Sarah Martin

Northwestern University, School of Professional Studies

Foundations of Data Engineering, Professor Lance Levenson

February 22, 2019

Project 1: Business Processes and Requirements Definition for a Movie Recommender Engine

Section 1: Management Overview and Recommendations

The task at hand is to define general requirements for the technical system and related business processes for a new movie recommender engine. Ideas for technical solutions that might help fulfill these requirements are also occasionally suggested. However, additional clarity around the product's needs would be required to align on any final technical solutions for adoption.

To build and support this recommender engine, data should be collected from three sources, and technical solutions to properly store and analyze this data would need to be built and/or bought. Components would also have to be built that would allow users to interact with our product by submitting their own movie preferences, requesting recommendations from our engine, having their data applied to our model to generate predictions, and receiving results.

The next section goes into more detail about the data that could be extracted from each of the data sources, and our needs for storing the data. The following section covers our needs for tools to use in analyzing the data, fitting models, and producing user-specific recommendations. The last section touches on our relative computing needs to support the above functions and potential solutions for those computing needs.

Section 2: Data and Database Software

The movie recommender engine currently requires input data from three sources: The Internet Movie Database (IMDb) owned by Amazon, MovieLens.org hosted by GroupLens Research (GroupLens), a group of students, staff, and visitors engaged in social computing research at the University of Minnesota, and users of the recommender engine themselves.

Section 2.1: IMDb Data

While IMDb does not provide an API for public queries, it offers seven daily updated downloadable compressed tab-separated UTF-8 encoded plain text files that contain much of the data found on IMDb via this link: https://datasets.imdbws.com/. The data is listed as available for personal and non-commercial uses only per their Non-Commercial Licensing ("Can I Use IMDb Data in My Software?"). Immediate research is required to determine if we can publish modeling results leveraging the data. If we eventually intend to monetize our recommender engine, we must first determine the viability of Commercial licensing with IMDb ("Content Licensing"). If Commercial licensing is an option, technical and legal resources will be required to coordinate with IMDb to develop and maintain a relationship to ensure continued use of their data.

We will need to explore and manipulate the IMDb data at will and feed it into our analytics tooling of choice for Neural Network encoding. As such the data should be stored in-house. Along with command-line interface tools and a Java-based GUI that allows searches and displays of the information provided, IMDb offers a Python package, "IMDbPY", that can process and load the files into a number of different SQL-based databases. Thus, an initial assessment suggests the IMDbPY package may be well suited for accessing this data and feeding it into a SQL-based database.

A technical resource[1] should be put in charge of executing data acquisition, pre-processing, and batch loading into data storage. This resource must easily communicate with those managing any IMDb relationship *and* with the modelers to ensure that expectations regarding the data are in alignment.

Section 2.2: MovieLens.org Data

GroupLens offers downloads of 7 UTF-8 encoded csv structured zipped files sampling the

[1] Resource may always refer to a single individual or a team of people

MovieLens data. Each dataset comes with a README file outlining proper uses of the data. Specifically, "the user may not use this information for any commercial or revenue-bearing purposes without first obtaining permission from a faculty member of the GroupLens Research Project." ("GroupLen ml-latest-small-README") If we eventually intend to monetize our recommender engine, immediate research is required to assess the viability of using the MovieLens.org data. If viable, resources will again be required to coordinate with GroupLens to develop and maintain a relationship to ensure continued use of their data.

Similar to the system requirements for use of the IMDb data, our data-flow system must be able to retrieve the MovieLens files and load the data into storage. A technical resource will have to execute the MovieLens data acquisition, any pre-processing of the data, and batch loading it into storage. This resource could easily be the same resource handling the IMDb data acquisition and also, should have direct lines of communication with technical resources at GroupLens. Since it is a small research organization, developing the relationship with GroupLens will likely be far less demanding than doing so with IMDb, a subsidiary of one of the most powerful companies in the world.

Section 2.3: User Input Data

The last data source required for the recommender engine is input from users. For a prototype, any API can serve this function. To make the product user friendly, we should update existing users' movie preferences every time input is received. Thus, user input should be delivered to a server that will load the data into storage for future use.

A technical resource should develop and monitor the functionality of the user input API ensuring the condition of the data arriving at our server is as expected and that it is successful delivered into data storage. This resource should have direct lines of communication with the modelers. Ideally, the API should have some data checks in place and be able to automatically abandon data ingestion and produce appropriate error messages to relieve burden on our technical resources and the rest of our back-end system when faulty data is submitted.

