University of Pisa

Artificial Intelligence and Data Engineering
Business and Project Management

Innovative Product Feedback Analysis: Leveraging AI for Amazon *Review Summarization*

Ricky Marinsalda
Vittoria Acampora

September 11, 2023

Contents

Chapter 1

Introduction

In the ever-expanding digital marketplace, product reviews have become a crucial source of information for both consumers and manufacturers. These reviews often serve as a window into the quality and utility of products, enabling potential buyers to make informed decisions and providing valuable feedback to product producers. However, the sheer volume of reviews available on platforms like Amazon can be overwhelming, making it challenging for consumers to extract meaningful insights and for manufacturers to gather actionable feedback efficiently.

This project revolves around the analysis and synthesis of user reviews within a specific category of Amazon products. We selected the "Industrial and Scientific products" category, chosen at random, to simplify the process due to the vast number of reviews available in the broader dataset. Our objective was to process these reviews and harness the capabilities of the ChatGPT API, configuring it with suitable parameters to generate concise summaries for each product.

1.1 Research Question

The central research question driving this project is as follows:

Can the use of AI-driven text summarization, such as ChatGPT, effectively distill the essence of user reviews for Amazon products in a manner that is valuable to both consumers and manufacturers?

1.2 Goal of the Project

The primary goal of this project is to create informative and concise summaries of user reviews for Amazon products, with dual objectives in mind:

1. **Consumer Benefit:** Provide consumers with easily digestible insights about a product, allowing them to form an impression without the need to read through numerous reviews. This can significantly enhance the shopping experience by saving time and effort.

2. **Manufacturer Insight:** Offer manufacturers a means to gauge consumer sentiment and identify areas for product improvement. These summaries serve as a condensed representation of customer opinions, enabling manufacturers to make informed decisions regarding product enhancements.

To assess the effectiveness of AI-generated summaries, we employed various evaluation methods, including sentiment analysis, comparisons between AI-generated and human-authored summaries, and human evaluations of the generated summaries. Furthermore, we implemented a rudimentary prototype application, featuring a graphical user interface developed in Python, to demonstrate the practical application of our project.

This report will detail our methodology, findings, and conclusions regarding the utility and effectiveness of AI-driven review summarization in the context of Amazon product reviews.

Chapter 2

Process

2.1 Dataset Acquisition and Preparation

In this section, we describe the process of acquiring and preparing the dataset for our project. The dataset used in this study was obtained from the following source:

Source: Amazon Product Review Data

We specifically chose the category "Industrial and Scientific" reviews, which comprised a total of 1,758,333 reviews.

2.1.1 Dataset Format Adjustment

The dataset was initially provided in JSON format. However, it was not vectorized and required some adjustments to make it suitable for our analysis. To address this, we implemented the following Python code to fix the dataset:

```
def fix_json_file(file_path):
with open(file_path, 'r') as json_file:
data = json_file.read()

# Remove any extra white spaces
data = data.strip()

# Add commas between objects
data = data.replace('}\n{', '},\n{')

# Add square brackets to create an array
fixed_data = '[' + data + ']'

# Overwrite the file with corrected data
with open(file_path, 'w') as json_file:
json_file.write(fixed_data)

# Execute the correction
if __name__ == "__main__":
json_file_path = "Industrial_and_Scientific_adj.json"
fix_json_file(json_file_path)
```

The code above takes the original JSON file, removes extra white spaces, adds commas between objects, and encapsulates the entire dataset within square brackets to create a valid JSON array.

2.1.2 Loading the Adjusted Dataset

Once the dataset was formatted correctly, we were able to load it into our main program using the following code:

```
# Load the JSON file containing the reviews
def load_reviews_from_json(file_path):
with open(file_path, 'r') as json_file:
data = json.load(json_file)
return data
```

This code allowed us to easily manipulate and work with the dataset for our analysis and research purposes.

In the next sections, we will delve into the methods and techniques used to analyze and summarize the reviews within this dataset.

2.2 Generating Product Review Summaries with ChatGPT

In this section, we will discuss the crucial aspect of our project, which involves utilizing Python code to interact with the ChatGPT API (specifically, the gpt-3.5-turbo model). Our objective is to create weighted summaries of user reviews for the top 20 products within the "Industrial and Scientific" category on Amazon. Each summary will be composed by amalgamating the reviews and considering the associated user ratings, where ratings of 1 and 2 indicate negative feedback, and ratings of 4 and 5 indicate positive feedback.

2.2.1 API Configuration

To begin, we set up the API configuration by specifying our OpenAI API key. This key allows us access to the ChatGPT model for generating summaries.

```
import openai

# Set your OpenAI API key
openai.api_key = "YOUR_API_KEY"
```

2.2.2 Generating Weighted Summaries

We have implemented a Python function, ‘generate_summary(reviews)‘, to generate weighted summaries of the reviews. This function takes a list of reviews as input and combines them into a single text string. The weighting is based on user ratings, and the function calculates the total weight for reference.

```
def generate_summary(reviews):
weighted_summary = ""
total_weight = 0

for review in reviews:
rating = review.get("rating", "N/A")
review_text = review.get("text", "N/A")

weighted_summary += f"{review_text} [Rating: {rating}] "
total_weight += rating

return weighted_summary, total_weight
```

