

CASE STUDY

Sentiment Analysis Program in Python on Amazon Reviews Dataset

HU22CSEN0100999

ESHWAR DESHMUKH CHAVAN

Overview of the Tool: Sentiment Analysis

Sentiment analysis, also known as opinion mining, is a crucial technique within the field of Natural Language Processing (NLP) that systematically extracts subjective information from text data. By determining the sentiment or emotional tone conveyed within a piece of text, sentiment analysis plays a vital role across numerous industries, enabling the transformation of qualitative feedback into actionable insights. This technique classifies text into three primary sentiment categories: positive, negative, or neutral. As a result, it allows businesses, researchers, and individuals to gain deeper insights into the emotions embedded in large-scale textual data.

The application of sentiment analysis extends across diverse sectors, ranging from customer feedback analysis to social media sentiment monitoring. This versatility makes it an essential tool for understanding public opinion and trends. In this project, we apply sentiment analysis to Amazon customer reviews with the goal of classifying reviews as either "positive" or "negative." The extensive dataset provided by Amazon serves as a robust foundation for training and evaluating sentiment analysis models, unveiling general sentiments and patterns that can inform various applications, such as identifying product improvement areas and highlighting key drivers of customer satisfaction.

Applications of Sentiment Analysis

The versatility of sentiment analysis allows it to be effectively utilized across various industries, supporting distinct goals and use cases:

1. **Customer Feedback:** By analyzing reviews and feedback, companies can gain valuable insights into customer perceptions of their products or services. This analysis helps identify common pain points or cherished features, directly influencing product development and customer service strategies.
 - Example: A smartphone manufacturer analyzes customer reviews to discover that users frequently mention issues with battery life. This insight prompts the company to prioritize battery improvements in its next model, leading to enhanced customer satisfaction.
2. **Social Media Monitoring:** Organizations utilize sentiment analysis on social media data to gauge public opinion, track brand perception, and identify emerging trends. Real-time monitoring can highlight shifts in sentiment surrounding a brand or product, allowing companies to proactively manage public relations and enhance customer loyalty.
 - Example: A fashion retailer notices a spike in negative sentiment regarding a new clothing line on Twitter. The company quickly addresses customer concerns through a public statement and offers discounts to dissatisfied customers, which helps mitigate negative publicity.
3. **Market Research:** Sentiment analysis assists market researchers in understanding consumer preferences, improving survey analysis, and making informed business decisions. For instance, analyzing sentiment trends in product reviews or survey data can guide companies in tailoring marketing strategies or prioritizing product features based on consumer desires.
 - Example: A beverage company analyzes consumer sentiment from product reviews and discovers that customers love a specific flavor combination but dislike the packaging. This information drives the company to redesign its packaging while promoting the favored flavor, improving sales.

4. **Financial Markets:** The finance sector increasingly leverages sentiment analysis to gauge public sentiment toward companies or economic events. Understanding sentiment trends in news articles or social media can provide insights for forecasting stock movements or consumer behavior, helping traders and investors make more informed decisions.
 - Example: A hedge fund employs sentiment analysis on news articles related to a tech giant's quarterly earnings report. Positive sentiment spikes lead them to invest heavily in the company's stock before the price rises significantly after the earnings announcement.
5. **Customer Support:** Sentiment analysis can enhance customer service by prioritizing responses based on feedback sentiment. Negative feedback can be flagged for urgent attention, while positive feedback may help identify effective support strategies. This approach optimizes the customer experience by ensuring that significant issues are addressed promptly, potentially reducing customer churn and increasing satisfaction.
 - Example: An airline uses sentiment analysis to monitor customer feedback during a travel disruption. Negative comments are quickly flagged for the customer service team to address, ensuring timely communication with affected passengers and improving their overall experience.

Technical Foundations of Sentiment Analysis

Sentiment analysis models can be developed using a variety of approaches, from basic lexicon-based systems to advanced deep learning models. The choice of technique depends on the language's complexity, the data volume, and the desired accuracy.

1. **Lexicon-based Analysis:** This method utilizes predefined dictionaries containing words associated with positive, negative, or neutral sentiment scores. By counting or weighting the occurrences of sentiment-laden words in a text, lexicon-based methods assign an

overall sentiment score. However, while straightforward, this approach often struggles with context, sarcasm, and nuanced expressions, limiting its effectiveness.

