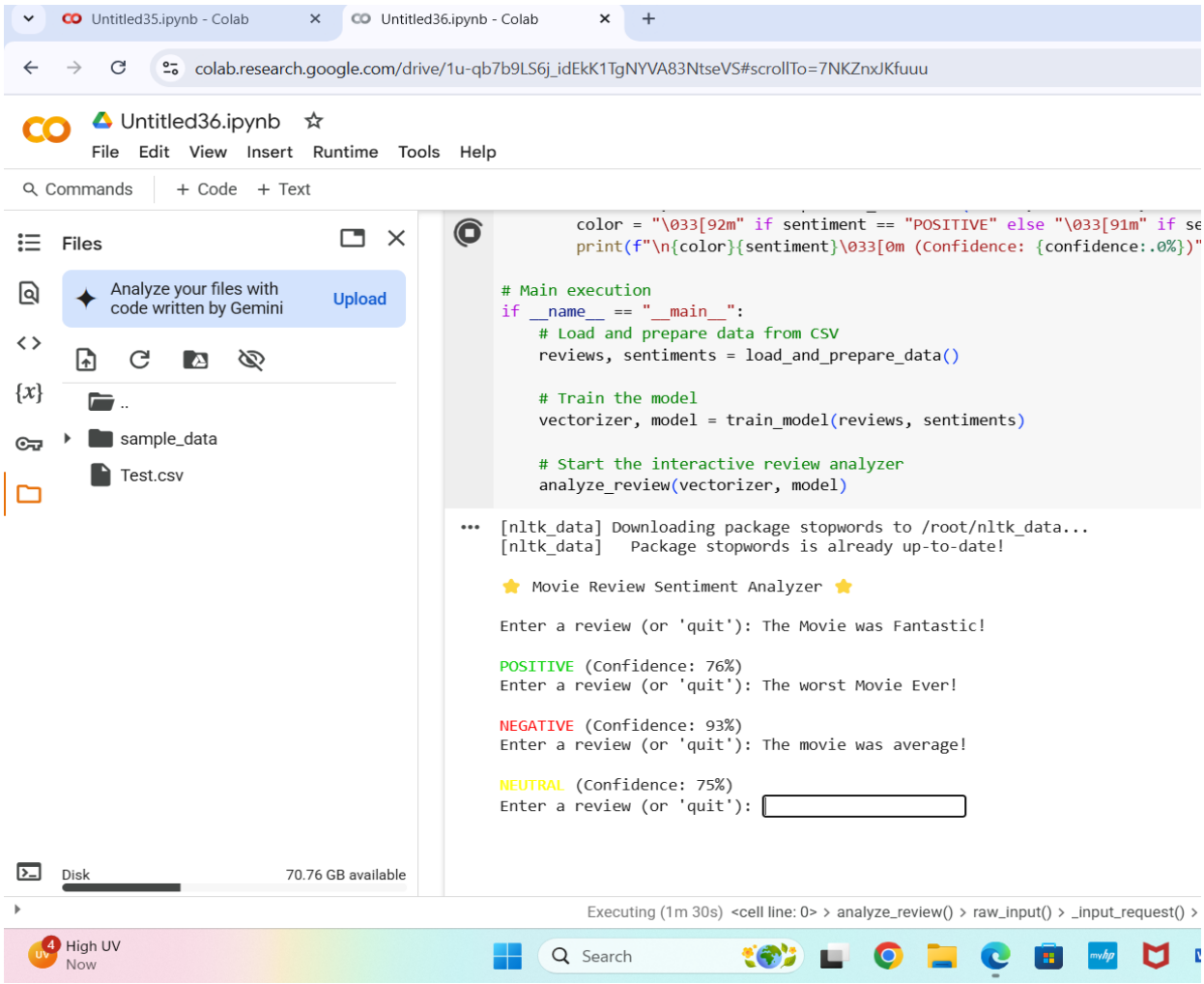
```
import re
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from nltk.corpus import stopwords
import nltk
import pandas as pd

nltk.download('stopwords')

def clean_text(text):
    text = text.lower()
    text = re.sub(r'<[^>]+>|https?://\S+|www\.\S+|[^\w\s]|\d+', ' ', text)
    text = re.sub(r'\s+', ' ', text).strip()

    stops = set(stopwords.words('english')) - {'not', 'no', 'never'}
    words = [word for word in text.split() if word not in stops]
    return ' '.join(words)
```

```
def load_and_prepare_data():
    # Load the CSV file
    df = pd.read_csv('/content/Test.csv')

    # Clean the reviews
    df['cleaned_text'] = df['text'].apply(clean_text)

    # Map labels: 0 (negative) and 1 (positive)
    # Since the model expects three classes (0=negative, 1=positive, 2=neutral),
    # we'll use only 0 and 1 from the CSV and handle neutral separately
    reviews = df['cleaned_text'].tolist()
    sentiments = df['label'].tolist()

    # Add some neutral examples to balance the dataset for three-class classification
    neutral_reviews = [
        "It was okay, nothing special.",
        "Average movie with some flaws.",
        "Not great but not terrible.",
        "Mediocre at best.",
        "Had potential but fell short."
