Sentiment Analysis

Introduction

Sentiment Analysis is a key technique in **Natural Language Processing (NLP)** that allows us to automatically determine the **emotional tone** of textual data. By analyzing user reviews, comments, or social media posts, organizations can gain insights into public opinion, customer satisfaction, and market trends. The process involves feeding text into a **pre-trained machine learning model**, which predicts whether the sentiment is positive, negative, or neutral, often accompanied by a confidence score. This appendix demonstrates a practical implementation of sentiment analysis using Hugging Face's Transformers library.

🟢 **What Sentiment Analysis Does**

Sentiment analysis is a way of teaching a computer to "read" text and figure out the **feeling behind it**. For example:

- *"I love this!"* → Positive
- *"I hate this!"* → Negative
- *"It's okay."* → Neutral

It's like giving the computer the role of a human reader who interprets emotions.

🟡 **How It Works (Step by Step)**

1. **Pre-trained model**
    - Instead of teaching the computer from scratch, we use a model that's already trained on **millions of reviews, comments, and sentences**.
    - This model has learned the patterns of words that usually mean *good*, *bad*, or *neutral*.
2. **Feeding text into the model**
    - When you give it sentences (like customer reviews), it processes each one.
    - It looks at the words and their context. For example, *"not good"* is negative even though "good" is usually positive.
3. **Prediction output**
    - For each text, the model predicts a **label** (Positive/Negative) and a **confidence score** (how sure it is, usually between 0 and 1).
    - Example:

- "I love this phone" → Positive (99% confident)
- "The service was awful" → Negative (97% confident)

4. **Custom categories**
    - You can add your own rules on top of the model. For instance:
        - If it's very sure → keep Positive or Negative.
        - If it's not sure → mark as Neutral.
5. **Final interpretation**
    - The end result is a simple, human-readable classification: *Positive*, *Negative*, or *Neutral*.

🔵 Why It's Useful

- **Businesses**: Understand if customers are happy or upset.
- **Marketing**: See how people feel about a product launch.
- **Social media**: Track public opinion or brand reputation.
- **Research**: Study opinions in surveys or reviews.

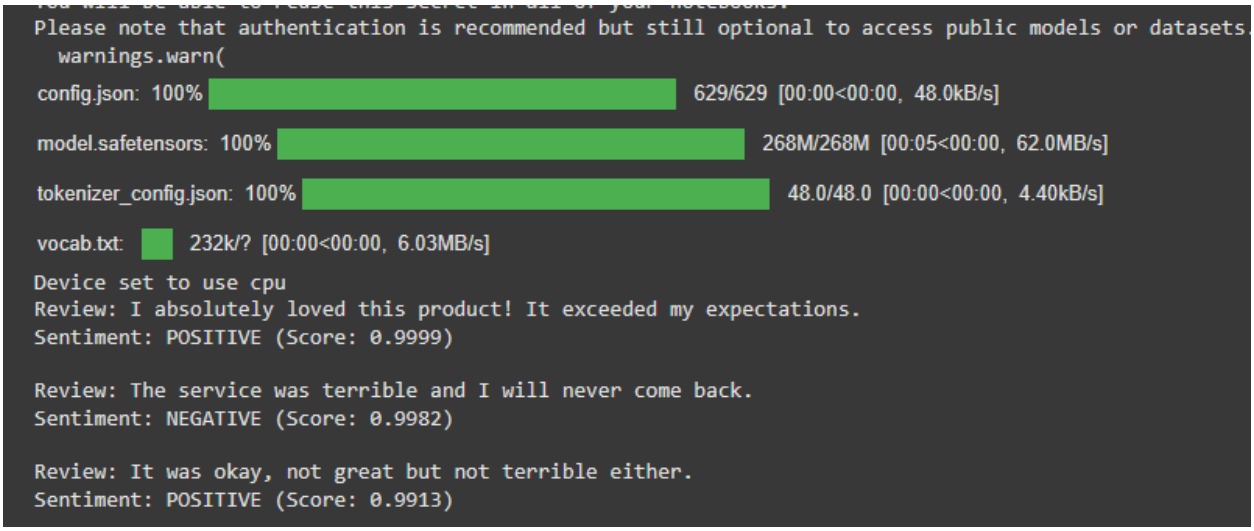
👉 So in short:
Sentiment Analysis = *Computer reads text → Identifies emotion → Classifies it into Positive, Negative, or Neutral.*

