INTEL PRODUCT SENTIMENT ANALYSIS FROM ONLINE REVIEWS

-By THE INTEL IGNITERS

S. Pranith Reddy - 21N31A66G4
B. Hemasree Alekhya - 21N31A6628
V. Yaswanth Kowsik Sai - 21N31A66J7
K. Raja Harsha Vardhan – 21N31A6690
G. Sahithi - 21N31A6649

ABSTRACT

In the realm of consumer electronics, user reviews serve as a valuable source of sentiment and feedback. This abstract focuses on sentiment analysis applied to online reviews of Intel products, aiming to extract insights and trends from large volumes of textual data. The study employs natural language processing (NLP) techniques to analyse sentiment polarity, identify key topics, and evaluate overall sentiment trends over time. This analysis explores the sentiment expressed in online reviews of Intel products. By leveraging sentiment analysis techniques, we can gain valuable insights into customer perception of Intel's offerings. This can include identifying strengths and weaknesses of specific products, understanding overall customer satisfaction, and discovering areas for improvement. The analysis can be conducted on reviews from various sources like e-commerce platforms, tech review websites, and social media. By employing techniques like lexicon-based analysis or machine learning models, we can categorize reviews as positive, negative, or neutral.

1. INTRODUCTION

1.1 Purpose:

The purpose of conducting sentiment analysis on Intel products from online reviews is multifaceted. It aims to provide deep insights into how consumers perceive Intel's range of products, ranging from CPUs to GPUs and motherboards. By analyzing sentiment expressed in these reviews, the project seeks to uncover patterns, trends, and recurring themes that influence consumer satisfaction and dissatisfaction. These insights are crucial for Intel as they help in refining product development strategies, identifying areas for enhancement, and aligning marketing efforts to better meet consumer expectations. Additionally, the analysis serves as a barometer of market sentiment, offering valuable intelligence on competitive positioning and potential areas of innovation. Ultimately, the project aims to leverage the power of natural language processing and machine learning to extract actionable insights that drive continuous improvement and customer-centric decision-making within Intel.

1.2 Background of project:

The background of conducting sentiment analysis on Intel products from online reviews stems from the increasing significance of consumer feedback in shaping product development and marketing strategies in the consumer electronics industry. In today's digital age, online reviews serve as a rich source of unfiltered opinions and sentiments expressed by users who have firsthand experience with Intel products. These reviews not only reflect individual experiences but also aggregate into collective insights that can reveal broader trends and sentiments across different product categories. For Intel, understanding these sentiments is crucial for several reasons. It provides a direct line of feedback from users, highlighting what aspects of their products resonate positively or negatively. This feedback can inform iterative improvements to existing products and guide the development of future innovations that better meet consumer needs and preferences

1.3 Scope of project:

The goal of this project involves analyzing sentiment from online reviews of Intel products to gain comprehensive insights into consumer perceptions and preferences. It encompasses collecting and preprocessing a diverse dataset of reviews across platforms, applying advanced natural language processing techniques to classify sentiments (positive, negative, neutral), and identifying key themes and topics discussed by users.

1.4 Project Features:

The project features are as follows:

- **Data Collection and Preprocessing**: Gathering a substantial dataset of online reviews across various platforms for Intel products, ensuring data quality through cleaning, normalization, and standardization processes.
- **Sentiment Analysis Techniques**: Applying natural language processing (NLP) techniques to analyze sentiment polarity (positive, negative, neutral) of reviews. This includes text tokenization, sentiment classification using machine learning models, and sentiment trend analysis over time.

2.SYSTEM REQUIREMENTS

2.1 Hardware Requirements:

The hardware interfaces of this product consist of architecture, processing power, memory, secondary
storage, display adapter, peripherals like CD-ROM drivers, keyboards, pointing devices, network devices, etc.

- Processor: i5 and above
- Ram: 8gb and above
- Hard Disk: 25 GB in local drive

2.2 Software requirements:

- Google Colab

2.2.1 Technologies used

- Python
- NLP

2.2.2 Libraries used

- Python
- Pandas
- Transformers
- Matplotlib and seaborn
- Word cloud
- spaCy
- Scikit-learn