2.2.3 Preparing the Review Data

We read the reviews from a TXT file, where each line contains a review text and its associated rating, separated by a tab. We load these reviews into a Python list for processing.

```
# Read reviews from the TXT file
reviews = []

with open("review/reviews.txt", "r") as file:
for line in file:
parts = line.strip().split("\t")
if len(parts) == 2:
review = {"text": parts[0], "rating": float(parts[1])}
reviews.append(review)
```

2.2.4 Generating Weighted Summaries

Next, we generate the weighted summary text by calling the 'generate_summary()' function. We then create a prompt that instructs ChatGPT to generate a clear and informative summary of the reviews, taking into account the weighted ratings. The generated summary is saved to a TXT file.

```
if len(reviews) > 0:
weighted_summary, total_weight = generate_summary(reviews)

prompt = "Write␣an␣English␣summary␣that␣encapsulates␣the␣sentiment␣of␣all␣the
    ↪ ␣reviews␣provided␣in␣the␣input,␣considering" \
"the␣weight␣given␣by␣the␣numeric␣values␣from␣1␣to␣5.␣Where␣1␣and␣2␣indicate␣
    ↪ negative␣sentiment,␣4␣and␣5␣indicate␣positive␣sentiment." \
"The␣summary␣should␣be␣clear,␣even␣if␣it␣involves␣rephrasing␣sentences␣where␣
    ↪ necessary." \
"This␣summary␣should␣be␣useful␣to␣a␣potential␣user␣who␣does␣not␣want␣to␣read␣
    ↪ all␣the␣reviews" \
"but␣wants␣to␣get␣an␣idea␣of␣what␣other␣users␣think␣about␣the␣product." \
"Reviews:\n\n" + weighted_summary

if total_weight > 0:
weighted_summary = weighted_summary.replace("\n", "␣")
generated_summary = openai.ChatCompletion.create(
model="gpt-3.5-turbo",
messages=[
{"role": "system", "content": "You␣are␣a␣helpful␣assistant."},
{"role": "user", "content": prompt}
],
temperature=1.2,
max_tokens=600
)
summary_text = generated_summary.choices[0].message.content

with open("summary.txt", "w") as output_file:
output_file.write(summary_text)
print("**Summary␣generated**")
else:
print("No␣reviews␣available␣to␣generate␣a␣summary.")
```

This code demonstrates the process of generating weighted summaries for product reviews using ChatGPT and saving the results to a text file. It allows users to quickly grasp the sentiment and feedback of other users without having to read all the individual reviews.

2.3 Sentiment Analysis and Visualization

In this paragraph, a comprehensive analysis of user sentiments expressed in product reviews and the generated summary will be presented. Additionally, results will be visually represented through various graphs.

2.3.1 Methodology of Analysis

2.3.2 Using VADER

To assess sentiments in both user reviews and the summary, we employed the use of VADER (Valence Aware Dictionary and sEntiment Reasoner). VADER is a widely-used dictionary-based tool for sentiment analysis in text. It assigns sentiment scores, including positivity, neutrality, and negativity, to the analyzed text.

Compound Score

The compound score is computed by summing the valence scores of each word in the lexicon, adjusted according to specific rules, and then normalized to be between -1 (most extreme negative) and +1 (most extreme positive). This is the most useful metric if you want a single unidimensional measure of sentiment for a given sentence. Calling it a 'normalized, weighted composite score' is accurate.

It is also useful for researchers who would like to set standardized thresholds for classifying sentences as either positive, neutral, or negative. Typical threshold values (used in the literature) are:

- Positive Sentiment: Compound score ≥ 0.5
- Neutral Sentiment: $(-0.5 <$ Compound score $< 0.5)$
- Negative Sentiment: Compound score ≤ -0.5

Pos, Neu, and Neg Scores

The *pos*, *neu*, and *neg* scores are ratios for proportions of text that fall in each category (so these should all add up to be 1, or close to it with float operations). These are the most useful metrics if you want multidimensional measures of sentiment for a given sentence.

Beyond the compound score, VADER provides a more nuanced understanding of sentiment by breaking it down into positive, neutral, and negative components. These scores enable a detailed analysis of the sentiment composition within a given text, making VADER a valuable tool for understanding the complex nature of sentiments expressed in user reviews, summaries, or any textual content.

2.3.3 Data Preparation

Initially, we read product reviews from a text file, extracting both the review text and the numeric ratings provided by users. Furthermore, we extracted the summary from the dedicated text file.

2.3.4 Sentiment Analysis

2.3.5 Sentiment Scores with VADER

We applied VADER sentiment analysis to each review and the summary, obtaining sentiment scores for each piece of text. The initial VADER scores were subsequently adjusted to account for the numeric user ratings. This correction allowed for a more accurate representation of user-expressed sentiments.

2.3.6 Comparative Analysis

We categorized reviews based on their numeric ratings, dividing them into low, medium, and high rating groups. Subsequently, we analyzed the sentiment scores associated with each category. This allowed us to compare user-expressed sentiments with corrected VADER scores.

2.3.7 Results Visualization

To visually represent the analysis results, we created a series of graphs for one product (taken as an example):

- **Comparison of Original and Corrected VADER Scores**: This graph illustrates the comparison between the original and corrected VADER scores for each review.