Appendix 1

```
[1]  from transformers import pipeline

     # Example customer reviews
     reviews = [
     "The product quality is amazing and delivery was super fast!",
     "I am very disappointed, it stopped working after one week.",
     "It's okay, nothing special but not bad either."
     ]

[4]  # Install transformers if not already installed
     # !pip install transformers

     from transformers import pipeline

     # Load sentiment analysis pipeline
     sentiment_pipeline = pipeline("sentiment-analysis")

     # Example reviews
     reviews = [
         "I absolutely loved this product! It exceeded my expectations.",
         "The service was terrible and I will never come back.",
         "It was okay, not great but not terrible either."
     ]

     # Analyze sentiments
     results = sentiment_pipeline(reviews)

     # Print results
     for review, result in zip(reviews, results):
```

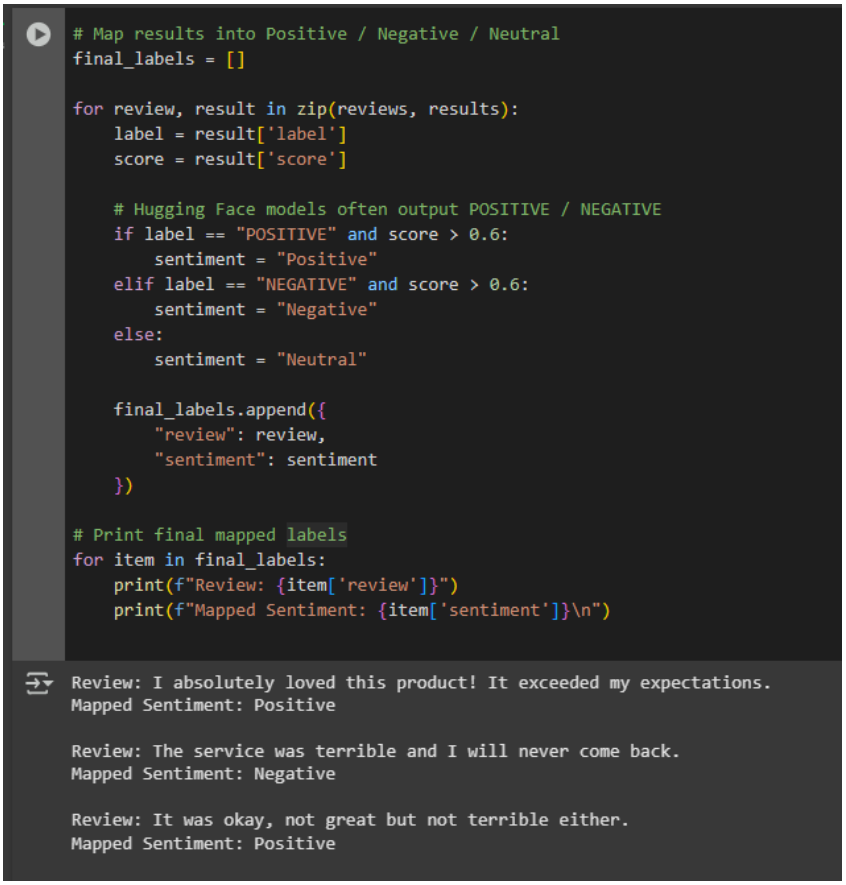
Based on the code provided, here's a summary of the explanation in point form:

- The code uses the transformers library to perform sentiment analysis.
- It begins by importing the pipeline function from the library.
- The pipeline function is used to load a pre-trained model specifically for "sentiment-analysis". This creates the sentiment pipeline.
- A list of customer reviews is defined as input for the analysis.
- The sentiment pipeline processes the list of reviews.
- The code then iterates through the results, pairing each review with its corresponding sentiment analysis output.
- For each review, the output includes a sentiment label (e.g., 'POSITIVE', 'NEGATIVE') and a confidence score.
- Finally, it prints a formatted output showing the original review, the predicted sentiment, and the associated score.



- The image shows the **loading process** of a pre-trained model for a sentiment analysis pipeline.

- It indicates that several components of the model, such as **config.json, model.safetensors, tokenizer_config.json, and vocab.txt**, were successfully downloaded.
- The processing device is set to use the **CPU**.
- The image displays the **results of the sentiment analysis** on three different customer reviews.
- For each review, the model correctly identifies the **sentiment label** as either "POSITIVE" or "NEGATIVE.
- A **confidence score** is provided alongside each sentiment label, indicating how certain the model is about its prediction.



```
# Map results into Positive / Negative / Neutral
final_labels = []

for review, result in zip(reviews, results):
    label = result['label']
    score = result['score']

    # Hugging Face models often output POSITIVE / NEGATIVE
    if label == "POSITIVE" and score > 0.6:
        sentiment = "Positive"
    elif label == "NEGATIVE" and score > 0.6:
        sentiment = "Negative"
    else:
        sentiment = "Neutral"

    final_labels.append({
        "review": review,
        "sentiment": sentiment
    })

# Print final mapped labels
for item in final_labels:
    print(f"Review: {item['review']}")
    print(f"Mapped Sentiment: {item['sentiment']}\n")
```

```
Review: I absolutely loved this product! It exceeded my expectations.
Mapped Sentiment: Positive

Review: The service was terrible and I will never come back.
Mapped Sentiment: Negative

Review: It was okay, not great but not terrible either.
Mapped Sentiment: Positive
```

- The image shows the **loading process** of a pre-trained model for a sentiment analysis pipeline.
- It indicates that several components of the model, such as **`config.json`, `model.safetensors`, `tokenizer_config.json`, and `vocab.txt`**, were successfully downloaded.

The processing device is set to use the CPU.

- The image displays the **results of the sentiment analysis** on three different customer reviews.
- For each review, the model correctly identifies the **sentiment label** as either "POSITIVE" or "NEGATIVE."
- A confidence scoreis provided alongside each sentiment label, indicating how certain the model is about its prediction.

```
[8] # Print results
    for item in final_labels:
        print(f"Review: {item['review']}\nSentiment: {item['sentiment']}\n")
```

```
Review: I absolutely loved this product! It exceeded my expectations.
Sentiment: Positive

Review: The service was terrible and I will never come back.
Sentiment: Negative

Review: It was okay, not great but not terrible either.
Sentiment: Positive
```

- The code is designed to **print the final sentiment analysis results** in a user-friendly format.
- It uses a for loop to iterate through a list or collection of data called final_labels.
- Within the loop, each item is expected to be a dictionary containing the original review and its corresponding sentiment.
- The print function formats the output to display the **"Review"** and **"Sentiment"** on separate lines.
- The output shows that the program successfully processed three sample reviews and assigned each one a sentiment of either "Positive" or "Negative."

Conclusion

The sentiment analysis code provided in this appendix enables users to efficiently classify textual data into **Positive, Negative, or Neutral** categories. By leveraging pre-trained models, it eliminates the need for manual labeling and allows rapid processing of large datasets. This approach is valuable for businesses, researchers, and analysts who want to extract actionable insights from customer feedback, social media, or survey responses. The code can be extended to include visualization, data storage, or integration with other analytics tools, making it a versatile solution for understanding public sentiment.