Importing the neccessary packages

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
from dotenv import load_dotenv
import re
from transformers import pipeline,  AutoTokenizer, AutoModelForSequenceClassifica
import torch
import seaborn
import openai
```

```
/Users/gt-europe/Documents/RediQuest-/turmerik/lib/python3.9/site-packages/ur
  warnings.warn(
/Users/gt-europe/Documents/RediQuest-/turmerik/lib/python3.9/site-packages/tq
  from .autonotebook import tqdm as notebook_tqdm
```

Loading the variables

```
load_dotenv()

# Access the variables
client_id = os.getenv("APP_ID")
client_secret = os.getenv("CLIENT_SECRET")
user_agent = os.getenv("USER_AGENT")
openai.api_key = os.getenv("OPENAI_API_KEY")
```

```
import praw
reddit = praw.Reddit(
    client_id=client_id,
    client_secret=client_secret,
    user_agent=user_agent,
    user_name= "RediQuest",
)
```

Verifying Authentication

```
# Access a public subreddit to verify authentication
subreddit = reddit.subreddit("test")  # "test" is a public, general subreddit

# Fetch and print the titles of the top 5 posts in the "test" subreddit
for post in subreddit.hot(limit=5):
    print(post.title)
```

```
Some test commands
Testing 123
test2 title
test1 title
New post test comment 04/11/2024 12:49:46
```

Example: Fetching posts from a subreddit

```
subreddit = reddit.subreddit("clinicaltrials")
for post in subreddit.hot(limit=10):
    print(post.title)
```

```
Clinical Trials Discussion Thread - Week of 2021-07-18
Consider supporting clinical research education and accessibility this holida
Seeking Feedback from Pregnant & Post-Partum Patients Living with Moderate to
The Story of Mucinex - How a Common Drug Became a Household Name
Webinar Tomorrow: Hear how experiential data and feedback from the patient's p
Video on Eroom's Law (Opposite of Moore's Law for Pharma R&D)
Hear from Jane Myles, Vice President of Clinical Trial Innovation at Curebase
Do you use e-cigarettes? Are you 16-20 years old?
Remote Research Study for Current Smokers!
Research Volunteers Needed - Connecticut
```

Step 1: Define Intent Keywords for Classification

These keywords help identify the main intent of each comment. We have three primary intents:

"interest_in_research": Identifies users expressing interest in participating in studies.

"treatment_frustration": Identifies users frustrated with current treatments.

"curiosity_in_advancements": Identifies users curious about neuroscience advancements.

```
intents = {
    "interest_in_research": [
        "clinical trial", "research study", "participate", "join study", "interes
        "medical study", "volunteer", "sign up", "join research", "take part", "r
        "looking to join", "study enrollment"
    ],
    "treatment_frustration": [
        "not working", "side effects", "ineffective", "need alternative", "frustr
        "treatment failure", "drug not effective", "need new options", "no improv
        "disappointed with treatment", "unsatisfied with results", "lack of progr
        "tired of side effects", "doesn’t help", "stopped working"
    ],
    "curiosity_in_advancements": [
        "new treatment", "latest research", "neuroscience development", "cutting-
        "new discovery", "breakthrough", "advances in neuroscience", "upcoming st
        "future research", "recent findings", "neuroscience updates", "new therap
        "innovative treatment", "explore research", "scientific progress"
    ]
}
```

Step 2: Define Data Structure for Storing Results

This list will hold dictionaries for each comment, including the post and comment text, recognized intent, and sentiment analysis results.

```
data = []
```

Step 3: Helper Function to Clean Text Data

This function removes personal or identifiable information such as Reddit usernames (u/username), subreddit mentions (r/subreddit), and common locations or institutions that may reveal identity.

```
def clean_text(text):
    text = re.sub(r'u/[A-Za-z0-9_-]+', '[REDACTED]', text)
    text = re.sub(r'r/[A-Za-z0-9_-]+', '[REDACTED]', text)
    text = re.sub(r'\b(New York|California|Johns Hopkins)\b', '[LOCATION]', text)
    return text
```

Step 4: Intent Recognition Function

This function uses the intent keywords defined above to classify each comment by intent.

It iterates over each intent category and checks for matching keywords in the comment text.

If a keyword is found, the function returns the relevant intent.

If no keyword matches, it returns "other" to mark the comment as having no specific intent.