2. **Machine Learning Models:** Machine learning-based sentiment analysis relies on labeled datasets to learn patterns corresponding to different sentiment classes. Common models include:
 - **Naive Bayes:** A probabilistic model that assumes feature independence. It is efficient and interpretable, making it a solid baseline for text classification tasks.
 - **Support Vector Machines (SVM):** A classifier that identifies the optimal hyperplane separating sentiment classes in feature space, often achieving high accuracy in sentiment classification.
 - **Logistic Regression:** Frequently used for binary classification, it models the probability of a sample belonging to a specific class, making it useful for distinguishing between positive and negative sentiments.
3. **Deep Learning Models:** Advanced models such as Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), and transformers like BERT capture intricate patterns in text by leveraging contextual information. These models are particularly effective for complex sentiment analysis tasks but typically require more computational resources and larger datasets.

Tool Selection for Sentiment Analysis of Amazon Reviews

For this project, we have chosen a machine learning approach utilizing the Naive Bayes classifier in conjunction with TF-IDF vectorization. This combination is ideal for efficiently handling text classification tasks while minimizing the computational costs associated with deep learning models.

- **Naive Bayes Classifier:** Known for its effectiveness and interpretability in text classification, the Naive Bayes classifier operates on the assumption that each word is independent given the

sentiment label. This makes it highly suitable for document classification tasks, yielding accurate results with relatively low computational overhead. Its simplicity and speed make it a popular choice for applications dealing with large datasets, such as Amazon reviews.

- **TF-IDF Vectorization:** The Term Frequency-Inverse Document Frequency (TF-IDF) technique transforms text data into a numerical format by quantifying the importance of words within each document and across the entire dataset. This transformation mitigates the impact of common, uninformative words (such as "the" or "is") and allows the model to concentrate on terms that meaningfully differentiate between sentiments, thereby enhancing classification accuracy.

Together, the Naive Bayes classifier and TF-IDF vectorization provide a robust yet computationally efficient solution for sentiment analysis on large textual datasets. This approach enables accurate classification of Amazon reviews, facilitating scalable sentiment analysis for practical applications.

CODE:

```
# Import necessary libraries
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, accuracy_score
from sklearn.model_selection import GridSearchCV
import re

# Load the datasets (adjust the paths as needed)
train_data = pd.read_csv('/content/drive/MyDrive/archive (4)/train.csv')
test_data = pd.read_csv('/content/drive/MyDrive/archive (4)/test.csv')

# Rename columns for clarity
train_data.columns = ['Sentiment', 'Review_Title', 'Review_Text']
test_data.columns = ['Sentiment', 'Review_Title', 'Review_Text']

# Preprocess text function
def preprocess_text(text):
    text = text.lower() # Convert to lowercase
    text = re.sub(r'^\w\s', '', text) # Remove punctuation
    return text

# Apply preprocessing
train_data['Review_Text'] = train_data['Review_Text'].fillna('').apply(preprocess_text)
test_data['Review_Text'] = test_data['Review_Text'].fillna('').apply(preprocess_text)

# Prepare the training and testing features and labels
X_train = train_data['Review_Text']
y_train = train_data['Sentiment'].replace({1: 0, 2: 1}) # Encode sentiment: 1 (negative) -> 0, 2 (positive) -> 1
```

```

X_test = test_data['Review_Text']
y_test = test_data['Sentiment'].replace({1: 0, 2: 1})

# Simplified pipeline without GridSearchCV for initial testing
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words='english', max_features=5000, ngram_range=(1, 2))),
    ('nb_classifier', MultinomialNB(alpha=1.0)) # Fixed parameters
])

# Train the model on the training data
pipeline.fit(X_train, y_train)

# Predict on the test set
y_pred = pipeline.predict(X_test)

# Show the accuracy score
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")

# Classification report
print("\nClassification Report:\n", classification_report(y_test, y_pred, target_names=['Negative', 'Positive']))

# Function for predicting sentiment of new reviews
def predict_sentiment(review_text):
    prediction = pipeline.predict([preprocess_text(review_text)]) # Preprocess new review text
    sentiment = "Positive" if prediction[0] == 1 else "Negative"
    print(f"\nThe review sentiment is: {sentiment}")

# Example interactive input
while True:
    review = input("\nEnter a review (or type 'exit' to quit): ")

```

```

# Example interactive input
while True:
    review = input("\nEnter a review (or type 'exit' to quit): ")
    if review.lower() == 'exit':
        break
    predict_sentiment(review)

```

PROCEDURE

Step 1: Importing Libraries

import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.naive_bayes import MultinomialNB

from sklearn.pipeline import Pipeline

from sklearn.metrics import classification_report, accuracy_score

import re

Libraries Used:

- pandas: For data manipulation and analysis, particularly for handling the datasets.
- TfidfVectorizer: Converts text data into numerical features by calculating the Term Frequency-Inverse Document Frequency (TF-IDF).
- MultinomialNB: Implements the Naive Bayes classifier suitable for multinomially distributed data, often used for text classification.
- Pipeline: Combines multiple steps into a single object to streamline the workflow for training and prediction.
- accuracy_score and classification_report: Evaluate the model's performance by measuring accuracy and providing detailed classification metrics.
- re: Regular expressions for text preprocessing.