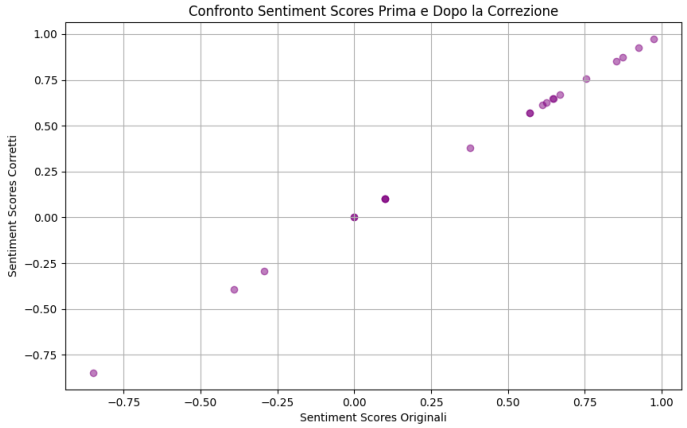


Figure 2.1

- **Relationship Between User Numeric Ratings and VADER Scores**: This graph showcases the relationship between user-assigned numeric ratings and corresponding VADER scores (not adjusted).

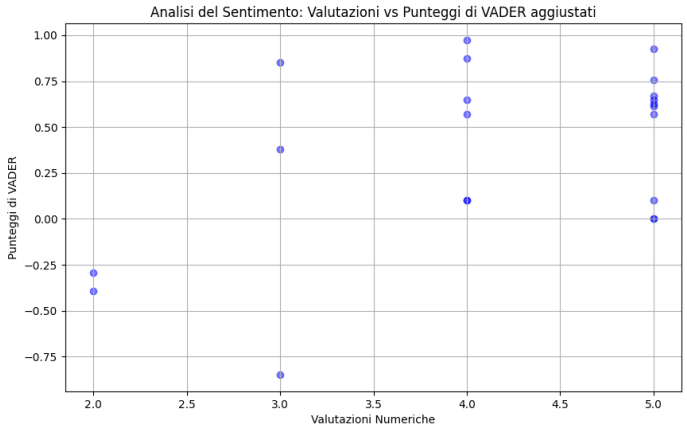


Figure 2.2

- **Radar Chart Analysis of Sentiment Scores**: This radar chart offers a multi-attribute analysis of sentiment scores for different rating groups. The orange dashed line on the graph corresponds to the scores assigned to the summary.

  Radar charts serve as a valuable tool for simultaneously comparing sentiment scores across various categories, with each axis representing a specific sentiment category, such as positive, negative, or neutral, facilitating quick visual assessments of sentiment variations, while also aiding in the identification of outliers or extreme values that may indicate unusual patterns warranting further investigation.

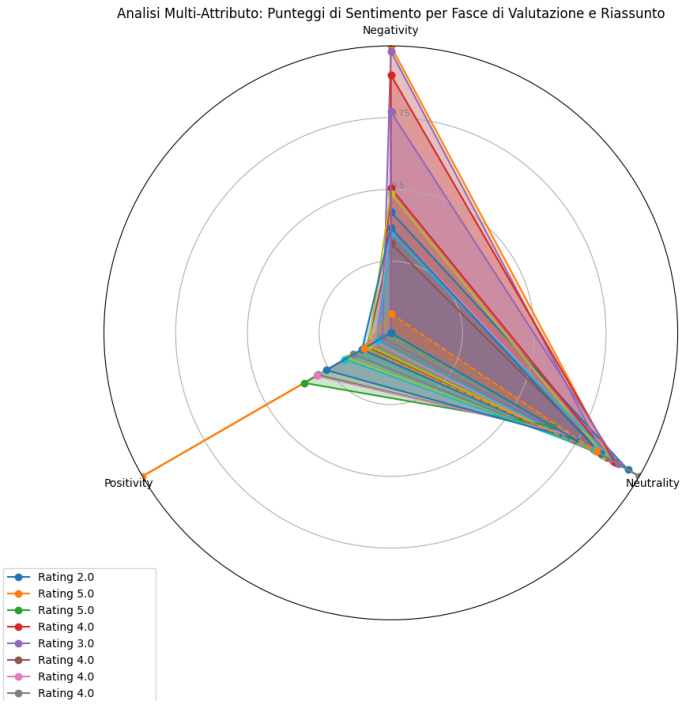


Figure 2.3

- **Comparison of Summary and Review Sentiment Scores**: This graph compares the sentiment scores of the summary with those of the reviews, providing a comprehensive overview of expressed sentiments.

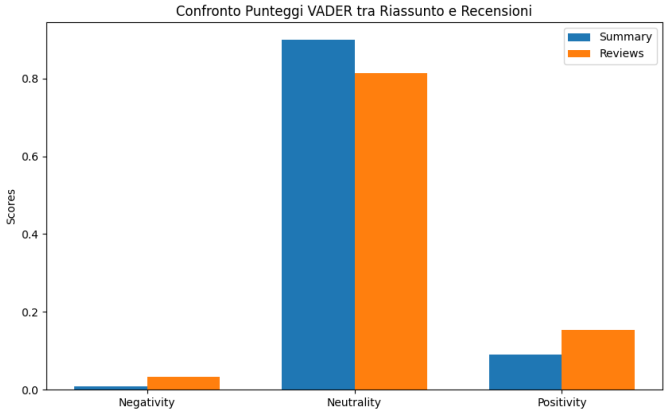


Figure 2.4

These graphs effectively allowed us to visually represent the sentiment analysis results, providing an in-depth understanding of user opinions and emerging trends.