```
def recognize_intent(comment):
    for intent, keywords in intents.items():
        for keyword in keywords:
            if keyword in comment.lower():
                return intent
    return "other"
```

Step 5: Initialize Sentiment Analysis Pipeline

We use Hugging Face's `transformers` pipeline for sentiment analysis.

This pipeline will analyze each comment's sentiment to determine if it's positive, neutral, or negative.

```
# Initialize the tokenizer and model
tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased-finetuned-sst-2-
model = AutoModelForSequenceClassification.from_pretrained("distilbert-base-uncasec


# Function to split text by tokens and analyze sentiment for each chunk
def analyze_sentiment(text, max_length=512):
    # Tokenize and split text into manageable chunks
    tokens = tokenizer.encode(text, add_special_tokens=False)
    chunks = [tokens[i:i + max_length] for i in range(0, len(tokens), max_length)]

    # Analyze sentiment for each chunk
    sentiments = []
    for chunk in chunks:
        inputs = torch.tensor([chunk]).to(model.device)
        with torch.no_grad():
            outputs = model(inputs)
            logits = outputs.logits
            probabilities = torch.softmax(logits, dim=-1)
            label = 'POSITIVE' if probabilities[0][1] > probabilities[0][0] else 'N
            score = probabilities[0][1].item() if label == 'POSITIVE' else probabil
            sentiments.append({"label": label, "score": score})

    # Calculate average sentiment score across chunks
    avg_score = sum(s['score'] for s in sentiments) / len(sentiments)
    final_label = sentiments[0]['label']  # Using the label of the first chunk as a

    return final_label, avg_score
```

Step 6: Fetch and Analyze Reddit Posts and Comments

This section collects posts and comments from the "neuroscience" subreddit.

For each post, we gather a limited number of comments, clean the text, recognize intent,

and perform sentiment analysis. The results are stored in the `data` list.

```
subreddit = reddit.subreddit("neuroscience")
post_limit = 500  # Number of posts to fetch
comment_limit = 5  # Number of comments per post to analyze
```

```
for post in subreddit.hot(limit=post_limit):
    post_title = clean_text(post.title)
    post_body = clean_text(post.selftext)

    post.comments.replace_more(limit=0)
    for comment in post.comments[:comment_limit]:
        comment_body = clean_text(comment.body)

        # Analyze sentiment by splitting into chunks if necessary
        sentiment_label, avg_sentiment_score = analyze_sentiment(comment_body)

        intent = recognize_intent(comment_body)

        # Append result to data structure
        data.append({
            "post_title": post_title,
            "post_body": post_body,
            "comment_body": comment_body,
            "intent": intent,
            "sentiment_label": sentiment_label,
            "sentiment_score": avg_sentiment_score
        })

# Step 9: Save the collected and processed data to a CSV file
df = pd.DataFrame(data)
df.to_csv("neuroscience_analysis_data.csv", index=False)

print("Data collection, intent recognition, sentiment analysis, and saving to CSV
```

```
Token indices sequence length is longer than the specified maximum sequence le
Data collection, intent recognition, sentiment analysis, and saving to CSV cor
```

```
df1 = pd.read_csv("/Users/gt-europe/Documents/RediQuest-/neuroscience_analysis_da
```