Step 2: Loading and Inspecting the Datasets

```
train_data = pd.read_csv('/content/drive/MyDrive/archive
(4)/train.csv')
```

```
test_data = pd.read_csv('/content/drive/MyDrive/archive (4)/test.csv')
```

```
train_data.columns = ['Sentiment', 'Review_Title', 'Review_Text']
```

```
test_data.columns = ['Sentiment', 'Review_Title', 'Review_Text']
```

The dataset includes:

- Sentiment: The review's label (1 for negative, 2 for positive).
- Review_Title and Review_Text: Text features analyzed for sentiment.

Step 3: Preprocessing Text Data

Preprocess text function

```
def preprocess_text(text):
```

```
    text = text.lower() # Convert to lowercase
```

```
    text = re.sub(r'^\w\s', "", text) # Remove punctuation
```

```
    return text
```

Apply preprocessing

```
train_data['Review_Text'] =
```

```
train_data['Review_Text'].fillna("").apply(preprocess_text)
```

```
test_data['Review_Text'] =
```

```
test_data['Review_Text'].fillna("").apply(preprocess_text)
```

Purpose of Preprocessing:

- Converts text to lowercase to ensure uniformity.
- Removes punctuation to focus on the words, enhancing text data quality.

Step 4: Preparing Data for Training and Testing

```
X_train = train_data['Review_Text']
```

```
y_train = train_data['Sentiment'].replace({1: 0, 2: 1}) # Encode
```

```
sentiment: 1 (negative) -> 0, 2 (positive) -> 1
```

```
X_test = test_data['Review_Text']
```

```
y_test = test_data['Sentiment'].replace({1: 0, 2: 1})
```


Encoding:

- Sentiment labels are encoded as 0 (negative) and 1 (positive), simplifying binary classification.

Step 5: Defining the Pipeline

python

Copy code

```
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words='english', max_features=5000,
ngram_range=(1, 2))),
    ('nb_classifier', MultinomialNB(alpha=1.0)) # Fixed parameters
])
```

Pipeline Components:

1. TF-IDF Vectorizer: Converts text to numeric data, removes English stop words, and captures unigrams and bigrams.
2. Naive Bayes Classifier: Efficient for text classification, particularly with sparse data.

Step 6: Training the Model

```
pipeline.fit(X_train, y_train)
```

Training Purpose:

- The model learns patterns between the text features and sentiment labels to make predictions on unseen data.

Step 7: Model Evaluation

```

y_pred = pipeline.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")

print("\nClassification Report:\n", classification_report(y_test,
y_pred, target_names=['Negative', 'Positive']))

```

Evaluation Metrics:

1. Accuracy: Measures the overall accuracy of predictions, indicating the proportion of correct predictions.
2. Classification Report: Provides precision, recall, and F1-score for both classes, helping assess model performance in detail.

Step 8: Defining a Function for User Input

```

def predict_sentiment(review_text):
    prediction = pipeline.predict([preprocess_text(review_text)]) #
    Preprocess new review text
    sentiment = "Positive" if prediction[0] == 1 else "Negative"
    print(f"\nThe review sentiment is: {sentiment}")

```

Function Purpose:

- Allows input of new reviews and outputs the predicted sentiment based on the trained model, applying the same preprocessing.

Step 9: Adding Interactive User Input

while True:

```
review = input("\nEnter a review (or type 'exit' to quit): ")  
if review.lower() == 'exit':  
    break  
predict_sentiment(review)
```

User Interaction:

- Provides a simple interface for users to test the model interactively by entering reviews, making it user-friendly.

