--- Sentiment Analysis Program ---

Without library or LLM

```
import json
import random
import time

def get_sentiment(text: str) -> dict:
   # This is a placeholder for a real API call. In a live application,
   # you would use a library like 'requests' to send the text to a model
   # and get a response.
   # The prompt would be crafted to instruct the LLM to return a specific
output.
   # For example: "Analyze the sentiment of the following text and return
   # only 'Positive', 'Negative', or 'Neutral'."

   # Simulating a small delay to mimic network latency.
   time.sleep(random.uniform(0.5, 1.5))

   # A simple keyword-based logic to simulate the LLM's response.
   # A real LLM would be much more accurate and nuanced.
   negative_keywords = ["bad", "terrible", "worst", "disappointed",
"horrible", "sad"]
   positive_keywords = ["good", "great", "excellent", "best", "happy",
"amazing", "love"]

   lower_text = text.lower()

   # Check for negative and positive keywords
   is_negative = any(word in lower_text for word in negative_keywords)
   is_positive = any(word in lower_text for word in positive_keywords)

   if is_positive and not is_negative:
       sentiment = "Positive"
   elif is_negative and not is_positive:
       sentiment = "Negative"
```

```
    else:
        # If both or neither are found, classify as Neutral.
        sentiment = "Neutral"

    return {"sentiment": sentiment}

# --- Demonstrations of the function ---

print("--- Example 1: Positive Sentiment ---")
text1 = "This is a great tool for sentiment analysis!"
result1 = get_sentiment(text1)
print(f"Input: '{text1}'")
print(f"Output: {json.dumps(result1, indent=2)}\n")

print("--- Example 2: Negative Sentiment ---")
text2 = "I am so disappointed with the service."
result2 = get_sentiment(text2)
print(f"Input: '{text2}'")
print(f"Output: {json.dumps(result2, indent=2)}\n")

print("--- Example 3: Neutral Sentiment ---")
text3 = "The meeting will be at 3 PM today."
result3 = get_sentiment(text3)
print(f"Input: '{text3}'")
print(f"Output: {json.dumps(result3, indent=2)}\n")

print("--- Example 4: Mixed Sentiment ---")
text4 = "The product is great, but the delivery was terrible."
result4 = get_sentiment(text4)
print(f"Input: '{text4}'")
print(f"Output: {json.dumps(result4, indent=2)}\n")
```

```
--- Example 1: Positive Sentiment ---
Input: 'This is a great tool for sentiment analysis!'
Output: {
  "sentiment": "Positive"
}

--- Example 2: Negative Sentiment ---
Input: 'I am so disappointed with the service.'
Output: {
  "sentiment": "Negative"
}

--- Example 3: Neutral Sentiment ---
Input: 'The meeting will be at 3 PM today.'
Output: {
  "sentiment": "Neutral"
}

--- Example 4: Mixed Sentiment ---
Input: 'The product is great, but the delivery was terrible.'
Output: {
  "sentiment": "Neutral"
}
```