```
df1.shape
```

```
(1399, 6)
```

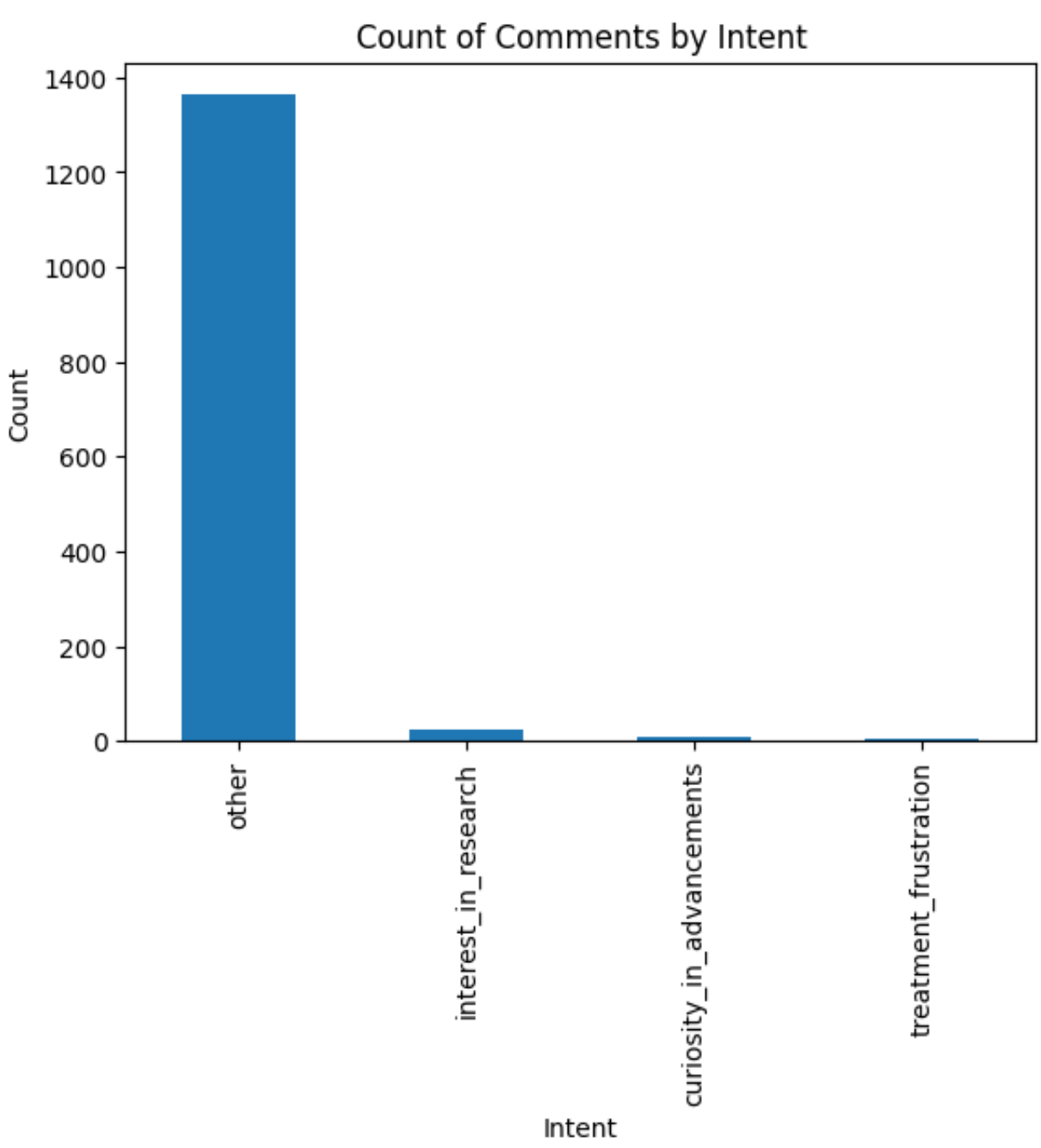
```
print(df1.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1399 entries, 0 to 1398
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   post_title       1399 non-null   object 
 1   post_body        673 non-null    object 
 2   comment_body     1399 non-null   object 
 3   intent           1399 non-null   object 
 4   sentiment_label  1399 non-null   object 
 5   sentiment_score  1399 non-null   float64
dtypes: float64(1), object(5)
memory usage: 65.7+ KB
None
```

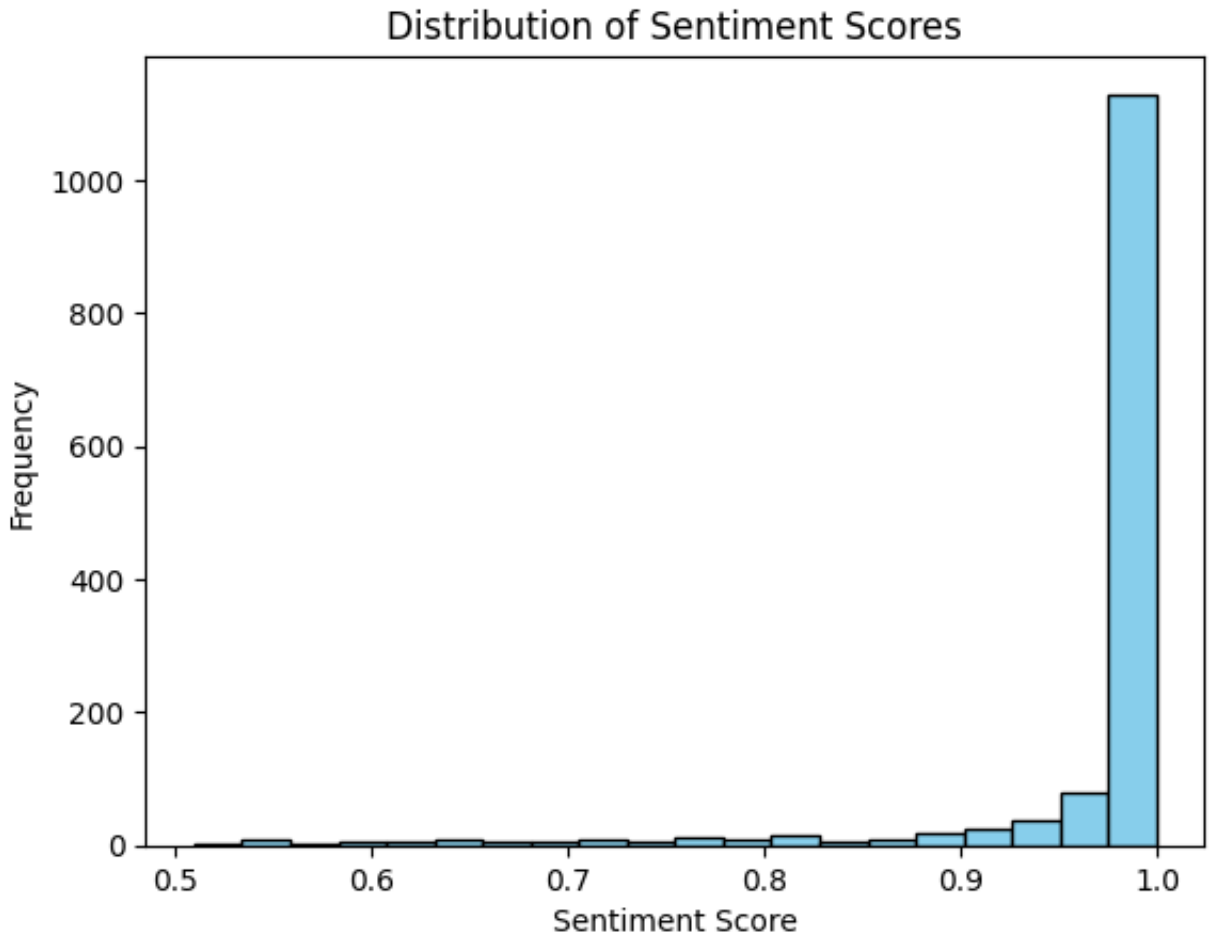
```
# Summary statistics for numerical columns
print(df1.describe())
```