    ]
    neutral_sentiments = [2] * len(neutral_reviews)

    reviews.extend([clean_text(review) for review in neutral_reviews])
    sentiments.extend(neutral_sentiments)
```

```
    return reviews, sentiments


def train_model(reviews, sentiments):
    vectorizer = TfidfVectorizer(ngram_range=(1, 2), max_features=5000)
    X = vectorizer.fit_transform(reviews)
    y = np.array(sentiments)
    model = LogisticRegression(class_weight='balanced', max_iter=1000)
    model.fit(X, y)
    return vectorizer, model


def predict_sentiment(text, vectorizer, model):
    cleaned = clean_text(text)
    vec = vectorizer.transform([cleaned])
    proba = model.predict_proba(vec)[0]
    pred = model.predict(vec)[0]


    labels = ["NEGATIVE", "POSITIVE", "NEUTRAL"]
    return labels[pred], max(proba)


def analyze_review(vectorizer, model):
    print("\n⭐ Movie Review Sentiment Analyzer ⭐\n")
    while True:
        review = input("Enter a review (or 'quit'): ").strip()
        if review.lower() == 'quit':
            break
```

```
        if len(review) < 10:
            print("Please enter a longer review.")
            continue

        sentiment, confidence = predict_sentiment(review, vectorizer, model)
        color = "\033[92m" if sentiment == "POSITIVE" else "\033[91m" if sentiment == "NEGATIVE" else "\033[93m"
        print(f"\n{color}{sentiment}\033[0m (Confidence: {confidence:.0%})")

# Main execution
if __name__ == "__main__":
    # Load and prepare data from CSV
    reviews, sentiments = load_and_prepare_data()

    # Train the model
    vectorizer, model = train_model(reviews, sentiments)

    # Start the interactive review analyzer
    analyze_review(vectorizer, model)
```

Output:


Untitled35.ipynb - Colab
Untitled36.ipynb - Colab
colab.research.google.com/drive/1u-qb7b9LS6j_idEkK1TgNYVA83NtseVS#scrollTo=7NKZnxJKfuuu
Untitled36.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
Files
Analyze your files with code written by Gemini
Upload
sample_data
Test.csv
Disk 70.76 GB available
color = "\033[92m" if sentiment == "POSITIVE" else "\033[91m" if se
print(f"\n{color}{sentiment}\033[0m (Confidence: {confidence:.0%})"
# Main execution
if __name__ == "__main__":
# Load and prepare data from CSV
reviews, sentiments = load_and_prepare_data()
Train the model
vectorizer, model = train_model(reviews, sentiments)
# Start the interactive review analyzer
analyze_review(vectorizer, model)
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
Movie Review Sentiment Analyzer
Enter a review (or 'quit'): The Movie was Fantastic!
POSITIVE (Confidence: 76%)
Enter a review (or 'quit'): The worst Movie Ever!
NEGATIVE (Confidence: 93%)
Enter a review (or 'quit'): The movie was average!
NEUTRAL (Confidence: 75%)
Enter a review (or 'quit'):
Executing (1m 30s) <cell line: 0> > analyze_review() > raw_input() > _input_request() >
High UV Now
Search