2.3.8 Final Sentiment Analysis Evaluation

The graphs presented throughout this analysis pertain to a specific product; however, the same graphs can be generated for any other product input. Nonetheless, we have observed that the sentiments expressed in the summaries closely resemble those found in the reviews. This demonstrates

that the summary effectively captures the average general sentiment expressed in the reviews.

Additionally, we have made adjustments to the VADER score. It was noticed that, for certain ratings, the sentiment expressed in the review conflicted with the sentiment implied by the star rating provided alongside the review. To address this, we deemed it necessary to adjust the sentiment score. The adjustment is demonstrated in the following code snippet:

```
def adjust_sentiment_score(sentiment_scores, rating):
# Apply rules for adjusting VADER scores
if rating >= 4 and sentiment_scores['compound'] < 0:
sentiment_scores['compound'] = max(sentiment_scores['compound'], 0.1)
elif rating <= 2 and sentiment_scores['compound'] > 0:
sentiment_scores['compound'] = min(sentiment_scores['compound'], -0.1)
return sentiment_scores
```

Furthermore, it's worth elaborating on the following:

2.3.9 Influence of Customer Expectations on Product Evaluation

The evaluation of products by customers is influenced by their pre-purchase performance expectations. These expectations are, in turn, shaped by the existence of standards against which the product can be assessed. According to the theory of Customer Benefit perceived Experience (CBE), the perception of product experiences varies among customers and is mediated by performance standards.

These characteristics may elucidate a recurring phenomenon in online product reviews known as "negativity bias." Negative information tends to carry more weight in customer evaluations. Negative product information from peers exerts a more significant impact. This aligns with the adage that "bad is stronger than good." [1]

However, it's important to note that our sentiment analysis, while valuable, may not comprehensively capture this aspect of customer evaluations. Our analysis focuses on sentiment expressions within reviews and summaries and may not fully account for the intricate relationship between pre-purchase expectations, performance standards, and the resulting bias in customer feedback. This complex interplay can involve factors beyond the scope of sentiment analysis, such as individual cognitive processes and the context of the evaluation.

2.4 Text Analysis

To perform text analysis on both summaries and reviews, we opted for a word count approach. This allowed us to analyze the most frequently occurring words within the summaries and reviews. While relatively straightforward, this method is effective in providing an overview of the most important keywords in the texts.

2.4.1 Word Count Methodology

The process involved splitting the text into tokens and subsequently removing common words, often referred to as "stop words." These stop words include articles, conjunctions such as "the," "is," and others that are not pertinent to our analysis, as they do not convey significant information about the text's themes. Following this, we recorded the frequency of the most meaningful words within the text and identified which keywords were most common.

Additionally, we generated a word cloud to visually represent the most frequent words in a clear and intuitive manner. In these word clouds, the size of the words corresponds to their frequency, with larger words indicating higher frequency.

[1] The observations made here were derived from the reading of the paper 'Technical Sentiment Analysis: Measuring Advantages and Drawbacks of New Products Using Social Media' by Filippo Chiarello, Andrea Bonaccorsi, and Gualtiero Fantoni, from the School of Engineering, Largo Lucio Lazzarino 2, 56122, Pisa, Italy.

2.4.2 Product Analysis

In our project, we analyzed four different products:

Product 1

(a) Reviews (b) Summary

Figure 2.5: Word Cloud for Product 1

It is evident from the word cloud that the most frequent words in the reviews for this product are "tool," "flooring," and "hardwood." These same keywords are also prevalent in the summary, providing insights into the product's characteristics.

Product 2

(a) Reviews (b) Summary

Figure 2.6: Word Cloud for Product 2

The word cloud for this product indicates that the most frequent words in the reviews are "amazon," "smell," and "price." These same keywords are mirrored in the summary, suggesting that it is an Amazon product and that reviewers have made comments regarding the price.

Product 3

(a) Reviews (b) Summary

Figure 2.7: Word Cloud for Product 3

For this product, the most frequent words in the reviews include "sandpaper," "adhesive," and "sticky." These keywords are also found in the summary, aiding in understanding the product's characteristics.

Product 4

(a) Reviews (b) Summary

Figure 2.8: Word Cloud for Product 4

In the case of this product, the most frequent words in the reviews are "positive," "good," and "price." These keywords are likewise reflected in the summary, suggesting that the product has received positive evaluations.

2.4.3 Comparison between API Summary and Human Summary

(a) Summary from chatgpt (b) Summary from an human

Figure 2.9: Word Cloud for Product 1

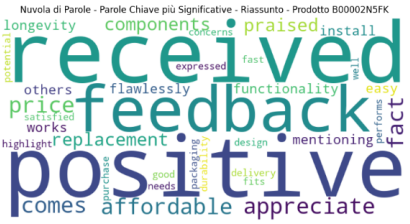
(a) Summary from chatgpt (b) Summary from an human

Figure 2.10: Word Cloud for Product 2

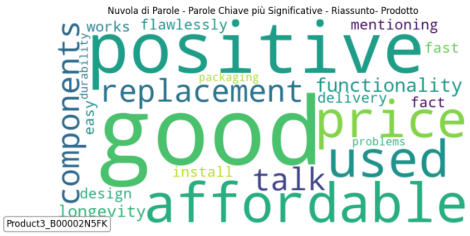
(a) Summary from chatgpt

(b) Summary from an human

Figure 2.11: Word Cloud for Product 3



(a) Summary from chatgpt



(b) Summary from an human

Figure 2.12: Word Cloud for Product 4

For all these products, it is evident that the most frequent words found in the summaries generated by the API are also present in the human-generated summaries. This serves as additional evidence that the human summary closely resembles that of the API, highlighting the same fundamental aspects.