3.IMPLEMENTATION

3.1: SOURCE CODE

```
from google.colab import drive
drive.mount('/content/drive')
from google.colab import files
uploaded=files.upload()
!pip install nltk
import nltk
nltk.download('vader_lexicon')

import pandas as pd
import matplotlib.pyplot as plt
from nltk.sentiment.vader import SentimentIntensityAnalyzer
# Download the VADER lexicon
nltk.download('vader_lexicon') # Download the lexicon data

# Import libraries
import pandas as pd
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import matplotlib.pyplot as plt
```

```
# Load dataset (replace with your file path or URL)
# Example with Google Drive path
file_path =  intel_product_reviews_large.csvdf = pd.read_csv(file_path)

# Initialize the VADER sentiment analyzer
sid = SentimentIntensityAnalyzer()
# Function to perform sentiment analysis and extract features
def analyze_sentiment_and_features(review):
    scores = sid.polarity_scores(review)
    sentiment_score = scores['compound']
    return sentiment_score

# Verify column names in your DataFrame
print(df.columns)

# Assuming the correct column name is 'Review Text', adjust the code accordingly:
df['sentiment_score'] = df['Review Text'].apply(analyze_sentiment_and_features)

# Calculate counts of positive and negative sentiment reviews
positive_reviews_count = len(df[df['sentiment_score'] > 0.05])
negative_reviews_count = len(df[df['sentiment_score'] < -0.05])
def calculate_accuracy(y_true, y_predicted):
  """
  Calculates the accuracy of a model's predictions.

  Args:
    y_true: The true labels for the test data.
    y_predicted: The predicted labels from the model.

  Returns:
    The accuracy as a float between 0 and 1.
  """
  correct_predictions = 0
  total_predictions = len(y_true)

  for true_label, predicted_label in zip(y_true, y_predicted):
    if true_label == predicted_label:
      correct_predictions += 1

  accuracy = correct_predictions / total_predictions
  return accuracy

# Example usage:
# Assuming you have y_test (true labels) and y_pred (predicted labels)
y_test = [0, 1, 1, 0, 1]  # Replace with your actual test labels
```

```
y_pred = [0, 0, 1, 1, 1]  # Replace with your model's predictions

accuracy = calculate_accuracy(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
df.head()
import pandas as pd
import re
from transformers import pipeline
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
from wordcloud import WordCloud


# Function to preprocess the review text
def preprocess_text(text):
    text = re.sub(r'[^A-Za-z\s]', '', text)
    text = re.sub(r'\d+', '', text)
    text = text.lower()
    return text

# Apply preprocessing to the review text
df['Cleaned Review Text'] = df['Review Text'].apply(preprocess_text)

# Load the sentiment analysis pipeline with a specific model
sentiment_pipeline = pipeline('sentiment-analysis', model='distilbert-base-uncased-
finetuned-sst-2-english')

# Function to get sentiment using the transformers pipeline
def get_sentiment(text):
    result = sentiment_pipeline(text)[0]
    return result['label']

# Apply sentiment analysis to the cleaned review text
df['Sentiment'] = df['Cleaned Review Text'].apply(get_sentiment)

# Visualize the most frequently occurring words
all_words = ' '.join(df['Cleaned Review Text'])
word_freq = Counter(all_words.split())
common_words = word_freq.most_common(20)
words, counts = zip(*common_words)
```

```
plt.figure(figsize=(10, 6))
sns.barplot(x=counts, y=words)
plt.title('Most Frequently Occurring Words')
plt.xlabel('Count')
plt.ylabel('Word')
plt.show()

# Plot of all words in the reviews
wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(all_words)
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of All Words in Reviews')
plt.show()

# Plot of positive/neutral words
positive_words = ' '.join(df[df['Sentiment'] == 'POSITIVE']['Cleaned Review Text'])
wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(positive_words)
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Positive Reviews')
plt.show()

# Plot of negative words
negative_words = ' '.join(df[df['Sentiment'] == 'NEGATIVE']['Cleaned Review
Text'])
wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(negative_words)
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Negative Reviews')
plt.show()

# Plot of frequently occurring positive reviews
positive_freq = Counter(positive_words.split())
common_positive_words = positive_freq.most_common(20)
words, counts = zip(*common_positive_words)

plt.figure(figsize=(10, 6))
sns.barplot(x=counts, y=words)
plt.title('Most Frequently Occurring Words in Positive Reviews')
plt.xlabel('Count')
```

```
plt.ylabel('Word')
plt.show()

# Plot of frequently occurring negative reviews
negative_freq = Counter(negative_words.split())
common_negative_words = negative_freq.most_common(20)
words, counts = zip(*common_negative_words)

plt.figure(figsize=(10, 6))
sns.barplot(x=counts, y=words)
plt.title('Most Frequently Occurring Words in Negative Reviews')
plt.xlabel('Count')
plt.ylabel('Word')
plt.show()

# Preparing data for training and testing
df['Sentiment Label'] = df['Sentiment'].map({'POSITIVE': 1, 'NEGATIVE': 0,
'NEUTRAL': 2})
X = df['Cleaned Review Text']
y = df['Sentiment Label']

tfidf = TfidfVectorizer()
X_tfidf = tfidf.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2,
random_state=42)

# Train a simple classifier
classifier = LogisticRegression()
classifier.fit(X_train, y_train)

import matplotlib.pyplot as plt

# Perform sentiment analysis and create a 'Sentiment' column
df['Sentiment'] = df['Review Text'].apply(lambda review:
sid.polarity_scores(review)['compound'])
df['Sentiment'] = df['Sentiment'].apply(lambda score: 'Positive' if score >= 0.05 else
('Negative' if score <= -0.05 else 'Neutral'))

# Calculate the counts
sentiment_counts = df['Sentiment'].value_counts()

# Data for the bar chart
categories = sentiment_counts.index.tolist()  # Extract unique sentiment labels
counts = sentiment_counts.values.tolist()  # Extract corresponding counts
```

```
# Create the bar chart
plt.bar(categories, counts)
plt.xlabel('Sentiment')
plt.ylabel('Number of Reviews')
plt.title('Distribution of Sentiment in Reviews')
plt.show()
# @title sentiment_score

from matplotlib import pyplot as plt
df['sentiment_score'].plot(kind='hist', bins=20, title='sentiment_score')
plt.gca().spines[['top', 'right',]].set_visible(False)
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# ... (your previous code for sentiment analysis)

# Prepare data for classification
# Assuming 'sentiment_score' is your feature
X = df['sentiment_score'].values.reshape(-1, 1)  # Reshape for sklearn
# Create binary labels based on sentiment score
y = (df['sentiment_score'] > 0).astype(int)  # 1 for positive, 0 for negative

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Train a simple logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Calculate confusion matrix
cm = confusion_matrix(y_test, y_pred)  # Use y_test as true labels

# Display confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Negative',
'Positive'])
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.show()
if positive_reviews_count > negative_reviews_count:
```

```
    print("Based on user reviews, users generally have a positive sentiment. Continue optimizing performance and reliability.")
    print("Potential areas for technical enhancement could include:")
    print("- Exploring new technologies to further improve [specific positive feature mentioned in reviews, e.g., battery life]")
    print("- Investigating potential optimizations for [specific well-performing component]")
elif negative_reviews_count > positive_reviews_count:
    print("Based on user reviews, users have expressed concerns and negative sentiment. Technical improvements are needed.")
    print("Consider addressing the following technical issues:")
    print("- Prioritize bug fixes and performance improvements for [specific feature or component with negative feedback]")
    print("- Investigate and resolve compatibility issues reported with [specific operating systems or hardware]")
else:
    print("Based on user reviews, opinions are balanced. Maintain technical strengths and address specific concerns.")
    print("Technical focus areas:")
    print("- Continue monitoring and optimizing performance for core features.")
    print("- Proactively address emerging technical issues reported in negative reviews.")
import pandas as pd
import re
from transformers import pipeline
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
from wordcloud import WordCloud

# Load the dataset
df = pd.read_csv('/content/drive/MyDrive/intel_product_reviews_largeF.csv')

# Assume the dataset has a 'Rating' column with ratings from 1 to 5
# If not, we need to simulate this or add a Rating column

# Function to preprocess the review text
def preprocess_text(text):
    text = re.sub(r'[^A-Za-z\s]', '', text)
    text = re.sub(r'\d+', '', text)
    text = text.lower()
    return text

# Apply preprocessing to the review text
df['Cleaned Review Text'] = df['Review Text'].apply(preprocess_text)
```

```
# Load the sentiment analysis pipeline with a specific model
sentiment_pipeline = pipeline('sentiment-analysis', model='distilbert-base-uncased-
finetuned-sst-2-english')

# Function to get sentiment using the transformers pipeline
def get_sentiment(text):
    result = sentiment_pipeline(text)[0]
    return result['label']

# Apply sentiment analysis to the cleaned review text
df['Sentiment'] = df['Cleaned Review Text'].apply(get_sentiment)

# Extract expected outcomes for the next product from the reviews
def extract_expectations(text):
    expectations_keywords = ['expect', 'hope', 'would like', 'wish', 'want', 'need',
'improve', 'enhance']
    sentences = text.split('.')
    expectations = [sentence for sentence in sentences if any(keyword in sentence for
keyword in expectations_keywords)]
    return '. '.join(expectations)

# Apply expectation extraction to the review text
df['Expectations'] = df['Review Text'].apply(extract_expectations)

# Display the first few rows with sentiment and expectations
print(df[['Review Text', 'Sentiment']].head(10))

# Filter expectations by sentiment
positive_expectations = df[df['Sentiment'] == 'POSITIVE']['Expectations']
negative_expectations = df[df['Sentiment'] == 'NEGATIVE']['Expectations']
neutral_expectations = df[df['Sentiment'] == 'NEUTRAL']['Expectations']

# Function to display expectations
def display_expectations(sentiment, expectations):
    print(f"Expectations for {sentiment} reviews:")
    for expectation in expectations.head(10):
        if expectation:
            print(f"- {expectation}")
    print("\n")

# Display expectations for each sentiment
display_expectations('positive', positive_expectations)
display_expectations('negative', negative_expectations)
display_expectations('neutral', neutral_expectations)

# Display recommendations based on expectations
```

