Natural Language Processing Project

III-II B.Tech (Artificial Intelligence & Machine Learning)

Malla Reddy University

Sentiment Analysis on Customer Reviews

Abstract

Sentiment analysis is a Natural Language Processing (NLP) task aimed at identifying and classifying the sentiment expressed in textual data. This project focuses on analyzing customer reviews to determine whether they convey positive, negative, or neutral sentiments. The process involves text preprocessing techniques such as cleaning, tokenization, lemmatization, and vectorization, followed by the development and evaluation of machine learning models for classification. The outcome of this project provides insights into customer opinions, aiding businesses in decision-making and improving customer satisfaction.

Methodology

1. Data Collection

Source: Use publicly available datasets such as Kaggle's customer review datasets or scrape data from e-commerce websites.

2. Data Preprocessing

a. Cleaning the Text:

Remove special characters, numbers, and punctuation.

Consert text to lowercase

Remove stopwords (eg... "is." "the," "and")

b. Tokenization:

Split the text into individual words or tokens.

c. Lemmatization:

Reduce words to their base or reat form (e it. "running" "run")

Ans:

Sentiment Analysis on Customer Reviews

Abstract

Sentiment analysis is a Natural Language Processing (NLP) task aimed at identifying and classifying sentiments expressed in textual data. This project focuses on analyzing customer reviews to determine whether they convey positive, negative, or neutral sentiments. The process involves text preprocessing techniques such as cleaning, tokenization, lemmatization, and vectorization, followed by the development and evaluation of machine learning models for classification. The insights from this project can aid businesses in understanding customer opinions, improving decision-making, and enhancing customer satisfaction.

Introduction

Customer feedback provides valuable insights into the quality of products and services. By analyzing customer reviews, businesses can identify pain points, improve their offerings, and make data-driven decisions. This project explores sentiment analysis as a method to classify customer reviews into positive, negative, or neutral sentiments, leveraging machine learning and NLP techniques.

Methodology

1. Data Collection

- **Source:**
  - Use publicly available datasets such as Kaggle's customer review datasets (e.g., Amazon, Yelp, or IMDb reviews).
  - Alternatively, scrape customer reviews from e-commerce platforms like Amazon or Flipkart using tools like BeautifulSoup or Scrapy.
- **Storage:**
  - Save the data in CSV or JSON format for ease of processing.

2. Data Preprocessing

Efficient preprocessing is crucial for converting raw text into structured data usable by machine learning algorithms.

a. Cleaning the Text:

- Remove special characters, numbers, and punctuation.
- Convert all text to lowercase to ensure uniformity.

- Remove common stopwords such as "is," "the," "and."

b. Tokenization:

- Split the text into individual words or tokens using libraries like NLTK or spaCy.

c. Lemmatization:

- Reduce words to their base form using lemmatizers like WordNetLemmatizer or spaCy (e.g., "running" → "run").

d. Vectorization:

- Transform the cleaned text into numerical representations using techniques like:
  - **Bag of Words (BoW)**
  - **TF-IDF (Term Frequency-Inverse Document Frequency)**
  - **Word Embeddings (Word2Vec, GloVe)**

3. Model Development

Train machine learning models to classify sentiments. The following approaches can be used:

a. Machine Learning Models:

- Logistic Regression
- Support Vector Machine (SVM)
- Random Forest
- Naïve Bayes

b. Deep Learning Models:

- Recurrent Neural Networks (RNNs)
- Long Short-Term Memory Networks (LSTMs)
- Bidirectional Encoder Representations from Transformers (BERT)

c. Model Training and Testing:

- Split the data into training and testing sets (e.g., 80-20 split).
- Train the model using the training set and evaluate performance on the testing set.

d. Hyperparameter Tuning:

- Use techniques like GridSearchCV or RandomizedSearchCV to optimize model parameters.

4. Model Evaluation

Evaluate the models based on performance metrics, such as:

- **Accuracy:** Percentage of correctly classified sentiments.
- **Precision:** Focus on positive predictions.

- **Recall:** Focus on identifying actual positive cases.
- **F1 Score:** Harmonic mean of precision and recall.
- **Confusion Matrix:** To visualize true positives, true negatives, false positives, and false negatives.

5. Visualization and Insights

- **Visualizations:**
  - Use libraries like Matplotlib and Seaborn to create bar charts, word clouds, and sentiment distributions.
- **Insights:**
  - Highlight trends in customer opinions.
  - Provide actionable insights for businesses to address customer concerns or enhance strengths.

Results

- Present the best-performing model with its accuracy and other metrics.
- Share insights derived from the analysis (e.g., "80% of customers expressed positive sentiments about Product X").

Conclusion

The sentiment analysis project successfully classified customer reviews into positive, negative, and neutral sentiments. By leveraging NLP and machine learning techniques, businesses can better understand customer feedback, address areas of improvement, and enhance overall customer satisfaction.

Future Work

- Explore multilingual sentiment analysis to support non-English reviews.
- Implement advanced deep learning models such as transformers (e.g., GPT, BERT).
- Integrate sentiment analysis into a real-time dashboard for continuous monitoring of customer feedback.

