Summary report on sentimental analysis

**Vigneswaran Vimalkumar**

Contents

[Description of the Dataset 3](#_Toc160304274)

[Details of the preprocessing steps 3](#_Toc160304275)

[Evaluation of results 3](#_Toc160304276)

[Insights into the model's strengths and limitations 4](#_Toc160304277)

# Description of the Dataset

The dataset is collected from Kaggle website which has a list of over 34,000 consumer reviews for Amazon products. The data was originally sourced from Datafiniti's Product Database and is provided in CSV format. It includes a range of information such as usernames, titles, reviews, ratings, dates, and other relevant attributes associated with each review entry.

# Details of the preprocessing steps

The following steps were taken to preprocess the dataset.

1. Firstly, the code removes stopwords, non-alphabetic characters, and lemmatizing the remaining words.
2. Remove missing value in the from the 'reviews.text' column. For this, dropna function was applied to remove a missing value within the 'reviews.text' column. This will ensure data integrity and provide accurate results when applying the sentimental analysis.
3. Have applied tokenization to break down the text into smaller units and have used SpaCy to tokenise the 'reviews.text' column.
4. Using the preprocess text function to review text in the 'reviews.text' column. The cleaned data is stored in a new column named 'cleaned\_review'.
5. Used the Textblob library to perform sentimental analysis.
6. A TextBlob object is created from the input review text.
7. The polarity score is returned as the output for the sentimental analysis.

# Evaluation of results

The following is my is output result for the sentimental analysis that I have produces:

A screenshot of a computer

Description automatically generated

From evaluation my output results, the first two output seems to be accurate. The first output is showing a positive sentiment for the review of a product. The second output is showing a negative sentiment for the review of a product. However, the sentiment classification for the third review seems inaccurate. While the review contains words like "Better" and "Okay", indicating a potentially neutral sentiment, the model incorrectly classified it as positive. This discrepancy suggests that the model may have misinterpreted certain phrases, highlighting the need for further refinement to better capture nuances in sentiment.

# Insights into the model's strengths and limitations

The following are some of the model’s strengths and limitations:

Strengths:

* TextBlob's sentiment analysis functionality provides quick and efficient sentiment polarity scores.
* The output of the sentiment analysis is easy to interpret with polarity scores being Positive, Negative or Neutral.

Limitations:

* **The model could have difficulty analysing mixed/neutral reviews. As mentioned in the evaluation**, reviews containing mixed sentiments or conflicting statements may cause challenges for accurate sentiment analysis.
* The sentiment analysis may be influenced by biases present in the training data.
* The model may misinterpret sarcastic or ironic statements.

# Code

import pandas as pd

import spacy

from spacytextblob.spacytextblob import SpacyTextBlob

# Load the spaCy model

nlp = spacy.load("en\_core\_web\_sm")

# Add TextBlob capabilities to spaCy pipeline

nlp.add\_pipe('spacytextblob')

# Step 1: Load and preprocess the data

# Assuming you have a dataframe named 'data' with 'review.text' column

def load\_and\_preprocess\_data(data):

# Select the 'review.text' column

reviews\_data = data['reviews.text']

# Remove missing values

clean\_data = data.dropna(subset=['reviews.text'])

return clean\_data

#C:/Users/user/Desktop/Anna/Python/archive/amazon\_product\_reviews.csv

# Step 2: Create a function for sentiment analysis

def analyze\_sentiment(reviews):

doc = nlp(reviews)

sentiment = doc.\_.polarity

return sentiment

# Step 3: Test the sentiment analysis function

def test\_sentiment\_analysis():

sample\_reviews = [

"This product is amazing! I love it.",

"I'm not satisfied with this product. It doesn't work as expected.",

"This product seems, okay."

]

for reviews in sample\_reviews:

sentiment = analyze\_sentiment(reviews)

if sentiment > 0:

print(f"Reviews: {reviews}\nSentiment: Positive\n")

elif sentiment < 0:

print(f"Reviews: {reviews}\nSentiment: Negative\n")

else:

print(f"Reviews {reviews}\nSentiment: Neutral\n")

#print(f"Reviews: {reviews}\nSentiment: Positive\n")

# elif sentiment < 0:

# print(f"Reviews: {reviews}\nSentiment: Negative\n")

#else:

# print(f"Reviews {reviews}\nSentiment: Neutral\n")

# Step 4: Write a brief report

def write\_report():

report = """

Sentiment Analysis Report:

Description of the Dataset

The dataset is collected from Kaggle website which has a list of over 34,000 consumer reviews for Amazon products. The data was originally sourced from Datafiniti's Product Database and is provided in CSV format. It includes a range of information such as usernames, titles, reviews, ratings, dates, and other relevant attributes associated with each review entry.

Details of the preprocessing steps

The following steps were taken to preprocess the dataset.

1. Firstly, the code removes stopwords, non-alphabetic characters, and lemmatizing the remaining words.

2. Remove missing value in the from the 'reviews.text' column. For this, dropna function was applied to remove a missing value within the 'reviews.text' column. This will ensure data integrity and provide accurate results when applying the sentimental analysis.

3. Have applied tokenization to break down the text into smaller units and have used SpaCy to tokenise the 'reviews.text' column.

4. Using the preprocess text function to review text in the 'reviews.text' column. The cleaned data is stored in a new column named 'cleaned\_review'.

5. Used the Textblob library to perform sentimental analysis.

6. A TextBlob object is created from the input review text.

7. The polarity score is returned as the output for the sentimental analysis.

Evaluation of results

The following is my is output result for the sentimental analysis that I have produces:

From evaluation my output results, the first two output seems to be accurate. The first output is showing a positive sentiment for the review of a product. The second output is showing a negative sentiment for the review of a product. However, the sentiment classification for the third review seems inaccurate. While the review contains words like "Better" and "Okay", indicating a potentially neutral sentiment, the model incorrectly classified it as positive. This discrepancy suggests that the model may have misinterpreted certain phrases, highlighting the need for further refinement to better capture nuances in sentiment.

Insights into the model's strengths and limitations

The following are some of the model’s strengths and limitations:

Strengths:

• TextBlob's sentiment analysis functionality provides quick and efficient sentiment polarity scores.

• The output of the sentiment analysis is easy to interpret with polarity scores being Positive, Negative or Neutral.

Limitations:

• The model could have difficulty analysing mixed/neutral reviews. As mentioned in the evaluation, reviews containing mixed sentiments or conflicting statements may cause challenges for accurate sentiment analysis.

• The sentiment analysis may be influenced by biases present in the training data.

• The model may misinterpret sarcastic or ironic statements.

"""

with open("sentiment\_analysis\_report\_vigneswaran.pdf", "w") as file:

file.write(report)

# Step 5: Test similarity of two product reviews

def test\_similarity(data):

review1 = data['reviews.text'][0]

review2 = data['reviews.text'][1]

doc1 = nlp(review1)

doc2 = nlp(review2)

similarity\_score = doc1.similarity(doc2)

print(f"Similarity between review 1 and review 2: {similarity\_score}")

# Load the dataset

data = pd.read\_csv("C:/Users/user/Desktop/Anna/Python/archive/amazon\_product\_reviews.csv")

# Step 1: Load and preprocess the data

clean\_data = load\_and\_preprocess\_data(data)

# Step 2: Create a function for sentiment analysis

# Function 'analyze\_sentiment' defined above

# Step 3: Test the sentiment analysis function

test\_sentiment\_analysis()

# Step 4: Write a brief report

write\_report()

# Step 5: Test similarity of two product reviews

test\_similarity(clean\_data)