Final Project Report

Author: Nicholas Swisher
Course: CSPB-3112

Introduction

This project aimed to gain experience with industry-relevant tools and build a portfolio worthy project. This allowed me to practice workflows that mirror real industry systems. The system chosen for the project was a recommendation engine, found in many social media platforms, streaming services, shopping platforms, and similar services. The system accepts one or more movie selections, builds a user taste profile by averaging their weighted metadata-based vectors, and ranks recommendations using cosine similarity. The metadata features included keywords, genres, collection membership, and production companies.

Specific goals included:

- Build a modularized Python data pipeline
- Design and utilize a SQLite database for data storage
- Create a REST API using FastAPI with clean endpoints
- Build a minimal UI using Jinja2 Templates
- Deploy the finished product

Background

Throughout my experience in the CSPB program, I have developed a strong appreciation for backend software development. I especially enjoyed projects that involved designing system architecture, working with APIs, building data pipelines, and integrating databases. Since I plan to pursue backend engineering, I chose a recommendation system for this project to align with those interests.

My technology choices were made to build an end-to-end, industry-style recommendation system while expanding my skillset. Python was selected for its strengths in data engineering tasks and its vast libraries for preprocessing and feature extraction. SQLite was chosen because the movie metadata has a clear relational structure (ex: movies can have one or more keywords) and using it expanded my experience with relational databases. FastAPI was selected because I wanted experience building my own REST API and its ease of use made it a strong fit for this project. Frontend design was not the focus of this project, so the UI was intentionally minimal and relied on basic web technologies.

Overall, the tech stack supported the project goals and enabled the construction of a backend-focused, full stack system.

Methodology, Materials and Methods

Many of the tools used in this project were either new to me or ones I had a high-level understanding of, but lacking the depth needed to apply them to the project. My general strategy in strengthening understanding of these topics was to use high-level AI prompting to give me an overview and the tools general usage, followed by more detailed questions pertaining to my use case. Next, I read official documentation for verification of inputs and outputs for each of the tools, as well as for viewing their quick start or tutorial sections. In some cases, I also viewed YouTube breakdowns of the tools, completed LinkedIn Learning modules, or referenced web resources like GeeksforGeeks and StackOverflow.

I have found that before diving into the often dense and specific documentation it is beneficial to have an overview of the library, its uses, and an idea of the specific items that I may like to use from it. This focused learning strategy worked well and saved time in not having to comb through the many features of a library that were not relevant to my application.

In my methods I subscribed to an iterative approach that consisted of constructing an MVP, finding its limitations or errors and refactoring the components necessary to fix or improve the areas of strain. This strategy exposed real limitations such as slow queries of the database, which were later fixed with indexing key items and only storing items necessary to the current process without considering broader scope. It also enabled me to find errors in dataset cleaning and transformation with printing spot checks.

Results

Through this project I learned several core techniques. Firstly, I learned about text vectorization and encoding of features through my data transformation section. This involved learning about and utilizing scikit-learn's feature extraction TF-IDF Vectorizer, which transforms the data into a weighted matrix that upweights rarer items and downweights very frequent ones. Next, I learned about encoding of categorical data using scikit-learn's MultiLabelBinarizer. This is used to make individual feature columns for each of the items, showing presence and allowing for multiple memberships. This was leveraged for features like genres and production companies. I also gained understanding of scikit-learn's OneHotEncoder, which creates a binary matrix also indicating presence and being used for items that only have single membership. This was applied to the collection membership feature. I also learned about REST APIs, how to structure and build endpoints and core functions of the endpoints (GET, POST, PUT, DELETE).

I know I learned these topics as I have implemented some of the text encoding tools in other projects since learning about them. I used the OneHotEncoder for a Data mining project to create categorical features for computing Pearson correlation coefficients on the dataset. For the tools not used in other applications, I now can explain how and when to use them and understand what the results of their usage would look like.

The assessments for this project included a Precision@5 metric to ensure that the recommendations being served were relevant. Many of the movie recommendations achieved a precision score of 0.8 or higher, but a few combinations were not. This qualifies as meeting the assessments I set forth in proving that the system functions as intended. It is also likely that with a larger, more recent dataset this would be closer to all movie recommendations scoring highly. With this evaluation metric, I was prompted to move away from a fixed number of recommendations and only serve recommendations meeting a minimum similarity score.

Reflection

Overall, I feel like all goals of the project were met. I was able to build an end-to-end system mirroring systems used in industry. I expanded my knowledge in backend design, data engineering, relational databases, basic web frameworks, and learned about REST APIs. This combined into a portfolio-worthy project, where I am satisfied with the end result.

The learning process used throughout development, coupled with an iterative approach, as well as working in an organized structure to construct the project
(data import -> preprocessing -> storage -> transformation -> recommender logic -> endpoints -> UI ->

validation)
is what allowed me to reach the main goals of the project.

In the end:

- The modularized Python pipeline enabled easy integration with the API
- The SQLite database was well structured and supported efficient queries
- The FastAPI endpoints served recommendations, cleanly integrating the frontend and backend
- The user interface allows interaction with the recommendation system in a user-friendly format
- The project was deployed for readily available user interaction
- The project is industry-relevant and portfolio-worthy

Conclusion

This project heavily focused on the backend system and was only made into a full-stack program to demonstrate that it is within my capabilities and for portfolio viewing purposes. Looking back on my school experience I always enjoyed courses like Data Structures, Databases, Computer Systems, and when working in groups I always volunteered to handle the tasks on the backend, even though at the time I didn't fully understand the implications of that. This project as well as some others from this semester really highlighted that these are the topics and types of problems I prefer working on. I now have a much clearer goal in my career in wanting to pursue backend work, where previously I would have generically said I want to be a software developer. In the future I would like to expand my knowledge into cloud infrastructure, authentication, scalable systems, and deeper database proficiency. I am planning to expand a small scale event management platform project using these topics to continue building upon my skills.

References

Scikit-Learn Documentation
https://scikit-learn.org/stable/modules/feature_extraction.html

FastAPI Documentation
https://fastapi.tiangolo.com/#example

Pandas Documentation
https://pandas.pydata.org/docs/user_guide/missing_data.html

Scipy Documentation
https://docs.scipy.org/doc//scipy-1.16.2/reference/generated/scipy.sparse.hstack.html.

LinkedIn Learning
https://www.linkedin.com/learning/build-rest-apis-with-fastapi/what-is-fastapi?

ChatGPT
https://www.chatgpt.com/

GeeksforGeeks
https://www.geeksforgeeks.org/nlp/different-methods-to-find-document-similarity/

Stack Overflow
https://stackoverflow.com/questions/28403744/why-should-json-loads-be-preferred-to-ast-literal-eval-

for-parsing-json?utm_source=chatgpt.com

YouTube Scikit-Learn Intro
https://www.youtube.com/watch?v=0B5eIE_1vpU&t=657s

Dataset
https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=movies_metadata.csv

GitHub
https://github.com/swish0621/Content-Based-Recommendation-System.git

Deployed System
https://content-based-recommendation-system-ofm5.onrender.com/