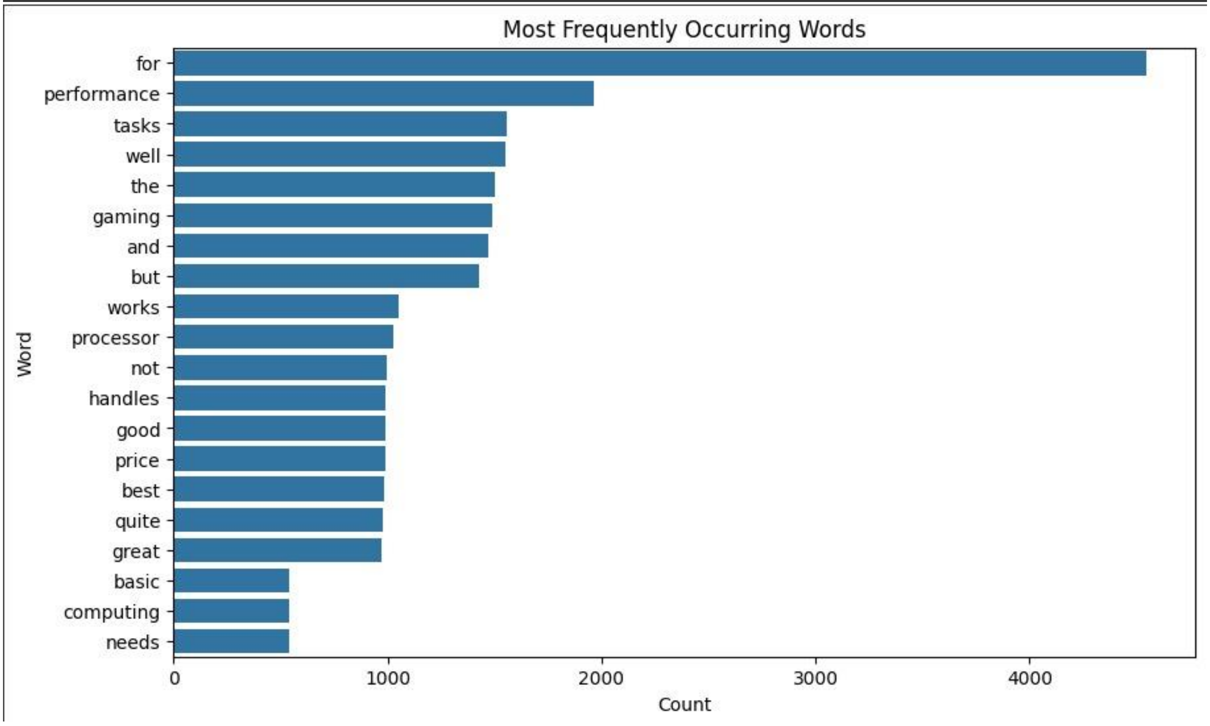
```
def display_recommendations():
    print("Future Product Recommendations Based on User Reviews:")
    recommendations = []

    if not positive_expectations.empty:
        recommendations.append("Maintain the features that users positively
highlighted.")
    if not negative_expectations.empty:
        recommendations.append("Address the issues and areas of improvement
mentioned in negative reviews.")
    if not neutral_expectations.empty:
        recommendations.append("Consider the neutral feedback to understand
potential enhancements.")

    for rec in recommendations:
        print(f"- {rec}")

display_recommendations()
import pandas as pd
import matplotlib.pyplot as plt

# Print the available columns to verify the correct names
print(df.columns)