Code And Output:

NLP Case Study

Sentiment Analysis on Customer Reviews

Importing libraries

In [1]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, accuracy_score
from sklearn.ensemble import RandomForestClassifier

# Download necessary NLTK resources
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\edbid\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\edbid\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\edbid\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[1]:
```
True
```

Load the dataset

In [2]:
```
# Load the dataset
df = pd.read_csv("amazon_reviews.csv")

# Display the first few rows of the dataset
print("Dataset Loaded:")
print(df.head())

# Check for missing values in the `reviewText` column
missing_count = df['reviewText'].isnull().sum()
print(f"\nNumber of missing values in 'reviewText': {missing_count}")
```

```
Dataset Loaded:
   Unnamed: 0  reviewerName  overall  \
0           0           NaN      4.0
1           1          0mie      5.0
2           2           1K3      4.0
3           3           1m2      5.0
4           4  2&amp;1/2Men      5.0

                                          reviewText  reviewTime  day_diff  \
0                                         No issues.  2014-07-23       138
1  Purchased this for my device, it worked as adv...  2013-10-25       409
2  it works as expected. I should have sprung for...  2012-12-23       715
3  This think has worked out great.Had a diff. br...  2013-11-21       382
4  Bought it with Retail Packaging, arrived legit...  2013-07-13       513

   helpful_yes  helpful_no  total_vote  score_pos_neg_diff  \
0            0           0           0                   0
1            0           0           0                   0
2            0           0           0                   0
3            0           0           0                   0
4            0           0           0                   0

   score_average_rating  wilson_lower_bound
0                   0.0                 0.0
1                   0.0                 0.0
2                   0.0                 0.0
3                   0.0                 0.0
4                   0.0                 0.0

Number of missing values in 'reviewText': 1
```

In [3]:
```
print(df.head())
```

```
   Unnamed: 0  reviewerName  overall  \
0           0           NaN      4.0
1           1          0mie      5.0
2           2           1K3      4.0
3           3           1m2      5.0
4           4  2&amp;1/2Men      5.0

                                          reviewText  reviewTime  day_diff  \
0                                        No issues.  2014-07-23       138
1  Purchased this for my device, it worked as adv...  2013-10-25       409
2  it works as expected. I should have sprung for...  2012-12-23       715
3  This think has worked out great.Had a diff. br...  2013-11-21       382
4  Bought it with Retail Packaging, arrived legit...  2013-07-13       513

   helpful_yes  helpful_no  total_vote  score_pos_neg_diff  \
0            0           0           0                   0
1            0           0           0                   0
2            0           0           0                   0
3            0           0           0                   0
4            0           0           0                   0

   score_average_rating  wilson_lower_bound
0                   0.0                 0.0
1                   0.0                 0.0
2                   0.0                 0.0
3                   0.0                 0.0
4                   0.0                 0.0
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4915 entries, 0 to 4914
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Unnamed: 0            4915 non-null   int64  
 1   reviewerName          4914 non-null   object 
 2   overall               4915 non-null   float64
 3   reviewText            4914 non-null   object 
 4   reviewTime            4915 non-null   object 
 5   day_diff              4915 non-null   int64  
 6   helpful_yes           4915 non-null   int64  
 7   helpful_no            4915 non-null   int64  
 8   total_vote            4915 non-null   int64  
 9   score_pos_neg_diff    4915 non-null   int64  
 10  score_average_rating  4915 non-null   float64
 11  wilson_lower_bound    4915 non-null   float64
dtypes: float64(3), int64(6), object(3)
memory usage: 460.9+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

	Unnamed: 0	overall	day_diff	helpful_yes	helpful_no	total_vote	score_pos_neg_diff	score_average_rating	wilson_lower_bound
count	4915.000000	4915.000000	4915.000000	4915.000000	4915.000000	4915.000000	4915.000000	4915.000000	4915.000000
mean	2457.000000	4.587589	437.367040	1.311089	0.210376	1.521465	1.100712	0.075468	0.020053
std	1418.982617	0.996845	209.439871	41.619161	4.023296	44.123095	39.367949	0.256062	0.077187
min	0.000000	1.000000	1.000000	0.000000	0.000000	0.000000	-130.000000	0.000000	0.000000
25%	1228.500000	5.000000	281.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	2457.000000	5.000000	431.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	3685.500000	5.000000	601.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	4914.000000	5.000000	1064.000000	1952.000000	183.000000	2020.000000	1884.000000	1.000000	0.957544

In [6]:
```
df.columns
```

Out[6]:
```
Index(['Unnamed: 0', 'reviewerName', 'overall', 'reviewText', 'reviewTime',
       'day_diff', 'helpful_yes', 'helpful_no', 'total_vote',
       'score_pos_neg_diff', 'score_average_rating', 'wilson_lower_bound'],
      dtype='object')
```

Handle Missing Values

In [7]:
```
# Drop rows with missing 'reviewText'
df = df.dropna(subset=['reviewText'])