```
       sentiment_score
count      1399.000000
mean          0.967700
std           0.081223
min           0.510018
25%           0.985553
50%           0.997326
75%           0.998385
max           0.999820
```

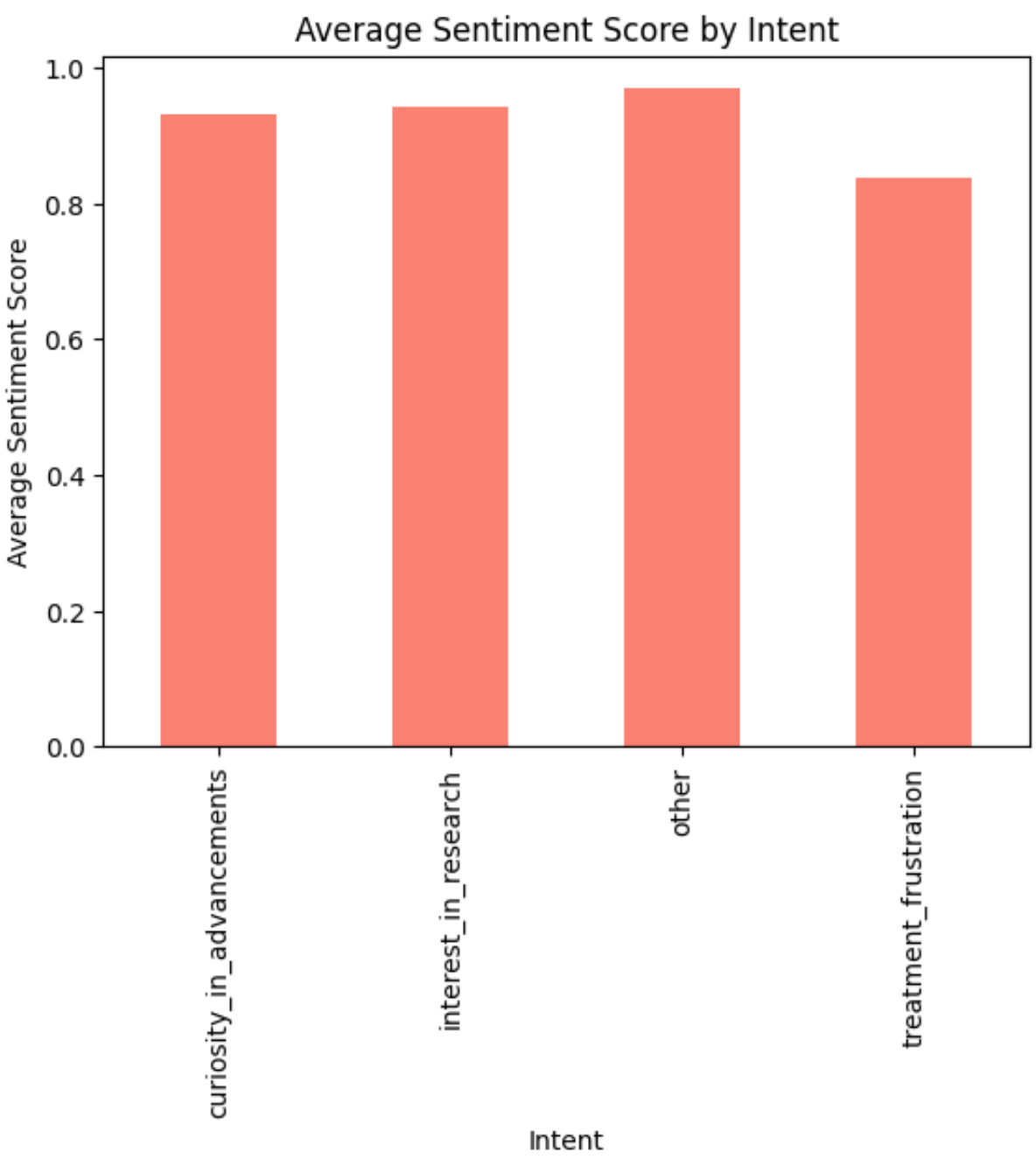
```
import matplotlib.pyplot as plt