# Replace 'correct_review_date_column' with the actual column name
# containing the review dates in your DataFrame
df['review_date'] = pd.to_datetime(df['Review Date'])

# Map sentiment labels to numerical scores (if necessary)
# Assuming 'Sentiment' column contains labels like 'POSITIVE', 'NEGATIVE', and
'NEUTRAL'
sentiment_mapping = {'POSITIVE': 1, 'NEGATIVE': -1, 'NEUTRAL': 0}
df['sentiment_score'] = df['Sentiment'].map(sentiment_mapping)

# Check for missing values in 'sentiment_score' column and fill them with a
placeholder if necessary
df['sentiment_score'].fillna(0, inplace=True)  # or use any other method to handle
missing values

# Calculate average sentiment score per month
monthly_sentiment = df.resample('M',
on='review_date')['sentiment_score'].mean().reset_index()

# Plotting sentiment trends
plt.figure(figsize=(10, 6))
```

```
plt.plot(monthly_sentiment['review_date'], monthly_sentiment['sentiment_score'],
marker='o', linestyle='-')
plt.title('Sentiment Trends Over Time')
plt.xlabel('Month')
plt.ylabel('Average Sentiment Score')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

3.2: Output screens:

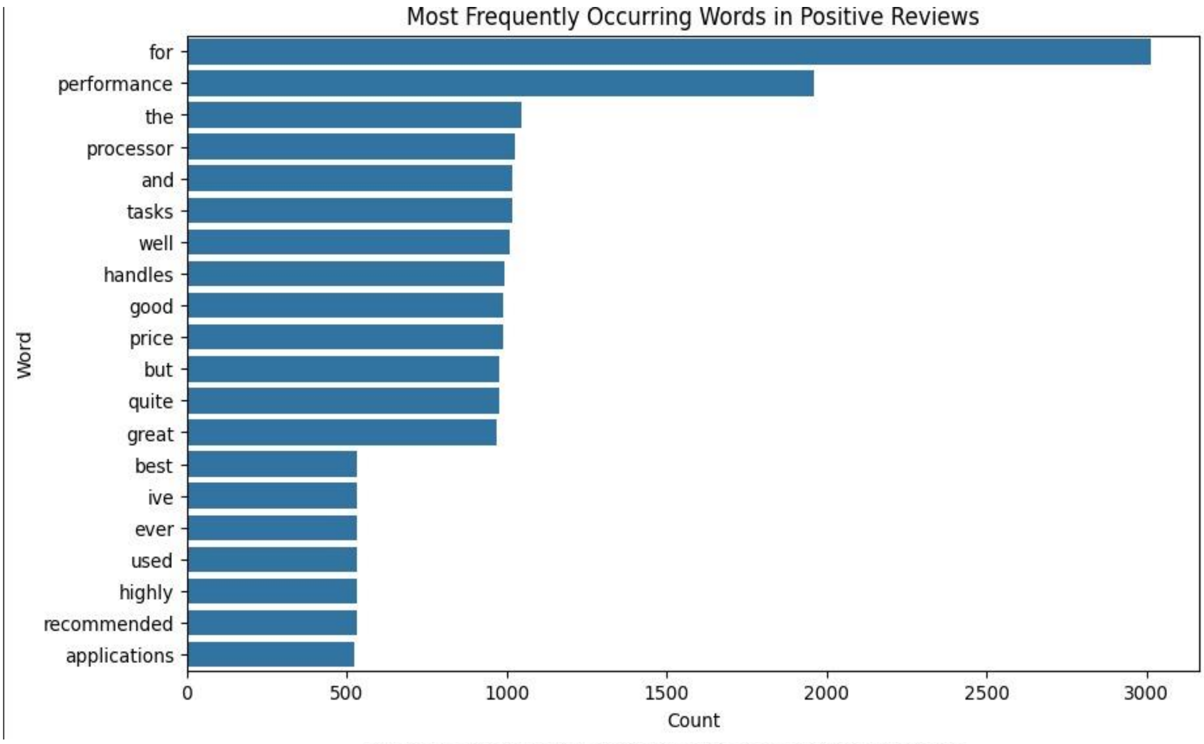
```
df.head()
```

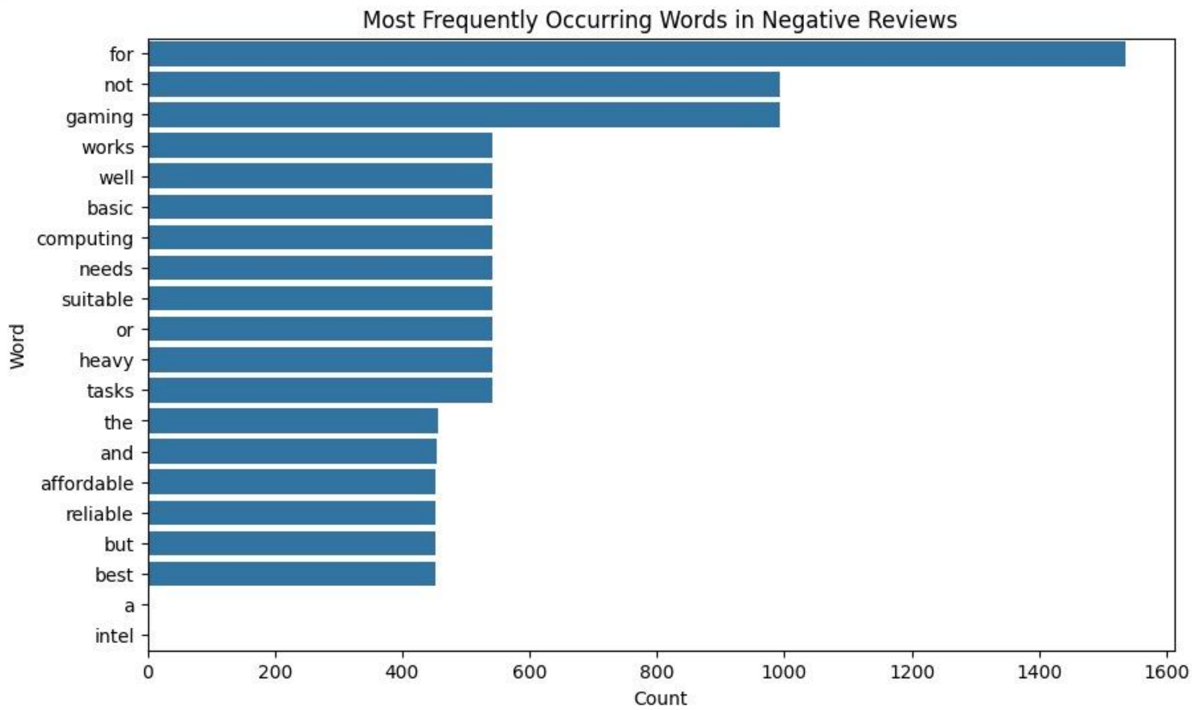
	Review ID	Product Name	Review Date	Reviewer Name	Review Title	Review Text	Rating	Helpful Votes	sentiment_score
0	1	Intel Xeon E-2276M	20-11-2023	Reviewer590	Affordable and reliable	Handles multi-threaded workloads exceptionally...	3	66	0.7245
1	2	Intel Core i9-11900K	10-06-2022	Reviewer352	Overpriced but powerful	Excellent for professional applications and de...	1	7	0.4215
