Film and TV Recommendations: A New Approach

Executive Summary

In this report, we define and propose our initial framework for a new recommendation engine for movie reviews. Unlike previous recommendation systems in this space, our proposal provides tailored, data science-backed recommendations that have the potential to change the way the current entertainment offerings interact with consumers.

To accomplish these business requirements, we offer a high-level overview of system and process needs, examining the pros and cons of available alternatives.

Business Processes

Our recommendation system will improve upon existing recommendation systems by utilizing deep learning neural networks to make more accurate recommendations, tailored to each user's specific interests. Rather than simply relying on user input, or aggregating ratings across many users for a particular film and using this average rating as a recommendation, our system will allow complex connections to be made between users, ratings, and the films themselves, providing far more accurate ratings to each individual user. Our target market is not only direct-to-consumer, but also business-to-business, as current companies in the film recommendation space (e.g. Netflix) could find this method highly desirable.

From a business requirement standpoint, our system will need to both generate and deliver recommendations in a high volume environment. It will need to be optimized for both analytics and end use, with potentially millions of users and billions of connections between those users and the films that might be recommended to them. It will need to ingest data from both IMDb and MovieLens.org, and store user inputs and data.

Ultimately, it will need the ability to feed these connections to multiple, disparate sources, from user-focused app and website solutions to ingestion into systems managed by external, business customers. As such, a key business requirement is that the system be reliable, scalable, and maintainable, as described in the next section.

System Requirements

As mentioned above, reliability is a key business requirement for this system, due to both its high potential volume and, especially, the fact that the business model relies on external customers. External customers such as Netflix require an always-on, fault tolerant system due to their global nature and substantial traffic load; as such, we will need to be resilient against hardware, software, and human-error faults, which we will accomplish via redundancy, production analysis of software performance, and clearly defined telemetry, respectively.

Another important consideration in this system is evolvability. We are clearly designing a prototype, and, by the very nature of data science in general and neural networks specifically, new data will likely lead to new results and new business requirements. The system needs to be robust against additional requirements and features.

Data Structure/Query Selections

While there are many options for data structures and related querying optimization, for our purposes here, we will examine three commonly used solutions: relational models, document models, and graph models.

Relational models are likely the most familiar of the group-- SQL and relational databases are actively used in most companies with “true” database systems in production, and offer several advantages. Joins are relatively cheap, computationally, and the schemas underpinning the database are generally familiar to most data scientists. Querying can be easier to navigate, as rows can be parsed, split, and joined to other tables with fairly low overhead.

The downsides to the relational model become apparent when we think about the relationships likely to be present in our data- namely, many-to-many relationships. By the nature of our network model, each user (node) will be connected to many films, as well as many other users. Additionally, many users will connect to one another. Attempting to force these relationships into traditional relational models can create unnecessary development overhead at the beginning of the project, and increase costs and complexity as the project matures. Finally, the strict schema structures of relational models do not conform to our business requirement evolvability-- as our business requirement s evolve, we certainly do not want to be tied to a strict schema.

For these reasons, we do not recommend a relational model for this project.

Document models also exhibit issues with regards to the data relationships we are likely to see in this deployment. While document models have many benefits, all of them are quickly overshadowed due to the many-to-many relationships we are likely to see with our data. Document models offer more schema flexibility, but our project cannot be neatly fit into a document model-- there will often be instances where partial queries and complex joins are needed, and the document model does not serve those purposes well.

Our third option, the graph model, is our recommendation for this project. Graph models can handle many-to-many relationships better than our above options. While relational models are an option at lower levels of complexity, our stated goal of reaching millions of users and large organizations such as Netflix mean that it is more than likely our levels of complexity with outstrip acceptable levels of efficiency than can be expected from that model.

Conceptually, our data will create many nodes and arcs, and the graph model is a natural way to visualize and structure this type of data. Finally, it presents a flexible schema that can be easily manipulated with a declarative language.

Analytic Processing

We also need to prepare ourselves for the costs and effort necessary to both deliver the results of our recommendation engine while also analyzing its effectiveness. To accomplish these duel goals, we recommend creating separate development and analytic environments.

Our analytic environment will utilize column-oriented storage. This will allow us to speed queries to the system for analytic purposes, increase compression potential for space-saving purposes, and maintain a structure that still lends itself to self-service analytics. Comparing this with the cube/OLAP model, which offers speed at the expense of flexibility, we believe this option will offer the greatest functionality to our analysts.

Conclusion

In conclusion, we have identified business needs, usages, and requirements, while also laying out the justification for our chosen data structures and processing requirements. Future reports will delve into greater detail on the specifics required to meet this challenge and compete with industry-leading recommendation engines on the market today.

References

Kleppmann, M. (2017.) *Designing Data-Intensive Applications: The Big Ideas behind Reliable, Scalable, and Maintainable Systems.* Sebastopol, Calif.: O’Reilly. [ISBN-13: 978-1449373320]