2.5 Cosine similarity

Another approach that we can use to evaluate the feasibility and efficiency of summaries generated by API is through the cosine similarity method, which allows us to compare the summary with a human reference one and analyze how similar or dissimilar these two are.

2.5.1 How does it work?

Cosine Similarity is a metric used to measure the similarity between two data vectors in a multidimensional space. It is widely used in information retrieval, text mining, pattern recognition, and text analysis.

- **Vector Representation:** Initially, textual data is represented as numerical vectors in a vector space. This is often done using techniques such as TF-IDF (Term Frequency-Inverse Document Frequency) or Word Embeddings.
- **Calculation of similarity:** Once the texts have been converted into vectors, cosine similarity calculates the geometric similarity between these vectors. This metric measures the angle between vectors, where an angle of 0 degrees indicates complete similarity (vectors point in the same direction), and an angle of 90 degrees indicates complete dissimilarity (vectors are orthogonal).

The formula for calculating the cosine similarity between two vectors A and B is as follows:

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

Where:

- $A \cdot B$ is the dot product between vectors A and B.
- $\|A\|$ is the norm (length) of vector A.
- $\|B\|$ is the norm (length) of vector B.

2.5.2 Advantages

- **Length independent:** Cosine similarity is independent of the length of the vectors, which means that it is robust with respect to the length of the text.
- **Easy to calculate:** The cosine similarity calculation is computationally efficient and can be easily implemented.
- **Content Sensitivity:** Measures similarity based on document content, which makes it useful for finding documents with similar keywords or similar themes.

However, it is important to note that cosine similarity does not take into account the semantic context or grammatical structure of the text. Thus, it may not be suitable for all types of text analysis, especially if the meaning of words needs to be considered. In such cases, more advanced representations, such as Word Embeddings, may be preferable.

2.5.3 Results

The output of the cosine similarity is a numerical value between -1 and 1, which represents the similarity between two vectors.

- **Maximum value (1.0):** Indicates that the two vectors are identical or very similar. In other words, the two texts are very similar from the point of view of the terms considered.
- **Value close to 0 (close to 0.0):** Indicates that the two vectors are very different or uncorrelated. In this case, the two texts are very dissimilar from the point of view of the terms considered.
- **Negative value (-1.0):** This value can occur in exceptional cases and indicates total dissimilarity or opposition between the two vectors.

In the context of text comparison, it is common to use a similarity threshold (for example, 0.5) to determine whether two texts are considered similar or different. For example, if the cosine similarity between two summaries is greater than or equal to 0.5, you might decide to classify them as "similar"; otherwise, you would classify them as "different."

2.5.4 Our case

We took three products and compared the human summary with the summary generated by the API using cosine similarity.

```
Cosine Similarity tra i due riassunti per Product1: 0.55
I riassunti per Product1 sono molto simili.
Cosine Similarity tra i due riassunti per Product2: 0.50
I riassunti per Product2 sono molto simili.
Cosine Similarity tra i due riassunti per Product3: 0.62
I riassunti per Product3 sono molto simili.
```

Figure 2.13

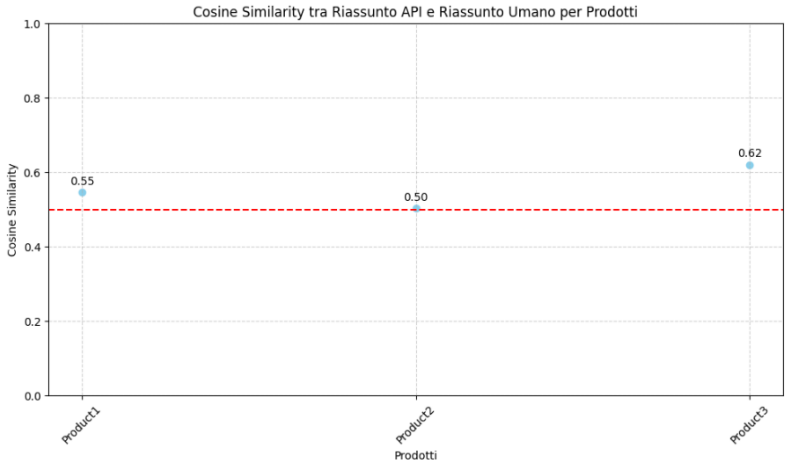


Figure 2.14: Cosine Similarity between API Summary and Human Summary for Products

- For all three products, the cosine is greater than 0.5, so we can say that the two summaries are similar. However, it is important to keep in mind that cosine similarity is a metric based on the similarity between word vectors and does not take into account the full semantic meaning of the texts. It can be useful for identifying similarity based on terms used in texts but may not capture more complex shades of meaning.