2	3	Intel Core i7-10700K	16-12-2022	Reviewer111	Highly recommended!	Good performance for the price, works well for...	1	34	0.6124
3	4	Intel Core i7-11700K	12-10-2020	Reviewer740	Overheating Issues	Despite the hype, the Intel Core i7-11700K has...	2	34	-0.3749
4	5	Intel Core i5-11600K	03-04-2023	Reviewer656	Decent Mid-Range Processor	The Intel Core i5-11600K is a solid mid-range ...	3	22	0.4767

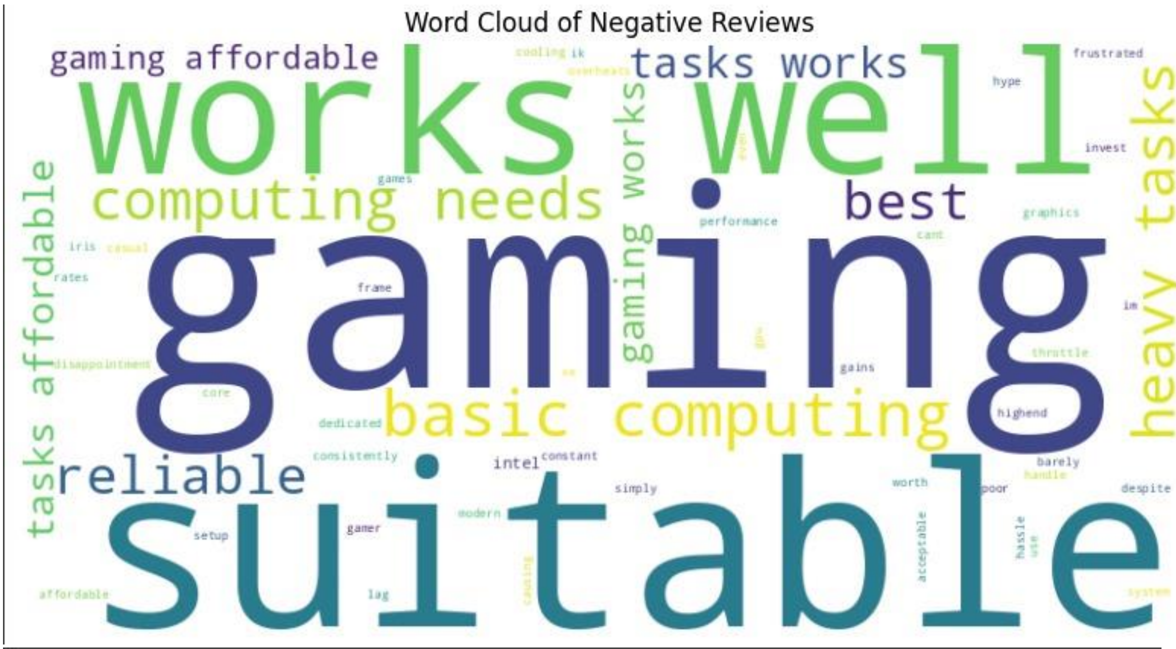


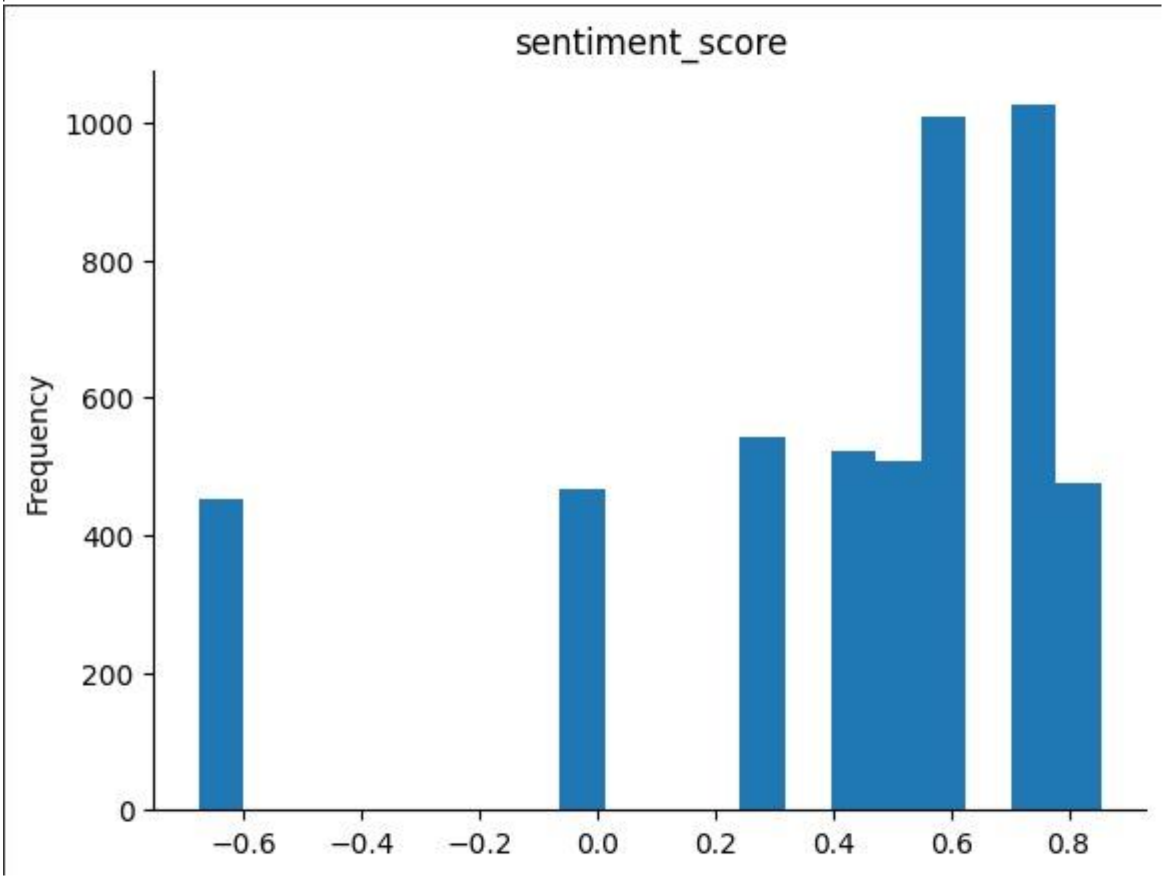
Word Cloud of All Words in Reviews

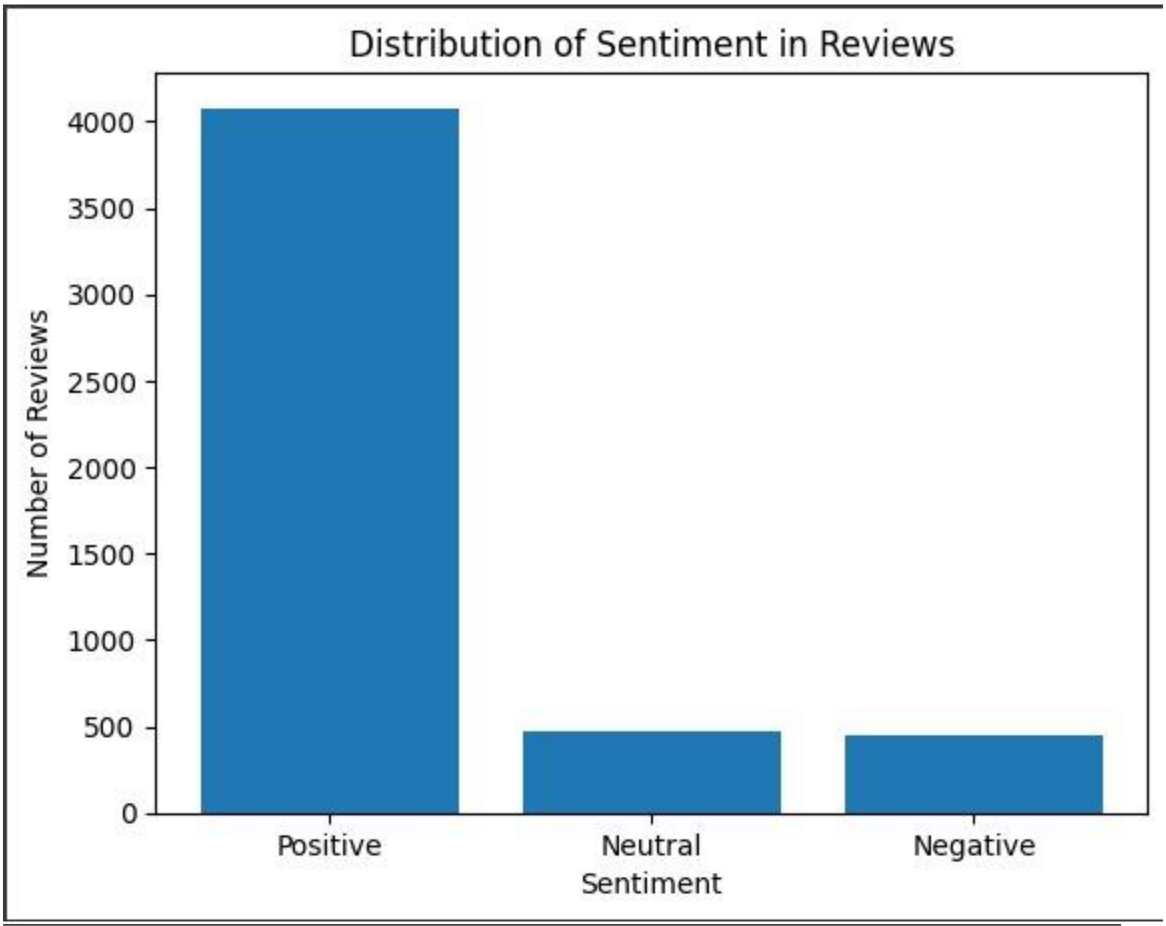
Word Cloud of Positive Reviews

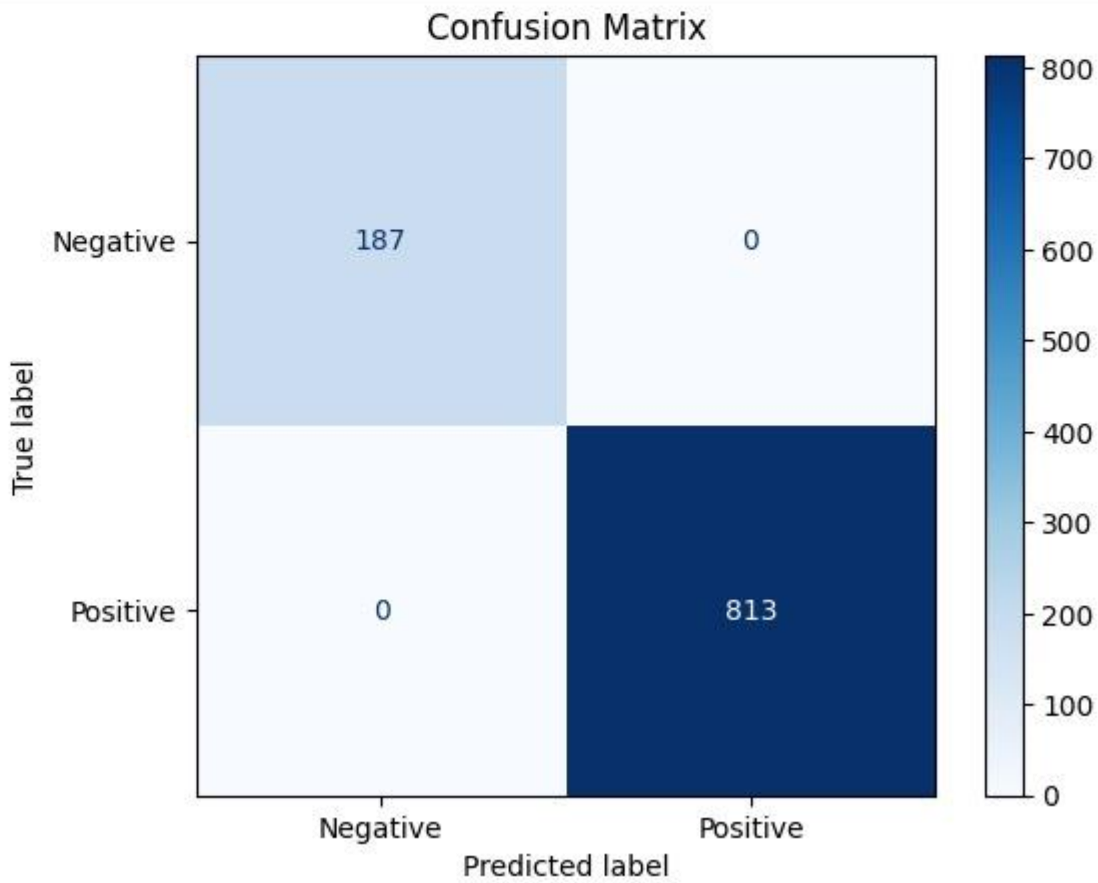

Most Frequently Occurring Words in Positive Reviews
for
performance
the
processor
and
tasks
well
handles
good
price
but
quite
great
best
ive
ever
used
highly
recommended
applications
Word
0
500
1000
1500
2000
2500