Advantages and Challenges

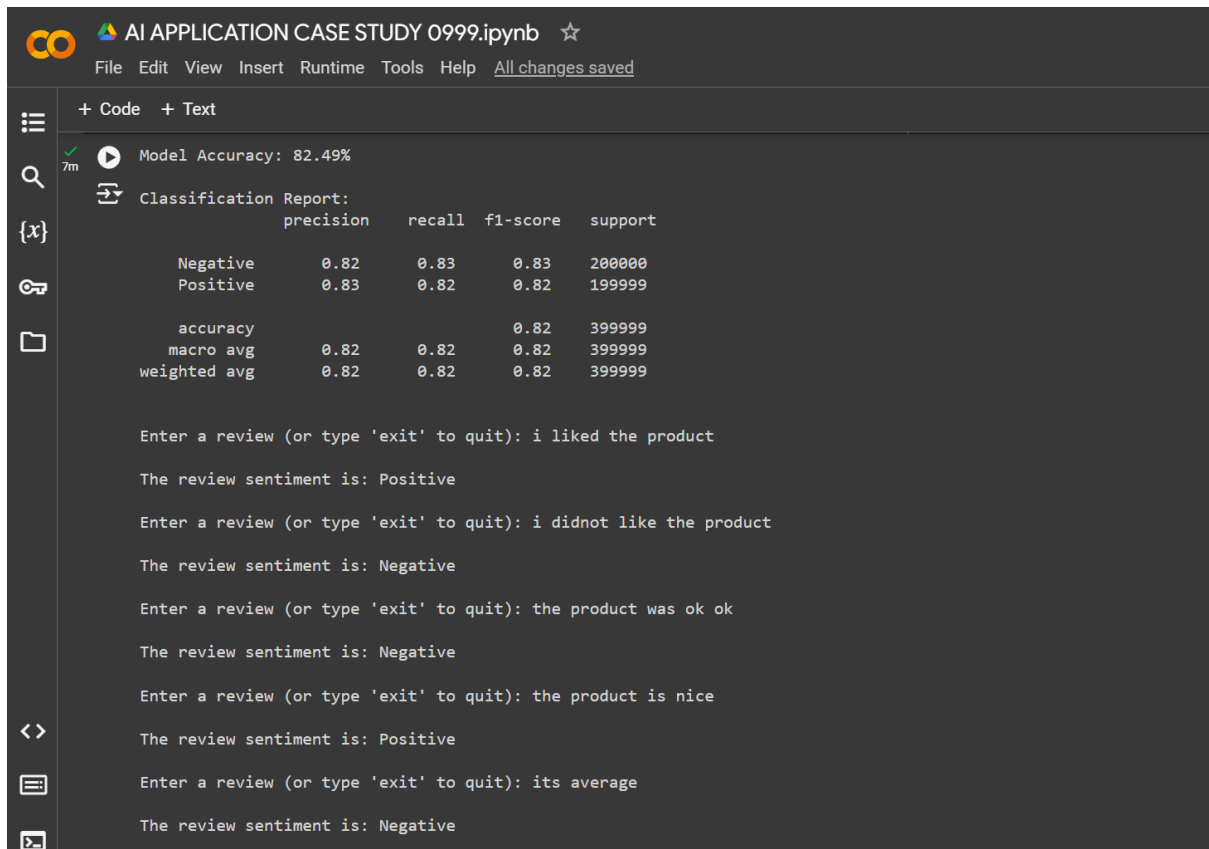
Advantages:

- **Scalability:** Processes large volumes of text, offering insights impractical to obtain manually.
- **Real-Time Analysis:** Once trained, the model can perform sentiment classification in real-time.
- **Quantitative Insight from Qualitative Data:** Transforms feedback into quantifiable metrics.

Challenges:

- **Context and Sarcasm:** The model may misinterpret sarcasm or complex sentiment.
- **Data Imbalance:** Imbalanced datasets may bias results, requiring careful model tuning.

OUTPUT SCREENSHOTS:



AI APPLICATION CASE STUDY 0999.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

7m ✓ Model Accuracy: 82.49%

Classification Report:

	precision	recall	f1-score	support
Negative	0.82	0.83	0.83	200000
Positive	0.83	0.82	0.82	199999
accuracy			0.82	399999
macro avg	0.82	0.82	0.82	399999
weighted avg	0.82	0.82	0.82	399999