With TextBlob library

```python
# sentiment_analyzer.py
#
# This program takes a text string as input and uses the TextBlob library
# to determine its sentiment as Positive, Negative, or Neutral.
# It then prints the sentiment classification for a series of test
strings.

from textblob import TextBlob

def get_sentiment(text: str) -> str:
    """
    Analyzes the sentiment of the input text using the TextBlob library.

    Args:
        text: The string to analyze.

    Returns:
        A string indicating the sentiment: "Positive", "Negative", or
"Neutral".
    """
    # Create a TextBlob object from the input text
    blob = TextBlob(text)

    # Get the polarity score, which ranges from -1.0 (very negative) to 1.0 (very positive)
    polarity = blob.sentiment.polarity

    # Classify the sentiment based on the polarity score
    if polarity > 0:
        return "Positive"
    elif polarity < 0:
        return "Negative"
    else:
        return "Neutral"

def main():
    """
    Runs the sentiment analysis on a predefined set of test strings to
    demonstrate the program's functionality for all three sentiments.
    """
    # Define a list of test strings for demonstration
    test_strings = {
        "Positive": [
            "I had a fantastic day and the weather was beautiful!",
            "This is the best movie I have ever seen.",
            "The customer service was excellent and the staff were very helpful."
        ],
        "Negative": [
            "The service was incredibly slow and the food was terrible.",
            "I'm so frustrated with this situation, nothing is working.",
            "The product was a complete disappointment."
        ],
        "Neutral": [
            "The sky is blue and the clouds are white.",
            "The meeting is scheduled for 2 PM on Tuesday.",
            "He walked from the office to the car."
        ],
        "":[
            "Last session of the day  http://twitpic.com/67ezh",
            "Shanghai is also really exciting (precisely -- skyscrapers galore). Good tweeps in China:  (SH)  (BJ).",
            "Recession hit Veronique Branquinho, she has to quit her company, such a shame!",
            "happy bday!"
        ]
    }

    print("--- Sentiment Analysis Program ---")
    print("This program analyzes the sentiment of a given text string.")
    print("-" * 35)

    for sentiment_type, strings in test_strings.items():
        print(f"\nTesting for {sentiment_type} sentiment:")
        for text in strings:
            result = get_sentiment(text)
            print(f'Text: "{text}"')
            print(f'Sentiment: {result}')
            print("-" * 35)

if __name__ == "__main__":
    main()
```

```
--- Sentiment Analysis Program ---
This program analyzes the sentiment of a given text string.
-----------------------------------

Testing for Positive sentiment:
Text: "I had a fantastic day and the weather was beautiful!"
Sentiment: Positive
-----------------------------------
Text: "This is the best movie I have ever seen."
Sentiment: Positive
-----------------------------------
Text: "The customer service was excellent and the staff were very helpful."
Sentiment: Positive
-----------------------------------

Testing for Negative sentiment:
Text: "The service was incredibly slow and the food was terrible."
```

```
Sentiment: Negative
----------------------------------
Text: "I'm so frustrated with this situation, nothing is working."
Sentiment: Negative
----------------------------------
Text: "The product was a complete disappointment."
Sentiment: Negative
----------------------------------

Testing for Neutral sentiment:
Text: "The sky is blue and the clouds are white."
Sentiment: Neutral
----------------------------------
Text: "The meeting is scheduled for 2 PM on Tuesday."
Sentiment: Neutral
----------------------------------
Text: "He walked from the office to the car."
Sentiment: Neutral
----------------------------------

Testing for  sentiment:
Text: "Last session of the day  http://twitpic.com/67ezh"
Sentiment: Neutral
----------------------------------
Text: "Shanghai is also really exciting (precisely -- skyscrapers galore). Good tweeps in China:
(SH)  (BJ)."
Sentiment: Positive
----------------------------------
Text: "Recession hit Veronique Branquinho, she has to quit her company, such a shame!"
Sentiment: Neutral
----------------------------------
Text: "happy bday!"
Sentiment: Positive
----------------------------------
```

The section highlighted in red is a great example of an inaccurate result from the current library. This is precisely where an LLM could be beneficial, as it would be able to correct these minor errors