# Count plot for intents
intent_counts = df1['intent'].value_counts()
intent_counts.plot(kind='bar', title='Count of Comments by Intent')
plt.xlabel('Intent')
plt.ylabel('Count')
plt.show()
```



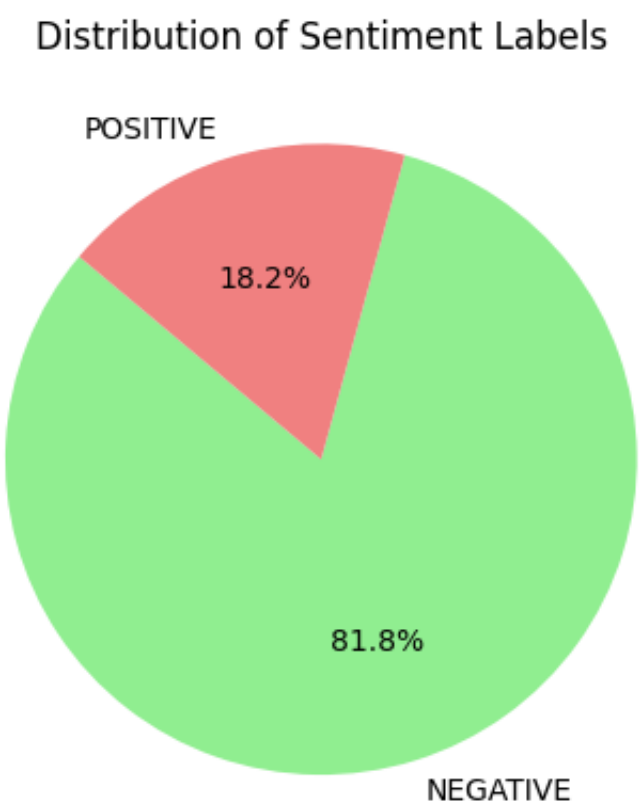
```
# Distribution of sentiment scores
plt.hist(df1['sentiment_score'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Sentiment Scores')
plt.xlabel('Sentiment Score')
plt.ylabel('Frequency')
plt.show()
```




