Movie Recommendation and Sentiment Analysis System

Project Documentation

Haonan Jin
Hang Yao
Bruna Barraquer Torres

June 19, 2025

Abstract

This document presents the design, methodology, and implementation details of a system focused on movie recommendation and sentiment analysis. It integrates structured and unstructured data using modern data engineering techniques. We justify our choices based on concepts covered in class, such as streaming analytics, NLP, recommendation systems, and hybrid data architectures. Additionally, high-level data pipeline diagrams and use case scenarios are included to support the practical relevance of the solution.

1 Domain and Context

We focus on the domain of **Movie Recommendation and Sentiment Analysis**, a high-impact area in the entertainment industry. Platforms such as Netflix, Amazon Prime, and Disney+ rely heavily on data-driven recommendation systems to enhance user experience and content discovery.

Our goal is to provide a more personalized and emotionally intelligent recommendation engine. We combine structured data (ratings, movie metadata, user preferences) with unstructured data (text reviews, social media posts) to offer nuanced insights using Natural Language Processing (NLP) techniques.

Problem Statement

How can we create a recommendation pipeline that not only suggests movies based on user ratings and behavior but also leverages sentiment analysis of movie reviews to refine those recommendations?

Use Cases

- **Use Case 1: Improving User Engagement** – Deliver recommendations based on watch history, sentiment, and context (e.g., mood, weather, time of day).
- **Use Case 2: Enhancing Content Discovery** – Enrich tagging and filtering by analyzing emotional tones from reviews.
- **Use Case 3: Predicting Market Trends** – Forecast audience sentiment and interest pre- and post-release using social media analysis.
- **Use Case 4: Supporting Film Production** – Help studios refine future projects using feedback-based insights from previous films.
- **Use Case 5: Leveraging Social Media Signals** – Integrate real-time trending data from platforms like TikTok and Reddit to improve recommendations and acquisition strategies.

2 Data Sources

For the Data Sources we needed to change the scope because we faced problems regarding some of the APIs that we wanted to connect. Specifically, Letterboxd API required a special access key which we were unable to obtain. Despite outreach efforts, we didn't receive a response. Also, Netflix, HBO, and Disney+ APIs were either restricted or not available publicly, which prevented us from accessing real-time content popularity data.

To maintain project continuity and meet our goals, we pivoted to simulating real-time data using Apache Kafka. Two main data streams were generated:

- **Simulated User Reviews and Ratings**: A Kafka producer generated synthetic but realistic reviews and star ratings, mimicking what would be collected from the sources mentiones.
- **Simulated Popularity Data**: Another Kafka stream emulated popularity metrics of movies, updated every "few minutes" (this value could change) ,to simulate a warm path (most-watched shows in the last 10 minutes).

This approach preserved the integrity of our architecture while allowing us to test streaming capabilities and sentiment-enhanced recommendations.

In the project we have used different types of data:

Structured Data

- **Movie data**: we obtain all kind of data from all movies (title, genres, cast,...) consuming data from IMBD and MovieLens APIs. The format is JSON and TSV files.

Semi-Structured Data

- **Movie Population**: We consume the from the boxoffice API addition information both structured and unstructured data (posters). Even though, the open-data option for this API, it only contains 42 different posters daily, so we will see if it is enough for the second part of the project. However, we can simulate images if necessary.

Unstructured Data

- **User Reviews and Comments and popularity metrics**: simulated using Kafka in a real-time format.

3 Justification of Architecture and Tools

Our architecture is based on a hybrid data pipeline strategy combining cold, warm, and hot paths:

Modeling Choices

We will use for baseline recommendations the data coming from MovieLens and IMBD. To improve, we will use the sentiment analysis based on user review polarity.

Technology Stack Justification

- **Apache Kafka**: we used kafka to simulate unstructured data which we did not have access to without a monetary cost.
- **Delta Lake**: ACID support and scalable storage of raw and processed data. We decided to use Delta Lake instead of Amazon Buckets or Google Cloud for the Landing Zone because it is an open software.

- **Redis**: It is used to simulate a temporal buffer for the warm path (e.g., most-watched movies in a hour).
- **Python & NLP Libraries**: For transformation, modeling, and text processing (e.g., NLTK, SpaCy).
- **Streamlit & Power BI**: For dashboards and end-user interfaces.
- **Airflow**: Manages batch workflows and ensures orchestration of ETL pipelines.