---

With TextBlob library running on Kaggle data set (test.csv)

```python
# sentiment_analysis.py
#
# This program reads a CSV file, analyzes the sentiment of a specific
# text column using the TextBlob library, and provides a summary
# of the sentiment distribution.
#
# Before running, you must install TextBlob and download its corpora by
# running the following commands in your terminal or notebook cell:
# pip install textblob
# python -m textblob.download_corpora

import pandas as pd
from textblob import TextBlob

def get_sentiment(text: str) -> str:
   """
   Analyzes the sentiment of the input text using the TextBlob library.

   Args:
       text: The string to analyze.

   Returns:
       A string indicating the sentiment: "Positive", "Negative", or
"Neutral".
   """
   # Create a TextBlob object for the input text.
   analysis = TextBlob(str(text))

   # Classify sentiment based on polarity score.
   # Polarity ranges from -1 (very negative) to +1 (very positive).
   if analysis.sentiment.polarity > 0:
       return "Positive"
   elif analysis.sentiment.polarity < 0:
       return "Negative"
   else:
       return "Neutral"

def main():
   """
   Reads the test.csv file, analyzes the sentiment of the text column,
   and prints a summary of the results.
   """
   file_path = 'test.csv'

   try:
       # First, try to read the CSV with standard UTF-8 encoding.
       df = pd.read_csv(file_path)
   except UnicodeDecodeError:
       # If a UnicodeDecodeError occurs, try with latin1 encoding.
       df = pd.read_csv(file_path, encoding='latin1')

   # Check if the 'text' column exists in the DataFrame
   if 'text' not in df.columns:
       print("Error: The 'text' column was not found in the CSV file.")
       return

   # Initialize counters for each sentiment
   sentiment_counts = {
       'Positive': 0,
       'Negative': 0,
       'Neutral': 0,
       'Error': 0
   }

   total_rows = len(df)
   print(f"Starting sentiment analysis on {total_rows} rows from
'{file_path}'...")

   for index, row in df.iterrows():
       text_string = row['text']

       # Skip rows where the text field is empty or missing
       if pd.isna(text_string) or text_string == "":
           continue

       # Get sentiment from TextBlob
       sentiment_result = get_sentiment(text_string)

       # Update counters based on the sentiment
       if sentiment_result in sentiment_counts:
           sentiment_counts[sentiment_result] += 1
       else:
           sentiment_counts['Error'] += 1

   # Print the final summary of the sentiment counts
   print("\n--- Sentiment Analysis Summary ---")
   print(f"Total entries analyzed: {sum(sentiment_counts.values())}")
   print(f"Positive: {sentiment_counts['Positive']}")
   print(f"Negative: {sentiment_counts['Negative']}")
   print(f"Neutral: {sentiment_counts['Neutral']}")
   print(f"Errors: {sentiment_counts['Error']}")
   print("-" * 35)

if __name__ == "__main__":
   main()
```

```
Starting sentiment analysis on 4815 rows from 'test.csv'...

--- Sentiment Analysis Summary ---
Total entries analyzed: 3534
Positive: 1566
Negative: 714
Neutral: 1254
Errors: 0
-----------------------------------
```