2.6 Human Evalutation

Another approach to evaluating API-generated abstracts is through human evaluation. We chose to create a questionnaire for our users, aimed at receiving feedback on the quality of the summaries generated by the API. Users are asked to provide a score ranging from 1 to 10. Specifically, we have designed three questions in which users can read product reviews and the API-generated summary, and then assign a numerical rating.

Additionally, it's important to note that the users we invited to participate in this survey were carefully selected individuals with no biases towards the products (This is also the reason why there aren't many votes). They were provided with the reviews and summaries generated. Their task was to evaluate the summaries solely based on the provided reviews, ensuring an unbiased evaluation process.

Now, let's analyze the same three products as in the cosine similarity section.

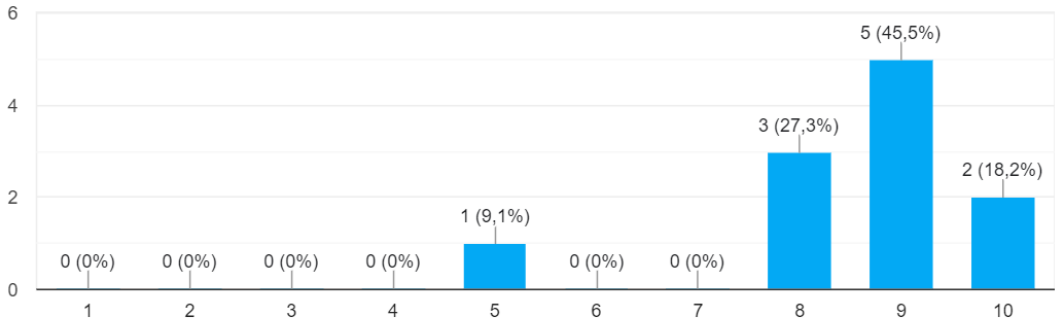


Figure 2.15: Distribution of User Ratings for API-Generated Summary for product 1

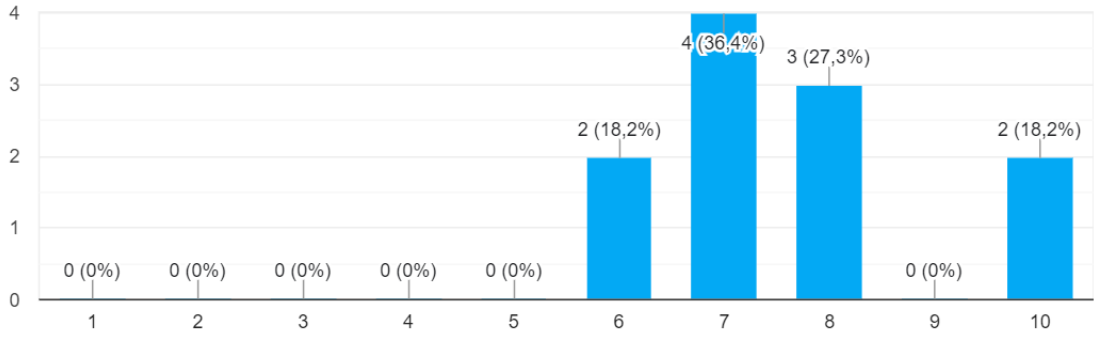


Figure 2.16: Distribution of User Ratings for API-Generated Summary for product 2

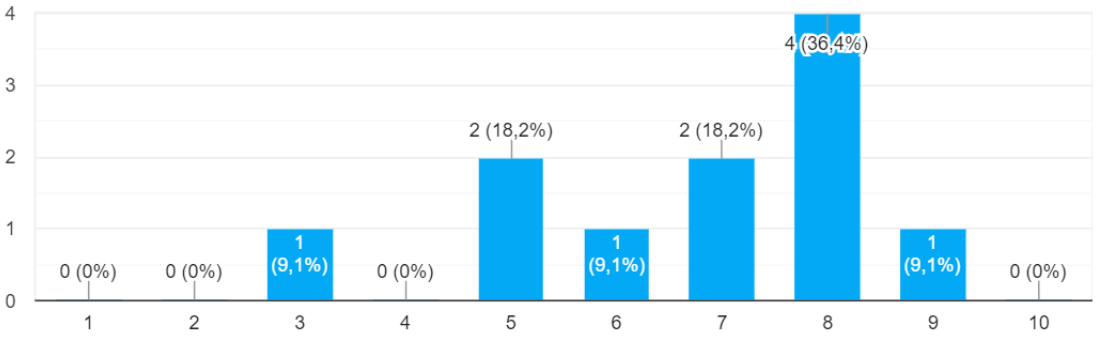


Figure 2.17: Distribution of User Ratings for API-Generated Summary for product 3

As depicted in Figure 2.15, 2.16 and 2.17, with the exception of a few outlier cases, most of the evaluations fall between 5 and 10. This suggests that the summaries generated by the API are generally effective in representing user reviews and, in many cases, can serve as viable replacements for human-generated summaries.

2.7 Graphical User Interface (GUI) and Application Prototype

In this section, we discuss the implementation of a Graphical User Interface (GUI) for our project and introduce a Python application prototype that simplifies the process of processing and summarizing product reviews. The GUI allows users to interact with the application effortlessly and visualize various aspects of the analysis, including sentiment scores and review summaries.