```
# Average sentiment score per intent
avg_sentiment_per_intent = df1.groupby('intent')['sentiment_score'].mean()

# Bar plot for average sentiment score by intent
avg_sentiment_per_intent.plot(kind='bar', color='salmon')
plt.title('Average Sentiment Score by Intent')
plt.xlabel('Intent')
plt.ylabel('Average Sentiment Score')
plt.show()
```

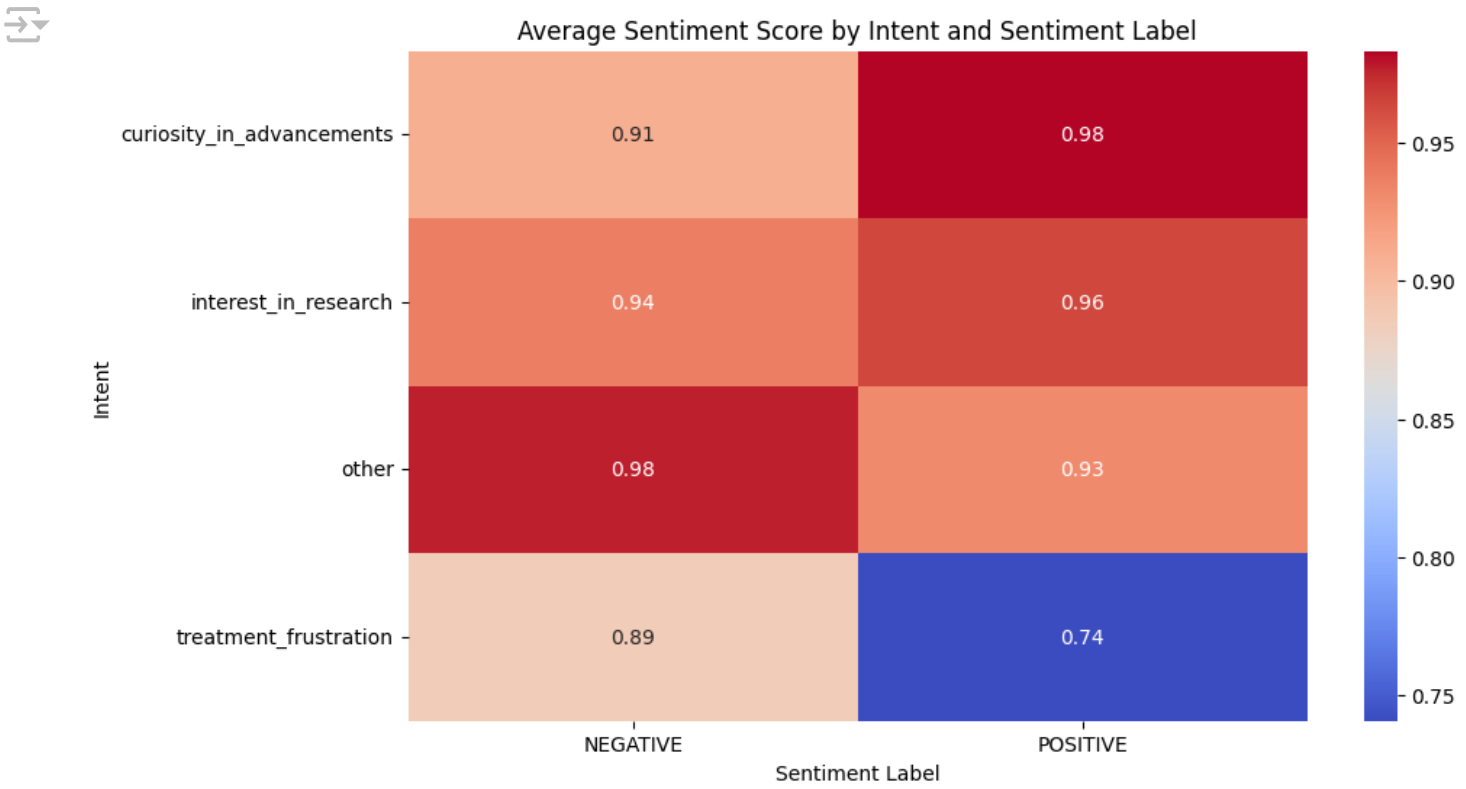

Average Sentiment Score by Intent
Average Sentiment Score
1.0
0.8
0.6
0.4
0.2
0.0
curiosity_in_advancements
interest_in_research
other
treatment_frustration
Intent