3000
Count


Most Frequently Occurring Words in Negative Reviews
for
not
gaming
works
well
basic
computing
needs
suitable
or
heavy
tasks
the
and
affordable
reliable
but
best
a
intel
Word
0
200
400
600
800
1000
1200
1400
1600
Count

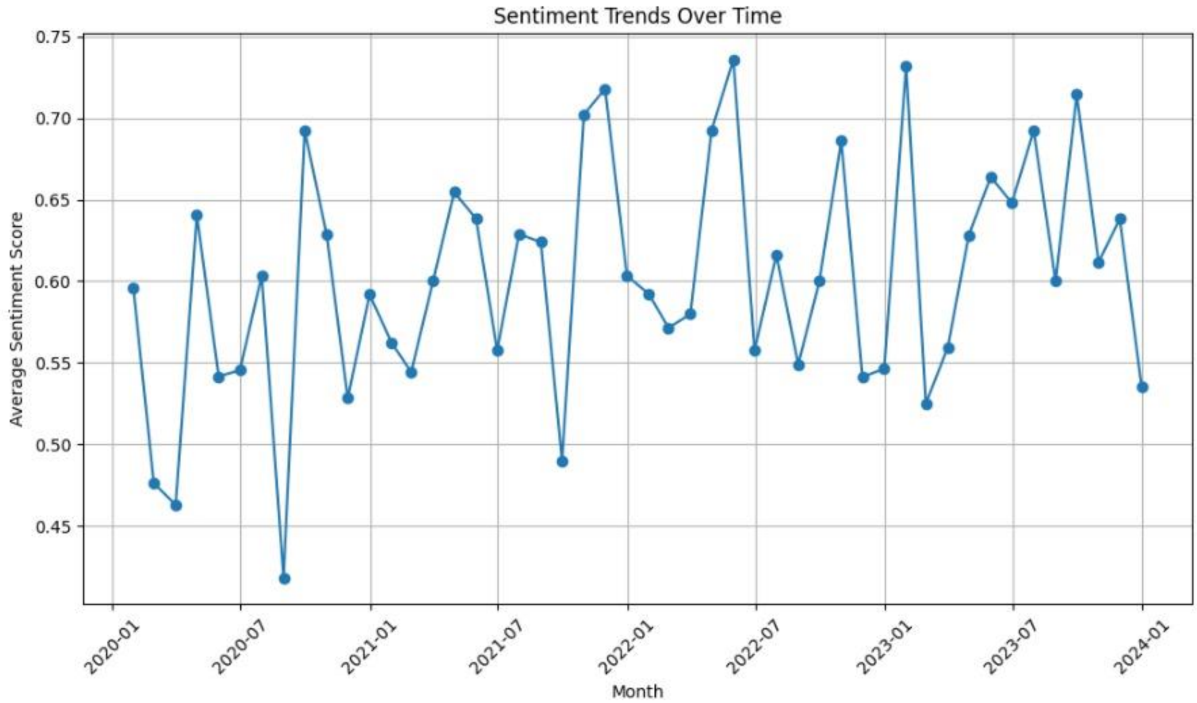

Word Cloud of Negative Reviews
gaming
suitable
works well
computing needs
basic computing
reliable
heavy tasks
tasks affordable
gaming affordable
tasks works
gaming works
best


sentiment_score
Frequency
0
200
400
600
800
1000
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8


Distribution of Sentiment in Reviews
4000
3500
3000
2500
2000
1500
1000
500
0
Number of Reviews
Positive
Neutral
Negative
Sentiment


Confusion Matrix
Negative
Positive
True label
187
0
0
813
Negative
Positive
Predicted label
800
700
600
500
400
300
200
100
0