Section 2.4: Storage Systems

Most of our input data is already structured in relational-friendly tables, so a relational (SQL) data model is a compelling option for storage needs. The exact database to use depends on existing infrastructure, technical expertise required and available, and cost. SQLite, a free serverless (cloud-based) database, would keep costs low assuming it can handle our load appropriately. Since the volume of writes will likely be higher than the volume of reads, another option is MySQL which offers a log-structured (LSM) alternative to its default B-tree solution. MySQL would give us the freedom to test LSM versus B-tree indexing within the confines of a relational. Many engineers are familiar with SQLite and MySQL making maintainability easier to fulfill.

Regardless of the products used, a database administrator will need to monitor database software, track issues, instantiate data checks, and monitor read/write volume and performance to ensure database tuning parameters are set properly.

Section 3: Analytics and Modeling Software

To *develop* the recommender engine, analysis would require large infrequent queries of the data, an ideal use case for data warehousing. The IMDb data is particularly predisposed to a star-schema structure. As most data warehousing solutions are not cheap, several unknowns -- including the size of queries for analysis, their frequency, our database read performance, and the impact that slow database read times have on engine development -- need answering before deciding to purchase a data warehouse solution.

To execute analyses such as model fitting and prediction generation, other analytical tooling is required. Python is free, well known to engineers and analysists alike, and may be used in other areas of the data pipeline. To leverage Python, or any scripting language that supports neural networks, whenever a recommendation request is submitted, the back-end server would process any new data submitted with the request, consolidate that data with the user's existing data sourced from storage, and run a Python script leveraging the fitted model (also from data storage) on that data. There are also several options for Python development environments for data exploration and model fitting, such as Jupyter Notebooks.

The modelers should be in charge of conducting model validation and tuning to ensure optimal accuracy of the engine. This resource should have direct communication with decision makers in case of model failure or required changes to the prototype.

Section 4: Computing and Communications Systems

The computing infrastructure for this project will have to support data storage (and maybe warehousing), ETL process flows, model development, and model predictions. With over 600,000 to 700,000 movie titles on IMDb plus upwards of 20 million movie ratings applied to 27,000 movies in a single file from MovieLens, storing and exploring the data, and fitting neural network models to it will take a great deal of computing power, requiring distributed and/or cloud computing. Since neural network fitting is notoriously slow but will not be a continuous process, the elasticity of a cloud system would allow us to dynamically and efficiently scale computing to accommodate model fitting when it is being conducted. To guide our understanding of the computing power required for prediction generation (post model fitting), the average response time for delivering recommendations upon user request for the prototype should be less than 1 second.

Works Cited

"Can I Use IMDb Data in My Software?" IMDb, IMDb.com, help.imdb.com/article/imdb/general-information/can-i-use-imdb-data-in-my-software/G5JTRESSHJBBHTGX?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=3aefe545-f8d3-4562-976a-e5eb47d1bb18&pf_rd_r=A4RHXCC91B96D7392C05&pf_rd_s=center-1&pf_rd_t=60601&pf_rd_i=interfaces&ref_=fea_mn_lk1#.

Choudhury, Sid. "A Busy Developer's Guide to Database Storage Engines - The Basics." YugaByte DB, 15 Aug. 2018, blog.yugabyte.com/a-busy-developers-guide-to-database-storage-engines-the-basics/.

"Content Licensing." IMDb, IMDb.com, www.imdb.com/licensing/?ref_=helpms_ih_gi_license.

Designing Data-Intensive Applications: the Big Ideas behind Reliable, Scalable, and Maintainable Systems." Martin Kleppmann, O'Reilly Media, 2018, pp. 3–107.

"GroupLens." GroupLens, grouplens.org/.

GroupLens. "ml-latest-small-README." *MovieLens Datasets*. http://files.grouplens.org/datasets/movielens/ml-latest-small-README.html.

Hooks, Ivy. Requirements Experts, reqexperts.com/resources/requirements-articles/articles-what-is-the-difference/.

"IMDb Datasets." IMDb, IMDb.com, www.imdb.com/interfaces/.

"MovieLens." GroupLens, 14 Jan. 2019, grouplens.org/datasets/movielens/.