```
# Count plot for sentiment labels
label_counts = df1['sentiment_label'].value_counts()
label_counts.plot(kind='pie', autopct='%1.1f%%', startangle=140, colors=['lightgr
plt.title('Distribution of Sentiment Labels')
plt.ylabel('')  # Hide the y-label
plt.show()
```



```
# Average sentiment score by intent and sentiment label
sentiment_by_intent_label = df1.groupby(['intent', 'sentiment_label'])['sentiment_
```

```
# Heatmap of average sentiment score by intent and sentiment label
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.heatmap(sentiment_by_intent_label, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Average Sentiment Score by Intent and Sentiment Label')
plt.xlabel('Sentiment Label')
plt.ylabel('Intent')
plt.show()
```

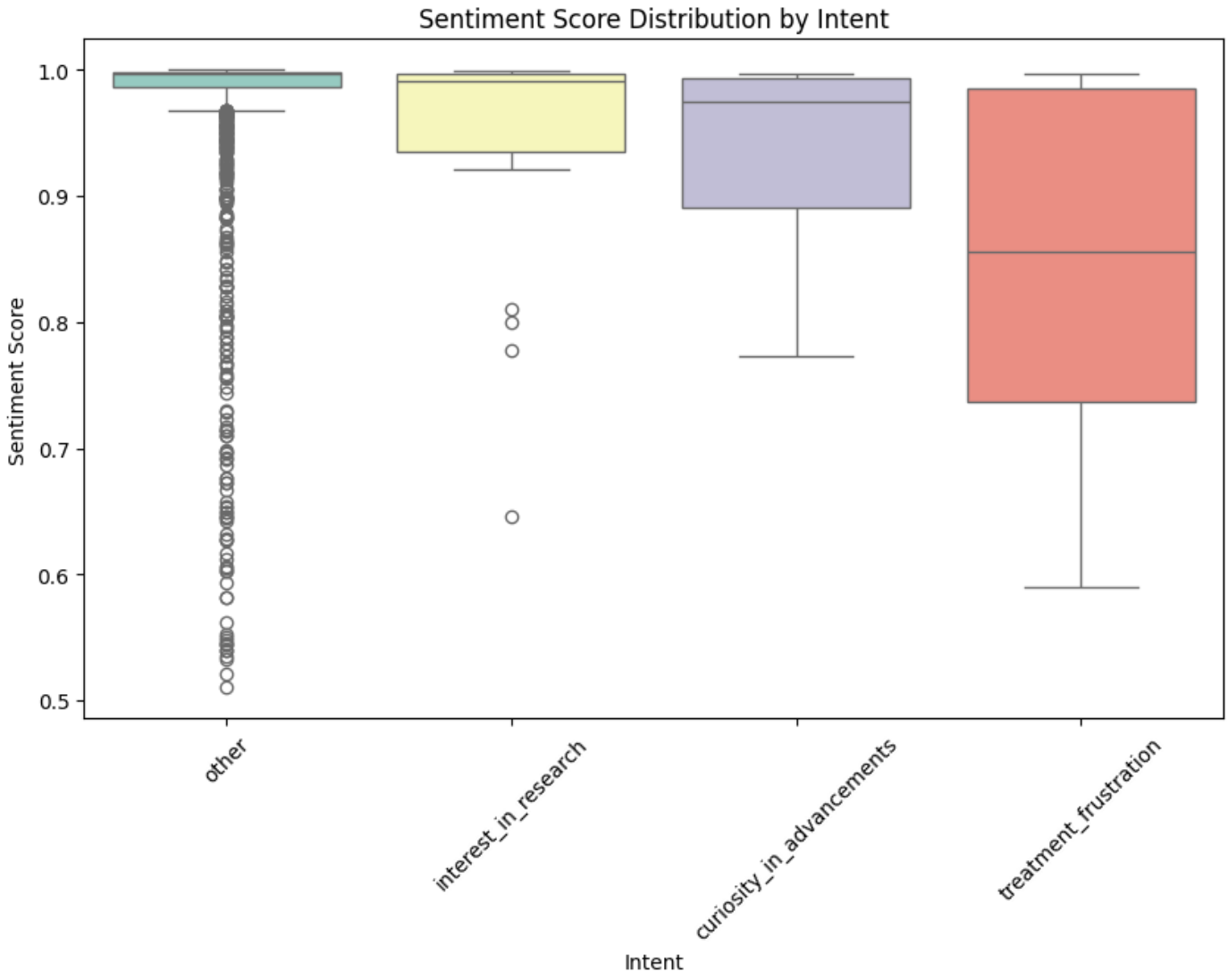