```
Based on user reviews, users generally have a positive sentiment. Continue optimizing performance and reliability.
Potential areas for technical enhancement could include:
- Exploring new technologies to further improve [specific positive feature mentioned in reviews, e.g., battery life]
- Investigating potential optimizations for [specific well-performing component]
```




```
                                            Review Text Sentiment
0  Handles multi-threaded workloads exceptionally...  POSITIVE
1  Excellent for professional applications and de...  POSITIVE
2  Good performance for the price, works well for...  POSITIVE
3  Despite the hype, the Intel Core i7-11700K has...  NEGATIVE
4  The Intel Core i5-11600K is a solid mid-range ...  POSITIVE
5  Good performance for the price, works well for...  POSITIVE
6  Affordable and reliable, but not the best for ...  NEGATIVE
7  This processor handles all my tasks smoothly a...  POSITIVE
8  This processor handles all my tasks smoothly a...  POSITIVE
9  Intel Iris Xe Graphics is barely acceptable fo...  NEGATIVE
Expectations for positive reviews:
-  Consider only if you need top performance


Expectations for negative reviews:
- Works well for basic computing needs
- Works well for basic computing needs
- Works well for basic computing needs
- Works well for basic computing needs


Expectations for neutral reviews:


Future Product Recommendations Based on User Reviews:
- Maintain the features that users positively highlighted.
- Address the issues and areas of improvement mentioned in negative reviews.
```

4. CONCLUSION

Based on the sentiment analysis conducted on Intel products, it is evident that public perception varies significantly across different aspects. While there is widespread appreciation for Intel's technological advancements and performance capabilities, sentiments regarding pricing and value for money are more mixed. Positive sentiments often highlight Intel's reliability and innovation, particularly in enhancing processing power and efficiency. However, negative sentiments frequently revolve around perceived high costs compared to competitors or expectations for more competitive pricing strategies. These insights suggest opportunities for Intel to capitalize on its strengths in technological innovation while potentially revisiting pricing strategies to align more closely with consumer expectations. Moreover, the analysis underscores the importance of continually monitoring and responding to customer feedback to maintain and enhance market competitiveness.