Storage (Landing Zone)

We chose Delta Lake for file storage because it efficiently handles large-scale, versioned data and supports ACID transactions. Its compatibility with both batch and streaming workflows made it well-suited for managing our static datasets, such as MovieLens. To organize the data pipeline, we implemented a two-stage storage system:

- A **Temporal** folder that holds raw, unprocessed data directly from the sources.
- A **Persistent** storage layer, where cleaned and transformed data is organized into subfolders based on data origin (e.g., ratings, reviews, metadata).

This structure improved clarity, simplified processing logic, and ensured a clear separation between raw input and processed datasets. It's important to note that data from Kafka (hot and warm paths) is currently **not stored**. The **hot path** is designed for immediate processing data is consumed and acted on in real time, so storage isn't necessary. As for the **warm path**, uses Redis, which acts like a temporal buffer to store the data that come from the simulation.

Orchestration

For orchestration and service management, we containerize our environment using **Docker**, which includes:

- **Apache Airflow** for managing and scheduling batch pipelines
- **Apache Kafka** for simulating streaming data
- **Redis** for managing state in the warm path, such as a temporal buffer to store the data

We also developed a monitoring tool called **UI.py**, which provides interfaces for:

- Batch ingestion
- Streaming ingestion for the hot path
- Streaming ingestion for the warm path
- Code quality checks
- CPU usage tracking

4 Proposed High-Level Architecture

Figure 1 shows our pipeline model combining multiple ingestion, transformation, and serving layers.

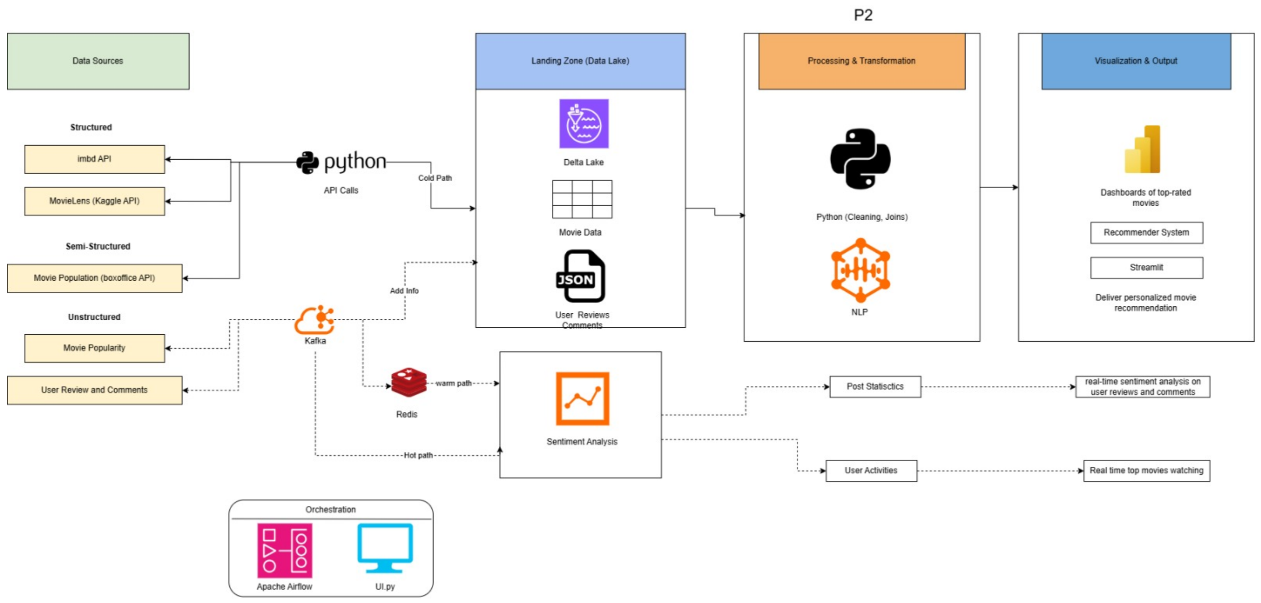


Figure 1: High-Level Data Pipeline Architecture

Data Ingestion Layer

- **Structured Data**: Retrieved via Python API calls from IMDb and MovieLens.
- **Semi-Structured Data**: Poster and box-office metadata are retrieved through the BoxOffice API.
- **Unstructured Data**: User reviews, comments, and simulated popularity signals are streamed in real time using Apache Kafka.