```
from wordcloud import WordCloud

# Combine all comment text
all_comments = " ".join(df1['comment_body'].astype(str))

# Generate a word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(a

# Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Most Common Words in Comments')
plt.show()
```


```
# Boxplot of sentiment scores by intent
plt.figure(figsize=(10, 6))
sns.boxplot(x='intent', y='sentiment_score', data=df1, palette="Set3")
plt.title('Sentiment Score Distribution by Intent')
```

```
plt.xlabel('Intent')
plt.ylabel('Sentiment Score')
plt.xticks(rotation=45)
plt.show()
```

```
/var/folders/gy/gfwflynn5vv33mfz9ph39zmc0000gn/T/ipykernel_3095/550964106.py:3
Passing `palette` without assigning `hue` is deprecated and will be removed in
  sns.boxplot(x='intent', y='sentiment_score', data=df1, palette="Set3")
```



Generating Personalized Messages Using OpenAI

```
# Load the dataset
df = pd.read_csv("neuroscience_analysis_data.csv")

# Filter for positive sentiment and 'interest_in_research' intent, select the top
positive_research_comments = df[
    (df["sentiment_label"].str.strip().str.lower() == "positive") &
    (df["intent"].str.strip().str.lower() == "interest_in_research")
].head(5)

# Display the comment body of the filtered comments
print(positive_research_comments[["comment_body"]])
```

```
                                          comment_body
96   "The European Medicines Agency (EMA) did rejec...
177  I always always use human subjects. People are...
190  I don't think anything about human neuroscienc...
683  Hi everyone,\n\nI am soon graduating with a BS...
856  AAV is a great vector for Neuroscience. Our la...
```

```
import pandas as pd

# Load the dataset
df = pd.read_csv("neuroscience_analysis_data.csv")

# Filter for positive sentiment and 'interest_in_research' intent, select the top
positive_research_comments = df[
    (df["sentiment_label"].str.strip().str.lower() == "positive") &
    (df["intent"].str.strip().str.lower() == "interest_in_research")
].head(5)
def generate_personalized_message(comment):
    prompt = (
        f"The user expressed interest in neuroscience research and wrote: '{comme
        "Write a friendly, non-intrusive message inviting them to learn more abou
    )

    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "system", "content": "You are a helpful assistant specialize
            {"role": "user", "content": prompt}
        ],
```

```
        max_tokens=100,
        temperature=0.7
    )

    message = response.choices[0].message['content'].strip()
    return message

# Generate personalized messages for each selected comment
positive_research_comments["personalized_message"] = positive_research_comments["


# Display the DataFrame with the generated messages
print(positive_research_comments[["comment_body", "personalized_message"]])

# Optionally, save the DataFrame to a CSV file
positive_research_comments.to_csv("personalized_messages_simulated.csv", index=Fa
```

```
                                          comment_body  \
96   "The European Medicines Agency (EMA) did rejec...
177  I always always use human subjects. People are...
190  I don't think anything about human neuroscienc...
683  Hi everyone,\n\nI am soon graduating with a BS...
856  AAV is a great vector for Neuroscience. Our la...

                                personalized_message
96   Hello! We noticed your interest in neuroscienc...
177  Hi there! It's great to see your passion for n...
190  Greetings! Your enthusiasm for neuroscience ca...
683  Hello! We saw your interest in neuroscience, a...
856  Hi! It’s inspiring to see your engagement with...
```

```
df2=  pd.read_csv("/Users/gt-europe/Documents/RediQuest-/personalized_messages_si


# Display the first row's comment and personalized message
first_comment = df2.loc[0, "comment_body"]
first_message = df2.loc[0, "personalized_message"]

print("Comment:", first_comment)
```

```
Comment: "The European Medicines Agency (EMA) did reject the marketing author

"Lecanemab has shown some effectiveness in clinical trials, particularly in s
```

```
print("Personalized Message:", first_message)
```

```
Personalized Message: Hello! We noticed your interest in neuroscience researcl
```

Start coding or generate with AI.