2.7.1 Application Overview

The Python application provides the following key features:

1. **File Selection**: Users can select a text file containing a set of product reviews. This file serves as the input for the review analysis.

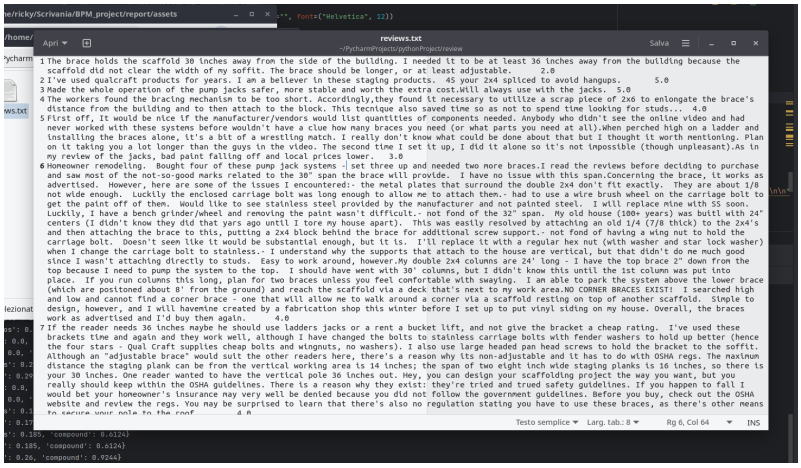


Figure 2.18: Example of a file review.txt

2. **Generate Summary**: By clicking the "Generate summary" button, users can initiate the process of generating a weighted summary of the reviews using ChatGPT. The resulting summary is displayed in the GUI for easy access.

3. **Generate Chart**: Users have the option to create a sentiment score chart by clicking the "Generate Chart" button. This chart provides insights into the sentiment of the reviews before and after correction.

4. **User-Friendly Interface**: The GUI features informative labels and clear instructions, making it user-friendly and accessible.

5. **Progress Indicator**: A progress bar is included to inform users about the status of the summary generation process.

2.7.2 Application Components

Here are some of the essential components of the Python application:

1. **Label and File Selection**: The label displays the selected file, and users can click the "Select a file" button to choose the input file for review analysis.

2. **Generate Summary Button**: Clicking this button initiates the process of generating a weighted summary of the reviews. The summary is displayed in a text widget within the GUI.

3. **Generate Chart Button**: This button allows users to create a sentiment score chart, which is displayed within the GUI.

4. **Progress Bar**: The progress bar indicates the progress of the summary generation process.

5. **Text Widget**: The text widget provides a space to display the generated review summary.

6. **Graphical Chart**: The GUI includes an area for displaying the sentiment score chart, providing visual insights into the review sentiment.

2.7.3 Usage Instructions

Here are the general steps for using the application:

1. Select a text file containing product reviews by clicking the "Select a file" button.

2. Click the "Generate summary" button to generate a weighted summary of the reviews using ChatGPT. The summary will be displayed in the text widget.

3. Optionally, click the "Generate Chart" button to create a sentiment score chart for visualizing sentiment trends.

Please note that the summary generation process may take some time, and the progress bar provides feedback on the process's status.

2.7.4 Graphical User Interface (GUI)

The GUI includes informative labels, clear instructions, and user-friendly buttons, making it accessible and intuitive for users. The window is centered on the screen for convenience.

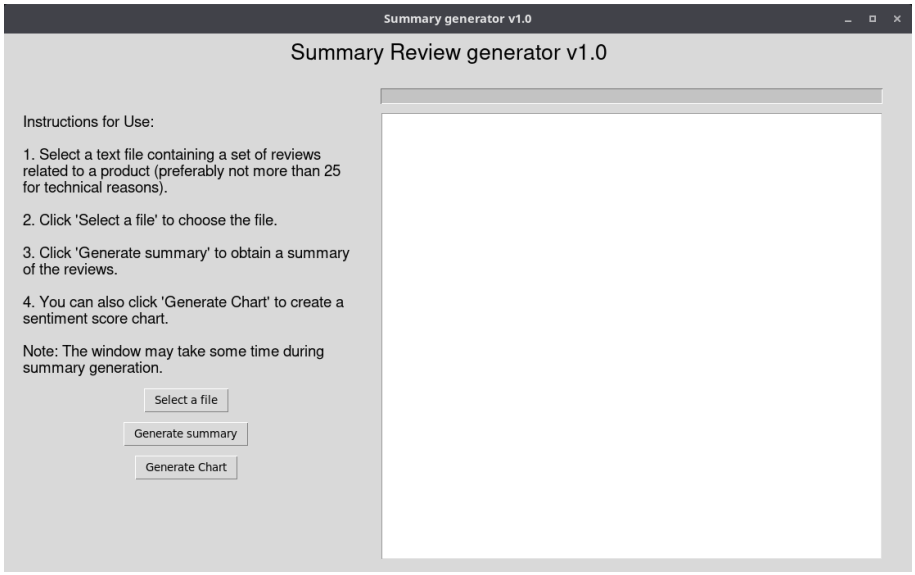


Figure 2.19: Screenshot of the Graphical User Interface (GUI)