Landing Zone (Data Lake)

- **Delta Lake**: Used as centralized cloud storage to persist both structured movie data and unstructured user comments in JSON format.

Processing & NLP Layer

- **Python (Cleaning and Joins)**: Transforms raw data, joining metadata and user interactions.
- **NLP-based Sentiment Analysis**: Analyzes user reviews for emotional tone and aspect-based sentiment (e.g., plot, acting, cinematography).

Real-Time and Near Real-Time Layers (Redis & Kafka Integration)

- **Warm Path**: Data from Kafka is routed through Redis for fast retrieval of movie popularity and activity stats.
- **Hot Path**: Enables real-time sentiment analysis and updates on top-trending movies and user engagement.

Recommendation Engine

- **Collaborative Filtering**: Using MovieLens and IMBD data so we can provide initial recommendations.
- **Sentiment Re-ranking**: Refines suggestions using sentiment polarity and contextual cues (e.g., mood or review trends).

Serving and Visualization

- **Dashboards**: They will be built using Streamlit for live interaction and Power BI for high-level summaries.
- **UI Layer**: Delivers real-time, personalized movie recommendations, and sentiment visualizations.

5 Conclusion

This project aims to solve real-world challenges in movie recommendation and user engagement. We have justified our modeling approaches and architectural choices based on scalability, data diversity, and analytical goals. Our system architecture supports both batch and real-time flows, enabling robust insights for both consumers and business stakeholders in the entertainment ecosystem.

6 Annex

6.1 GitHub

We attach here the link to the public GitHub
    `https://github.com/brunabarraquer/BDM`

6.2 Instructions to Execute Apache Airflow

To run Apache Airflow, Docker must be active. Then, open your browser and go to `http://localhost:8080/home`. You should see the following interface (2):

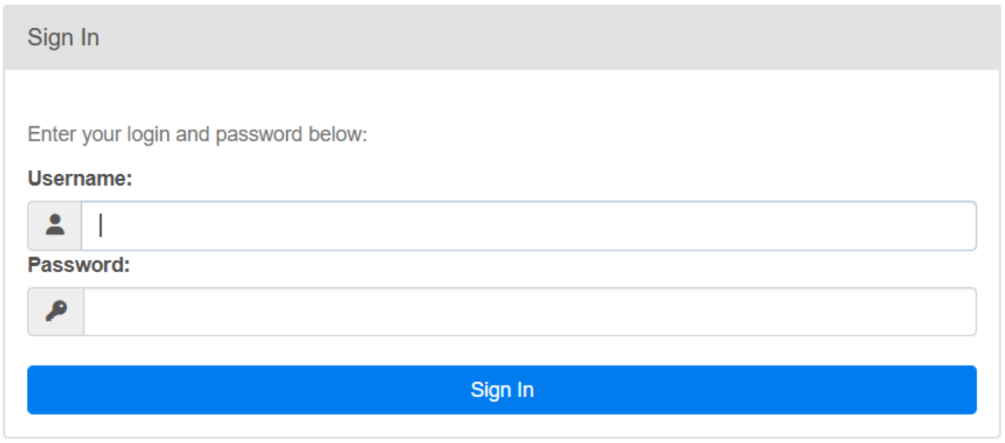
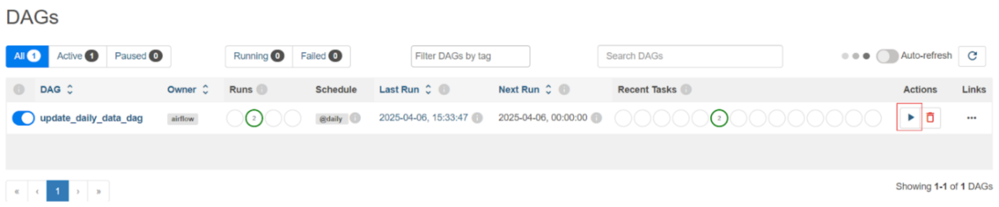


Figure 2: Apache Airflow Web Interface

The username and password is "admin" and "admin". When you get in you will a see the next screen (3) :



	DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
	update_daily_data_dag	airflow	2	@daily	2025-04-06, 15:33:47	2025-04-06, 00:00:00	2		...

Showing 1-1 of 1 DAGs

Figure 3: DAGs

Then you need to click on the play button to run the execution for our batch ingestion. With this, the execution for batch ingestion will start, and will start downloading the files of the different data sources and moved to the folders that we have create (probably this will stay minutes).