We can observe here it’s not accurate reading with textblob library

---

With Gemini LLM API key

```
import os
os.environ['GEMINI_API_KEY'] = 'AIzaSyAax5QUwurcMv3OL2kI62muzhrEHJWbAYg'
```

```
# sentiment_analyzer.py
#
# This program takes a text string as input and uses a large language
model (LLM)
```

```
# to determine its sentiment as Positive, Negative, or Neutral.
# It then prints the sentiment classification for a series of test
strings.
#
# Before running, you must set your GEMINI_API_KEY as an environment
variable.

import os
import requests
import json
import textwrap
import time # Import the time module for sleep functionality

def get_sentiment(text: str) -> str:
   """
   Analyzes the sentiment of the input text using a language model API.

   Args:
       text: The string to analyze.

   Returns:
       A string indicating the sentiment: "Positive", "Negative", or
"Neutral".
   """
   # Use the Gemini API.
   api_key = os.getenv("GEMINI_API_KEY")
   if not api_key:
       return "Error: GEMINI_API_KEY environment variable not set."

   # This URL points to the gemini-2.5-flash-preview-05-20 model, which is
a fast
   # and efficient model for this kind of classification task.
   url =
f"https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash
-preview-05-20:generateContent?key={api_key}"

   # The prompt is carefully crafted to instruct the model to act as a
   # sentiment classifier and to respond with only one of the three
   # specified words to ensure consistent output.
   prompt_text = textwrap.dedent(f"""
       Analyze the sentiment of the following text and classify it as
       'Positive', 'Negative', or 'Neutral'. Respond with only one of
these three words.

       Text:
       {text}
   """).strip()

   # The payload is the JSON object that contains the request data for the
API.
   payload = {
       "contents": [
           {"parts": [{"text": prompt_text}]}
       ],
       "generationConfig": {
           "temperature": 0.0,  # Set a low temperature for predictable
output
           "candidateCount": 1
       }
   }

   # Implement a retry mechanism with exponential backoff for a more
robust program
   max_retries = 5
   for i in range(max_retries):
       try:
           # Make the API call to the language model.
           response = requests.post(url, headers={"Content-Type":
"application/json"}, json=payload)
           response.raise_for_status()  # Raise an exception for bad
status codes

           result = response.json()

           # Extract the sentiment from the API's JSON response.
           sentiment =
result['candidates'][0]['content']['parts'][0]['text'].strip().capitalize(
)

           # Check if the output is one of the expected values.
```

```
            if sentiment in ["Positive", "Negative", "Neutral"]:
                return sentiment
            else:
                # Fallback for unexpected model output.
                return "Neutral (uncertain)"

        except requests.exceptions.RequestException as e:
            # Check for a "Too Many Requests" error (status code 429)
            if response.status_code == 429 and i < max_retries - 1:
                wait_time = 2 ** i # Exponential backoff: 1, 2, 4, 8, etc.
seconds
                print(f"Rate limit exceeded (429). Retrying in {wait_time}
seconds...")
                time.sleep(wait_time)
            else:
                # Re-raise the exception if it's not a 429 or if max
retries are reached
                return f"Error with API request: {e}"
        except (KeyError, IndexError) as e:
            # Catch and report any errors if the JSON response is not as
expected.
            return f"Error parsing API response: {e}"

    return "Error: Failed to get a response after multiple retries."

def main():
    """
    Runs the sentiment analysis on a predefined set of test strings to
    demonstrate the program's functionality for all three sentiments.
    """
    # Define a list of test strings for demonstration
    test_strings = {
        "Positive": [
            "I had a fantastic day and the weather was beautiful!",
            "This is the best movie I have ever seen.",
            "The customer service was excellent and the staff were very
helpful."
        ],
        "Negative": [
            "The service was incredibly slow and the food was terrible.",
            "I'm so frustrated with this situation, nothing is working.",
            "The product was a complete disappointment.",
            "Recession hit Veronique Branquinho, she has to quit her
company, such a shame!" # The test case you provided
        ],
        "Neutral": [
            "The sky is blue and the clouds are white.",
            "The meeting is scheduled for 2 PM on Tuesday.",
            "He walked from the office to the car."
        ]
    }

    print("--- Sentiment Analysis Program ---")
    print("This program analyzes the sentiment of a given text string.")
    print("-" * 35)

    for sentiment_type, strings in test_strings.items():
        print(f"\nTesting for {sentiment_type} sentiment:")
        for text in strings:
            result = get_sentiment(text)
            print(f'Text: "{text}"')
            print(f'Sentiment: {result}')
            print("-" * 35)

if __name__ == "__main__":
    main()
```

Output

```
--- Sentiment Analysis Program ---
This program analyzes the sentiment of a given text string.
-----------------------------------

Testing for Positive sentiment:
Text: "I had a fantastic day and the weather was beautiful!"
Sentiment: Positive
-----------------------------------
Text: "This is the best movie I have ever seen."
```

```
Sentiment: Positive
----------------------------------
Text: "The customer service was excellent and the staff were very helpful."
Sentiment: Positive
----------------------------------

Testing for Negative sentiment:
Text: "The service was incredibly slow and the food was terrible."
Sentiment: Negative
----------------------------------
Text: "I'm so frustrated with this situation, nothing is working."
Sentiment: Negative
----------------------------------
Text: "The product was a complete disappointment."
Sentiment: Negative
----------------------------------
Text: "Recession hit Veronique Branquinho, she has to quit her company, such a shame!"
Sentiment: Negative
----------------------------------

Testing for Neutral sentiment:
Text: "The sky is blue and the clouds are white."
Sentiment: Neutral
----------------------------------
Text: "The meeting is scheduled for 2 PM on Tuesday."
Sentiment: Neutral
----------------------------------
Text: "He walked from the office to the car."
Sentiment: Neutral
----------------------------------
```