# Verify that missing values are removed
print("\nDataset after removing rows with missing 'reviewText':")
print(df.info())
```

```
Dataset after removing rows with missing 'reviewText':
<class 'pandas.core.frame.DataFrame'>
Index: 4914 entries, 0 to 4914
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Unnamed: 0            4914 non-null   int64  
 1   reviewerName          4913 non-null   object 
 2   overall               4914 non-null   float64
 3   reviewText            4914 non-null   object 
 4   reviewTime            4914 non-null   object 
 5   day_diff              4914 non-null   int64  
 6   helpful_yes           4914 non-null   int64  
 7   helpful_no            4914 non-null   int64  
 8   total_vote            4914 non-null   int64  
 9   score_pos_neg_diff    4914 non-null   int64  
 10  score_average_rating  4914 non-null   float64
 11  wilson_lower_bound    4914 non-null   float64
dtypes: float64(3), int64(6), object(3)
memory usage: 499.1+ KB
None
```

Define Preprocessing Functions

In [8]:
```
# Initialize the lemmatizer and stop words
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))
```

```
# Function to clean text
def preprocess_text(text):
    # Remove special characters, numbers, and punctuation
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    # Convert to lowercase
    text = text.lower()
    # Tokenization
    tokens = word_tokenize(text)
    # Remove stopwords and lemmatize
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in stop_words]
    return ' '.join(tokens)
```

Apply Preprocessing to the Text Data

In [9]:
```
# Apply the preprocessing function to the 'reviewText' column
df['cleaned_review'] = df['reviewText'].apply(preprocess_text)

# Display the original and cleaned text for verification
print("\nOriginal and Cleaned Reviews:")
print(df[['reviewText', 'cleaned_review']].head())
```

```
Original and Cleaned Reviews:
                                          reviewText  \
0                                         No issues.
1  Purchased this for my device, it worked as adv...
2  it works as expected. I should have sprung for...
3  This think has worked out great.Had a diff. br...
4  Bought it with Retail Packaging, arrived legit...

                                      cleaned_review
0                                              issue
1  purchased device worked advertised never much ...
2  work expected sprung higher capacity think mad...
3  think worked greathad diff bran gb card went s...
4  bought retail packaging arrived legit orange e...
```

Save the Cleaned Dataset (Optional)

In [10]:
```
# Save the cleaned dataset to a new CSV file
df.to_csv("cleaned_amazon_reviews.csv", index=False)
```

```
print("\nCleaned dataset saved as 'cleaned_amazon_reviews.csv'.")
```

```
Cleaned dataset saved as 'cleaned_amazon_reviews.csv'.
```

Create Sentiment Labels

In [11]:

```
# Step 1: Load the preprocessed dataset
df = pd.read_csv("cleaned_amazon_reviews.csv")

# Step 2: Create Sentiment Labels
def assign_sentiment(overall):
    if overall >= 4:
        return "Positive"
    elif overall == 3:
        return "Neutral"
    else:
        return "Negative"

df['sentiment'] = df['overall'].apply(assign_sentiment)

# Step 3: Text Vectorization (TF-IDF)
tfidf = TfidfVectorizer(max_features=5000, stop_words='english')
X = tfidf.fit_transform(df['cleaned_review'])  # Use cleaned text column
y = df['sentiment']

# Step 4: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Train the Model

In [12]:

```
# Step 5: Train the Model
model = MultinomialNB()
model.fit(X_train, y_train)
```

Out[12]:

```
▾ MultinomialNB
MultinomialNB()
```

Evaluate the Model

In [13]:
```
# Step 6: Evaluate the Model
y_pred = model.predict(X_test)
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))
```

```
Classification Report:
               precision    recall  f1-score   support

    Negative       0.00      0.00      0.00        56
     Neutral       0.00      0.00      0.00        30
    Positive       0.91      1.00      0.95       897

    accuracy                           0.91       983
   macro avg       0.30      0.33      0.32       983
weighted avg       0.83      0.91      0.87       983

Accuracy Score: 0.9125127161749745
```

```
C:\Users\edbid\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-def
ined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\edbid\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-def
ined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\edbid\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-def
ined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Predict Sentiments for New Reviews

In [14]:
```
# Step 7: Predict Sentiments for New Reviews
new_reviews = ["The product is excellent and exceeded my expectations.",
               "Worst purchase ever. Bad product.",
               "It's okay, but could be better."]
new_reviews_cleaned = [" ".join(word for word in review.lower().split() if word.isalnum()) for review in new_reviews]
new_reviews_tfidf = tfidf.transform(new_reviews_cleaned)
predictions = model.predict(new_reviews_tfidf)

for review, sentiment in zip(new_reviews, predictions):
    print(f"Review: {review}\nSentiment: {sentiment}\n")
```

```
Review: The product is excellent and exceeded my expectations.
Sentiment: Positive

Review: Worst purchase ever. Bad product.
Sentiment: Negative

Review: It's okay, but could be better.
Sentiment: Neutral
```

In [ ]: