Amazon Product Review Analysis Service

Project Participants:

Trapti Damodar Balgi, Nidhi Choudhary

Project Overview:

The project aims to develop a service that collects and analyzes Amazon product reviews to enhance decision-making for consumers and businesses. The service is designed to automate the extraction of valuable insights from large volumes of customer feedback, which can be used to inform purchasing decisions and product improvements.

Project Goals:

- **Extract Product Names**: Automatically identify and retrieve product names from Amazon product pages.
- **Summarize Customer Feedback**: Generate concise summaries of customer reviews to highlight key feedback.
- **Sentiment Analysis**: Assess customer satisfaction by categorizing reviews into positive, negative, and neutral sentiments.
- **Topic Identification**: Analyze reviews to identify key topics and trends related to the products.
- **Data Storage**: Store processed review data in a secure storage system for future retrieval and analysis.

Software Components

1. **REST API (Flask-based API)**
   Purpose:
   Serves as the interface for users to interact with the application. Users submit Amazon product URLs and retrieve analysis results through this API.
   Why Chosen:
   - Advantages: Flask is lightweight, simple to integrate, and flexible, making it ideal for building APIs. It also has great community support.
   - Disadvantages: Flask is not as feature-rich as other frameworks like Django, although it may require more effort for scaling and adding advanced features.

2. **Message Queues (Redis)**
   Purpose: Redis is used to manage and distribute tasks across worker nodes, ensuring efficient task coordination.
   Why Chosen:
   - Advantages: Redis provides fast data access, handles high throughput, and supports message queuing efficiently. It's an ideal tool for decoupling components and handling concurrent tasks.
   - Disadvantages: Redis is an in-memory database, so large datasets may consume a significant amount of memory, and persistence of data needs careful management.

3. **Storage (MinIO)**
   Purpose: MinIO is used to store large volumes of processed review data, including sentiment analysis results, summaries, and topic models.
   Why Chosen:
   - Advantages: MinIO is lightweight, highly scalable, and compatible with the S3 API, making it ideal for cloud-native applications. It offers high performance for object storage.
   - Disadvantages: It requires proper configuration for high availability and fault tolerance in large-scale applications.

4. **Text Analysis (Pre-trained Models)**
   Purpose: These models handle various text processing tasks: summarization, sentiment analysis, and topic modeling.
   - Summarization: The "facebook/bart-large-cnn" model is used to generate concise summaries of product reviews.
   - Sentiment Analysis: The "joeddav/distilbert-base-uncased-go-emotions-student" model analyzes the emotional tone of the reviews (positive, negative, or neutral).
   - Topic Modeling: Latent Dirichlet Allocation (LDA) is used to identify common topics across the reviews.

   Why Chosen:
   - Advantages: These pre-trained models are state-of-the-art for NLP tasks. They save time and resources by eliminating the need to train models from scratch.
   - Disadvantages: Pre-trained models can be computationally expensive and require powerful hardware for processing large volumes of data.

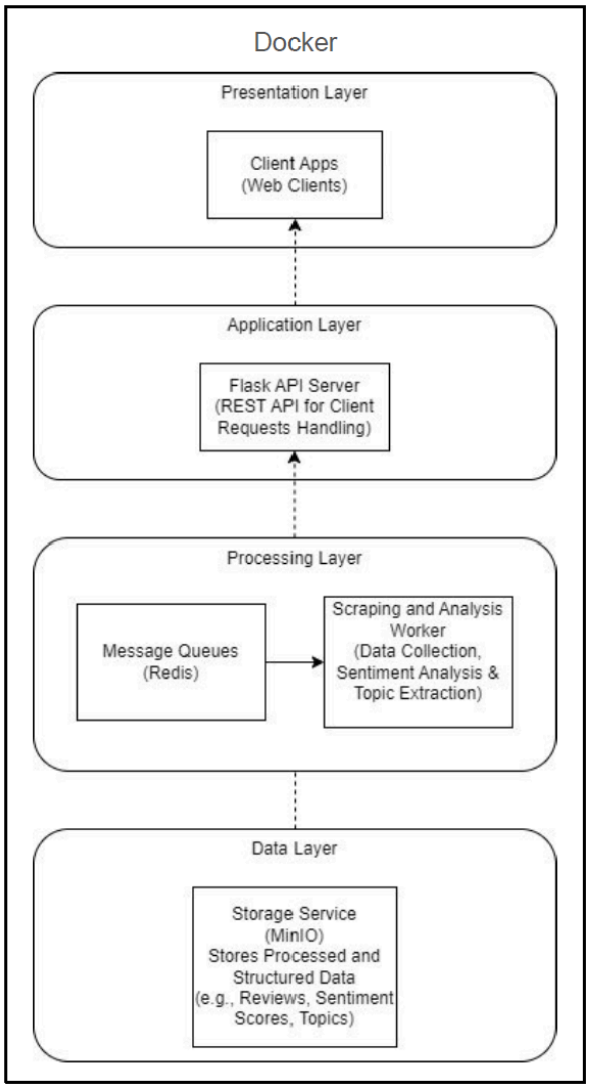
5. **Docker**

   Purpose: Docker containerizes the entire application, ensuring a consistent development, testing, and production environment.

   Why Chosen:

   - Advantages: Docker simplifies deployment and scaling by isolating the application and its dependencies into containers. It ensures the same environment across different systems.
   - Disadvantages: Docker introduces some overhead, and managing container orchestration for scaling can be complex if not properly configured.

Architectural Diagram



Interaction between different Software Components

- **Presentation Layer:**
  Users interact with the service through a web client interface, submitting Amazon product URLs and viewing analysis results, which are processed and delivered via the Flask REST API.
- **Application Layer:**
  The Flask API processes incoming requests from the user and forwards the tasks to Redis. It acts as the intermediary between the user-facing components and the backend processing layer.
- **Processing Layer:**
  Redis queues tasks for the worker node. The worker node performs scraping, sentiment analysis, topic extraction, and summarization using the pre-trained models. After processing, results are sent to MinIO for storage.
- **Data Layer:**
  MinIO stores all processed data, including the raw reviews, sentiment analysis results, topics, and summaries. This ensures that the data is persistently stored and can be easily accessed for later use.
- **Deployment:**
  Docker containers are used to ensure that the application is packaged in an isolated environment. This simplifies deployment, testing, and scaling of the components as needed.

By breaking down the components into these sections, we ensure that the system is modular and maintainable, with each component performing a specific role in the overall process.

Debugging and Testing

Developed and tested each component separately to ensure functionality:

- **Scraping:** Validated the accuracy of data extraction from Amazon product pages by scraping multiple pages and verifying the extracted product information and reviews.
- **UI and REST API:** Tested the functionality of API endpoints, ensuring correct data submission and retrieval. Verified the user interface (UI) for seamless interaction between the user and the system.
- **Summarization:** For the summarization process, we initially tested the T5-small model but switched to facebook/bart-large-cnn for better performance. While T5-small provided concise summaries, it sometimes missed important context or details. After switching to BART-large-cnn, we tested its ability to handle longer reviews and condense them while maintaining key insights and coherence. The BART-large-cnn model consistently produced more accurate and contextually

rich summaries, making it the ideal choice for summarizing Amazon product reviews.

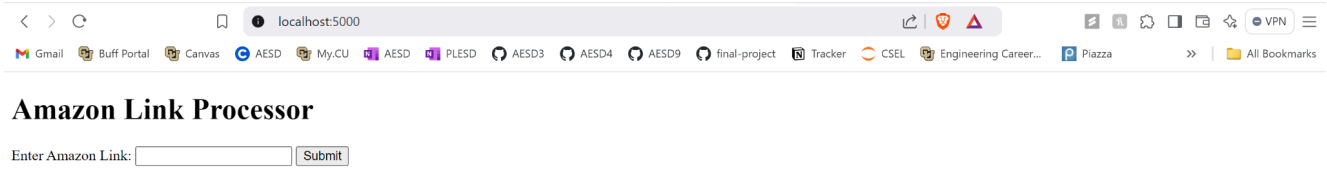
- **Product Name Extraction:** Verified the accuracy of product name extraction from Amazon pages to ensure correct identification and display of the product.
- **Topic Extraction and Sentiment Analysis:**Tested the effectiveness of topic modeling using Latent Dirichlet Allocation (LDA), ensuring that it accurately identified key themes and topics from the reviews. Additionally, the sentiment analysis was evaluated using the joeddav/distilbert-base-uncased-go-emotions-student pre-trained model, verifying that the sentiment classification correctly categorized customer sentiments into positive, negative, or neutral.
- **Callback Feature:** Ensured correct callback responses after task completion.

After individual tests, we integrated the components, checked the logs for errors, and verified the output through MinIO UI for proper data storage.

Working System Overview

The system begins with a simple user interface (UI) where the user enters the Amazon product link they want analyzed. After entering the link, the status updates to "processing." The REST API receives the request and adds it as a task to a Redis message queue. The worker retrieves tasks from the queue and performs the following steps: scrapes reviews, summarizes them, analyzes sentiment, and identifies key topics. The processed data is stored in the MinIO database, and the system sends a callback with the data. Once the callback is received, the REST API updates the status to "completed," and the results are displayed on the website.

UI to submit the Amazon URL:



UI after the analysis on the product is complete:

Amazon Link Processor

Enter Amazon Link: Submit
Status: completed

Callback Details

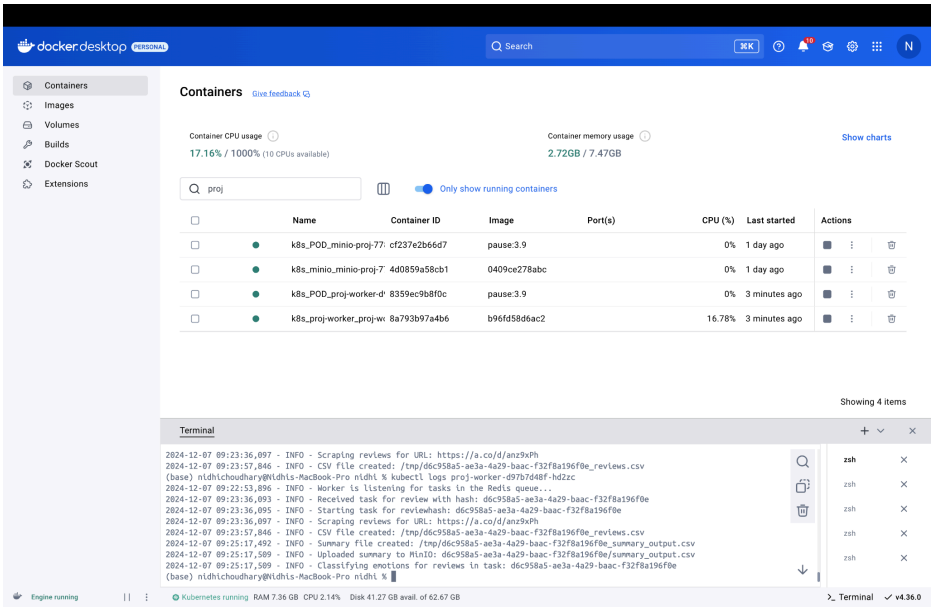
Product Name: Caseative for iPhone 14 Pro Case, Solid Color Curly Wave Frame Soft Compatible with iPhone Case (Green Blue,iPhone 14 Pro)

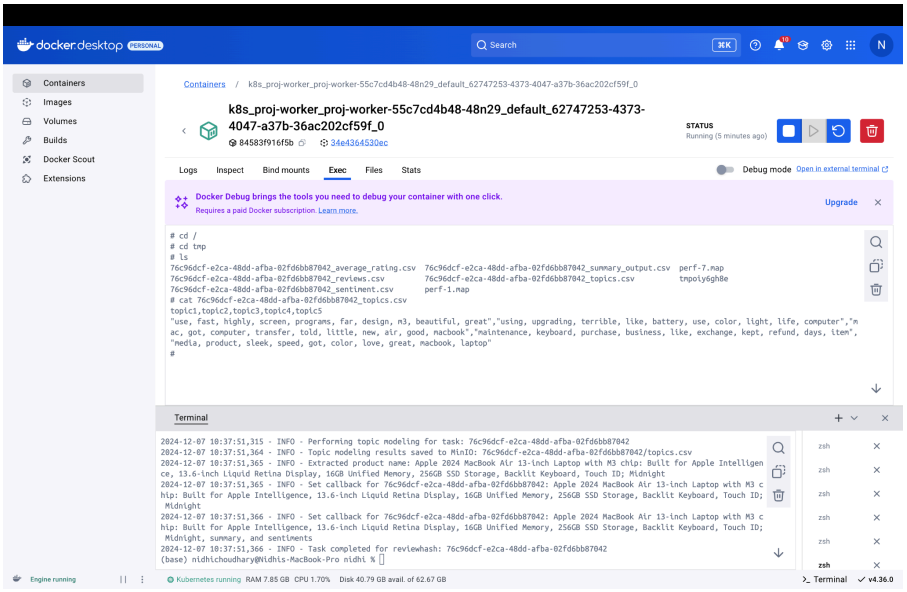
Summary: Love it but with this particular color it gets dirty quick! Would definitely buy again just in another color. Got so many compliments so that was a plus! Love how it protected my phone, wasn't bulky, & was ascetically pleasing! Also, you can't beat the price either!

Sentiments:

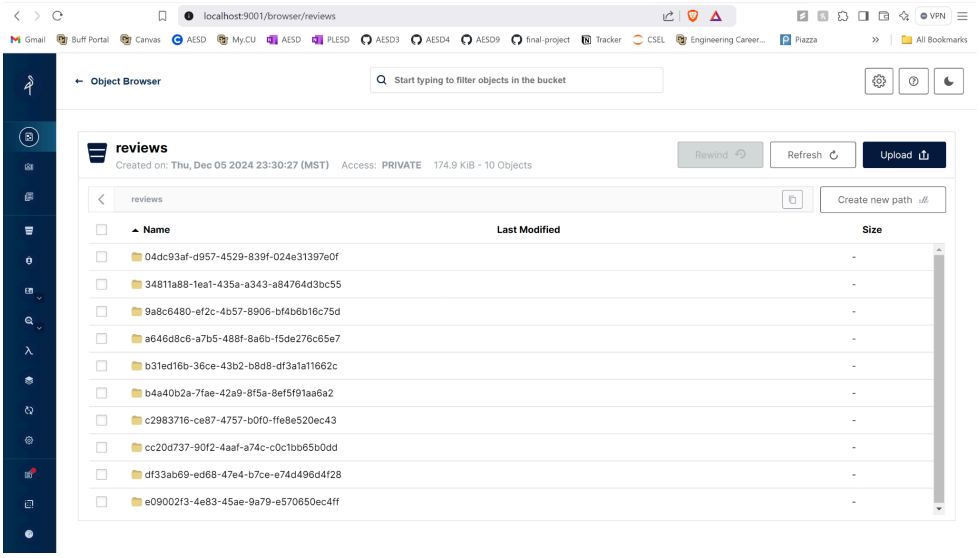
- Positive: 43.67010703897662%
- Neutral: 16.63010647495578%
- Negative: 39.6997864860676%

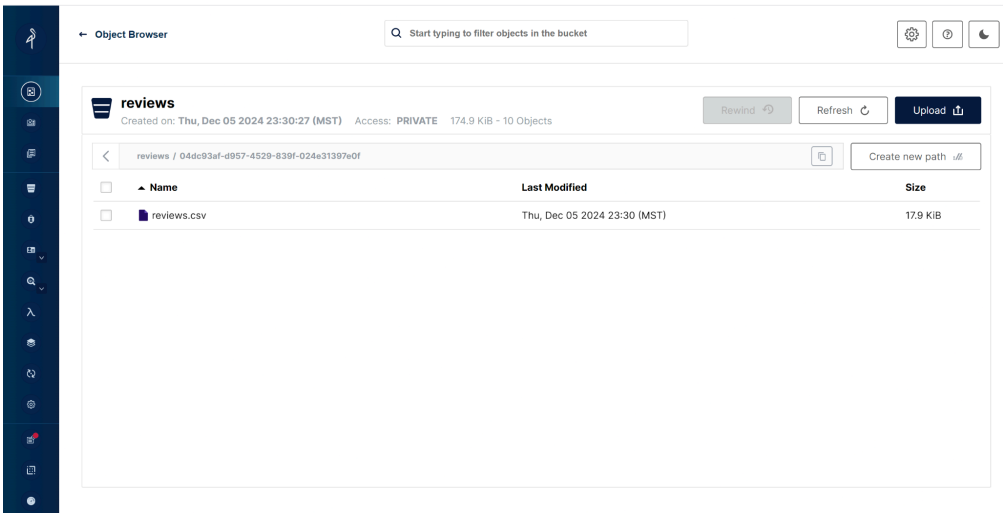
Debugging example by accessing the logs of the worker pod:





Example of review data stored in the MinIO bucket:





Workload Handling

The system can currently handle light workloads, processing a few hundred simple operations per second. This can be scaled as necessary.

Bottlenecks

- **Rate Limiting from Amazon:** Restrictions on the frequency and volume of requests to prevent overloading their servers. Currently, we extract reviews only from the first 5 pages to stay within these limits
  Future Work: Implementing request throttling and rotating proxy servers to bypass rate limits.

- **Fake Reviews:** Finding and removing unreliable or fake reviews.
  Future Work: Using advanced machine learning models to improve the detection and removal of fake reviews.
- **Models for Analysis:** Ensuring accurate and efficient text processing.
  OpenAI API: Limited by usage caps, requiring a paid version for continued analysis.
  Future Work: Improve performance with fine-tuning and faster models for real-time processing.

Capabilities and Limits of the Solution

The solution effectively handles tasks such as scraping Amazon product reviews, analyzing sentiment, summarizing feedback, and extracting key topics, all aimed at assisting businesses and consumers in making informed decisions. By utilizing pre-trained models like BART for summarization and DistilBERT for sentiment analysis, the system ensures accuracy and efficiency in processing text. Its modular design, supported by Docker for deployment and Redis for task distribution, makes it adaptable for various use cases. Processed data is stored in MinIO, offering a reliable storage solution for large-scale datasets.

However, there are some limitations to the solution. API rate limiting from Amazon can hinder the scraping process, requiring the implementation of strategies such as throttling or using proxies. Additionally, continued access to OpenAI's API for advanced analysis becomes restricted after a certain usage threshold, necessitating a paid plan for further usage. While the models used for analysis are powerful, processing large datasets can be slow due to their computational demands, and detecting fake reviews remains a challenge, especially when dealing with diverse and large volumes of reviews.

References

- https://www.geeksforgeeks.org/web-scraping-amazon-customer-reviews/
- https://arxiv.org/abs/1810.04805
- https://arxiv.org/abs/1910.13461
- https://huggingface.co/docs/transformers/index
- https://towardsdatascience.com/end-to-end-topic-modeling-in-python-latent-dirichlet-allocation-lda-35ce4ed6b3e0
- https://towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2
- https://realpython.com/python-redis/
- https://min.io/docs/kes/
- Class notes, slides, assignments