Enter a review (or type 'exit' to quit): i liked the product

The review sentiment is: Positive

Enter a review (or type 'exit' to quit): i didnot like the product

The review sentiment is: Negative

Enter a review (or type 'exit' to quit): the product was ok ok

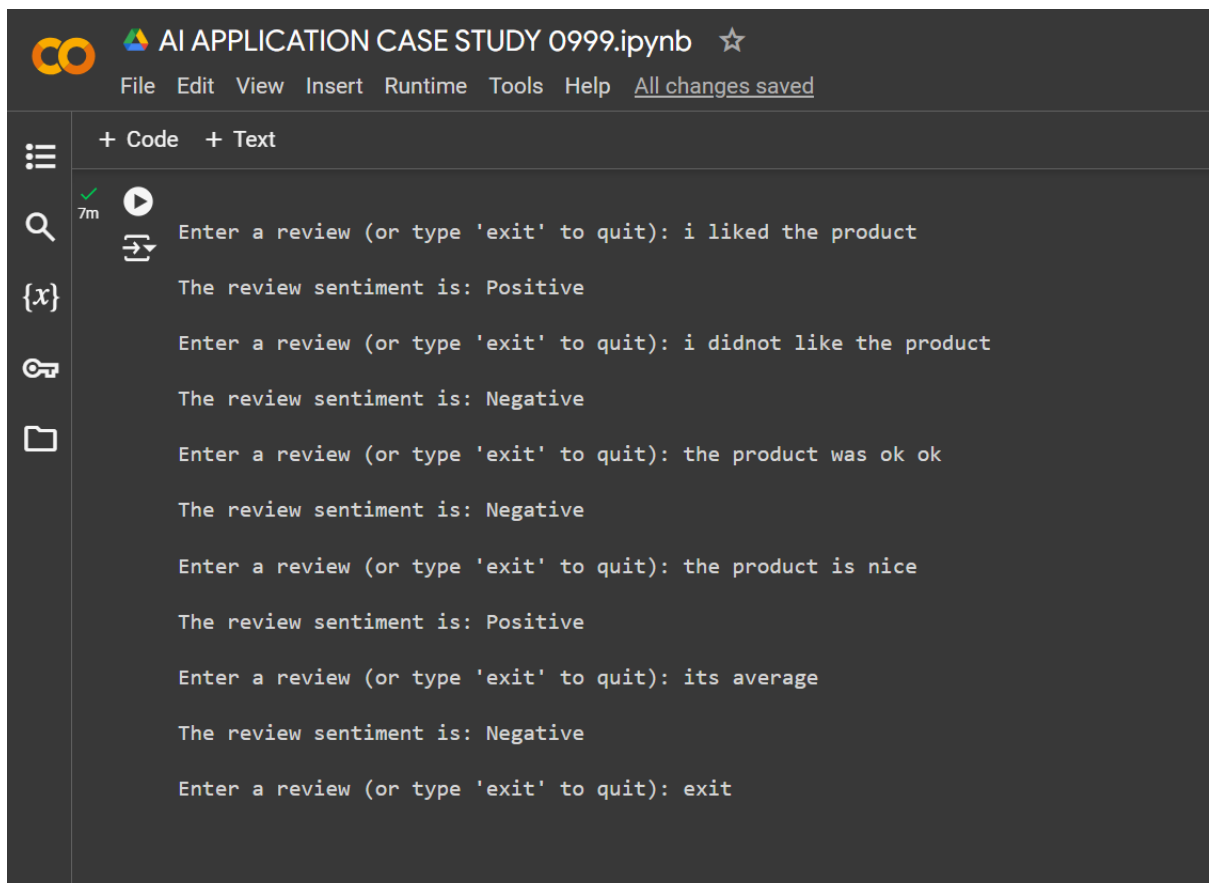
The review sentiment is: Negative

Enter a review (or type 'exit' to quit): the product is nice

The review sentiment is: Positive

Enter a review (or type 'exit' to quit): its average

The review sentiment is: Negative



AI APPLICATION CASE STUDY 0999.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

7m ✓

Enter a review (or type 'exit' to quit): i liked the product

The review sentiment is: Positive

Enter a review (or type 'exit' to quit): i didnot like the product

The review sentiment is: Negative

Enter a review (or type 'exit' to quit): the product was ok ok

The review sentiment is: Negative

Enter a review (or type 'exit' to quit): the product is nice

The review sentiment is: Positive

Enter a review (or type 'exit' to quit): its average

The review sentiment is: Negative

Enter a review (or type 'exit' to quit): exit

Conclusion

This sentiment analysis tool exemplifies the impact of machine learning in converting vast amounts of textual data into actionable insights. By systematically analyzing Amazon customer reviews, businesses can gain a deeper understanding of customer preferences and sentiments, ultimately leading to better product offerings and enhanced customer satisfaction. This project not only showcases technical proficiency in NLP and machine learning but also highlights the practical applications of these technologies in the real world, empowering stakeholders to make data-driven decisions..