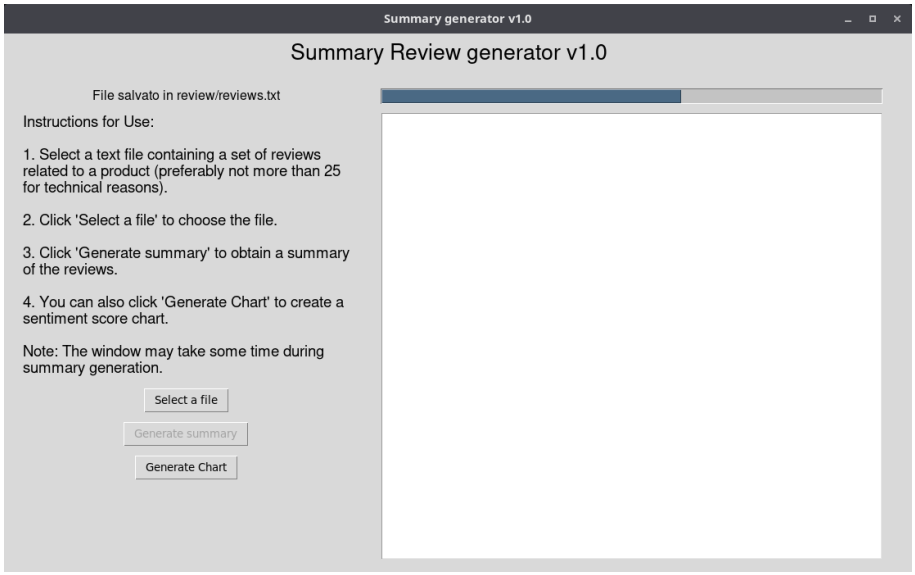


Figure 2.20: Screenshot of the Graphical User Interface (GUI)

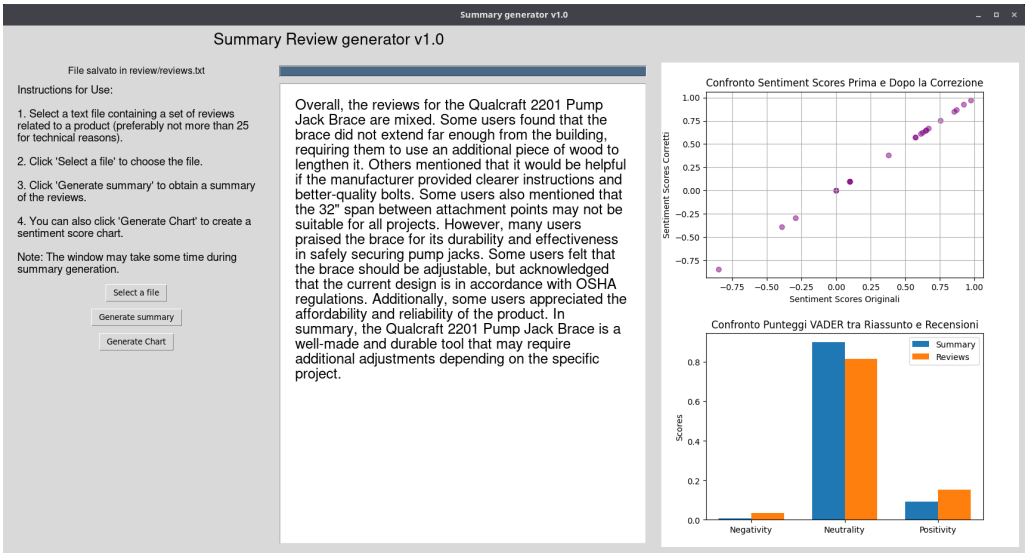


Figure 2.21: Screenshot of the Graphical User Interface (GUI)

The GUI aims to streamline the review analysis process, allowing users to obtain valuable insights without the need for extensive technical knowledge.

2.7.5 Summary and Conclusion

The Python application prototype, with its intuitive graphical user interface, offers a user-friendly and efficient way to process and summarize product reviews. Users can seamlessly select a file containing reviews, generate weighted summaries, and visualize sentiment trends with ease. This tool is designed to assist both consumers and manufacturers in making informed decisions and enhancing products based on user feedback. It's important to note that this application is a prototype, and there is potential for further development. Future enhancements could include the ability to analyze multiple products simultaneously, providing even more comprehensive insights for users and manufacturers.

Chapter 3

Results

Our objective was to generate Amazon review summaries for various products, providing Amazon users with the option to view a concise summary of reviews rather than going through all of them. After generating these summaries, we applied various evaluation methods.

It is important to note that none of the methods we employed can provide a perfect result, as there is no single approach that can definitively determine whether the API-generated summary is right or wrong. Furthermore, all the methods we used evaluate the summaries from a syntactical perspective, but not semantically. Therefore, a comprehensive evaluation of the meaning of the summary is not conducted. Instead, our focus was on assessing whether the API-generated summary effectively captures the content of the reviews and how closely it aligns with human-generated summaries.

In principle, the products generated by ChatGPT perform well. Based on the results of sentiment analysis, human evaluations through questionnaires, and similarity measurements, these summaries appear to be approaching excellence. However, there are instances where ChatGPT may fail to grasp all the nuances, primarily due to the informal and occasionally misleading language used in some reviews (where the review content may not align with the star ratings given).

It's worth considering that, for a company, relying solely on ChatGPT-generated summaries may not always be ideal, as a more reliable summary may be needed to ensure that no critical information is lost.