Software Requirements Specification for a Movie Recommender System

Seyed Ali Mousavi

April 15, 2024

Contents

Revision History

Date	Version	Notes
2024-02-05	1.0	Initial draft
2024-04-15	2.0	Final release

This template is intended for use by CAS 741. For CAS 741 the template should be used exactly as given, except the Reflection Appendix can be deleted. For the capstone course it is a source of ideas, but shouldn't be followed exactly. The exception is the reflection appendix. All capstone SRS documents should have a refelection appendix.

1 Reference Material

This section records information for easy reference.

1.1 Abbreviations and Acronyms

symbol	**description**
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
TM	Theoretical Model
KNN	K-nearest neighbor algorithm

2 Introduction

Most internet products we use today are powered by recommender systems. YouTube, Netflix, Amazon, Pinterest, and a long list of other internet products all rely on recommender systems to filter millions of contents and make personalized recommendations to their users. In this project, we will start from scratch and walk through the process of how to prototype a minimum-viable movie recommender using the Collaborative filtering method.

The dataset that I'm working with is MovieLens, one of the most common datasets that is available on the internet for building a Recommender System. The version of the dataset that I'm working with (100K) contains 100,000 anonymous ratings of approximately 1,700 movies made by 943 MovieLens users who joined MovieLens in 2000. ratings are in range 1 to 5. (5 for the best rating)

2.1 Purpose of Document

This document serves as a crucial foundation for our software development project. Its primary purpose is to provide a detailed and unambiguous description of what the movie recommender is expected to accomplish, outlining the requirements that need to be met for successful development. This document is typically created during the early stages of the software development life cycle and acts as a reference for all stakeholders involved in the project, including developers, testers, project managers, and users.

2.2 Scope of Requirements

The purpose of a recommender system is to suggest relevant items to users. To achieve this task, there exist two major categories of methods: collaborative filtering methods and content-based methods. This project will specifically employ a collaborative method, while other methodologies are explored for readers with a keen interest in the topic.

In the collaborative filtering method, we don't use additional information about users like age, gender, job, etc., and/or items(movies) like genre, and movie length in this project.

2.3 Characteristics of Intended Reader

Readers of this document should have proficiency in general mathematics (university level). familiarity with KNN(K-nearest neighbor) method is recommended.

2.4 Organization of Document

The document initiates with an overview of itself and recommender systems, encompassing (a) the purpose of the recommender system, (b) the project's scope, and (c) the intended reader's characteristics. Following this introduction, subsequent sections delve into the system with progressively detailed explanations. These sections include:

- **General System Description**: Basic details regarding the system are provided, outlining the interfaces connecting the system with its environment, delineating user characteristics, and enumerating system constraints.
- **Specific System Description**: In this section, the problem description is initially presented, offering a broad overview of the issue to be addressed. Following this, the solution characteristics specification is outlined, encompassing assumptions, theories, definitions, and ultimately the instance models.
- **Requirements**: In this section, the functional requirements, which outline the business tasks that the software must accomplish, are detailed alongside the nonfunctional requirements, which specify the desired qualities the software should demonstrate.
- **Likely changes**: This section outlines expected alterations to requirements, serving as guidance for design and implementation decisions.
- **Unlikely Changes**: This section outlines expected alterations to requirements, serving as guidance for design and implementation decisions.

3 General System Description

3.1 System Context

Our movie recommender system utilizes a collaborative filtering method. Collaborative methods for recommender systems are methods that are based solely on the past interactions recorded between users and items to produce new recommendations. These interactions are stored in the so-called "user-item interactions matrix". The rows in the matrix represent users indices and the columns represent items indices (movies, in our case). So the entry ij of the matrix represents the rating of the user i for the movie j (an integer number between 1 to 5). The main idea that rules collaborative methods is that these past user-item interactions are sufficient to produce good enough recommendations. After receiving a specified user of interest (target user index) for whom we want to make recommendations, the ANN or KNN method algorithm is executed to generate the top recommendations. Figure 1 illustrates a concise diagram of this:

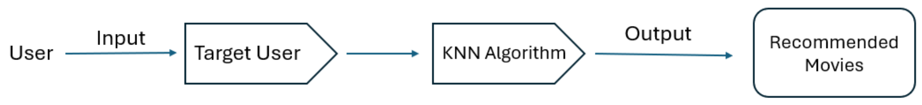


Figure 1: System context

There are 2 main types of collaborative filtering algorithms: 1)User-user method and 2)Item-item method. The main characteristics of user-user and item-item approaches is that they use only information from the user-item interaction matrix and they assume no model to produce new recommendations. We only implement user-user approach in this project.

User-user Collaborative filtering: To generate a new recommendation for a user, the user-user method employs a strategy that broadly seeks out users sharing the most similar interaction profiles (nearest neighbors). This method then suggests items that are highly favored among these identified neighbors and are also ”new” to our target user. This approach is deemed ”user-centered” as it characterizes users based on their interactions with items and calculates distances between them accordingly.

To illustrate, suppose we aim to provide a recommendation for a specific user. Initially, each user’s interactions with various items are represented by a vector (akin to ”its row” in the interaction matrix). Subsequently, a measure of ”similarity” is computed between our target user and all other users. This similarity metric is designed so that users exhibiting similar interactions with the same items are regarded as close. After computing similarities with every user, the algorithm retains the k-nearest-neighbors to our target user and proceeds to suggest the most popular items among this group (while considering only those items not yet interacted with by our reference user).

- User Responsibilities:
  - Provide correctly typed inputs: A user index within the range of users and a User-Item interaction matrix with integer entries in the range 1 to 5.
- Movie Recommender Responsibilities:
  - Detect and report input data type mismatch
  - Identify and report input constraint violations
  - Produce recommendations as output

3.2 User Characteristics

The the end user of the recommender system is only expected to provide the index of the target user (to whom the user wants to make movie recommendations). Hence there is no scientific requirement for the user.

4 Specific System Description

4.1 Problem Description

This project intends to build a recommender system based on the past recordings of users and items.

4.1.1 Terminology and Definitions

The following may be used in the subsequent sections:

- K-Nearest Neighbor(KNN): is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point
- Target User (or User of Interest): Someone whom the movie recommender is to make movie recommendations for. The target user's index number must already be in the user-item interaction matrix

4.1.2 Goal Statements

There are 2 main types of collaborative filtering algorithms: 1)User-user method and 2)Item-item method

- GS1: Using the target user index as input, the system generates recommended movies utilizing a user-user-based movie recommender.
- GS2: Presenting the evaluation of our approach using various metrics.

4.2 Solution Characteristics Specification

4.2.1 Assumptions

- A1: In real-world applications, higher accuracy in ratings is preferred. However, in our scenario, the ratings are integers rather than floating-point numbers.
- A2: input values are accurately typed and within the specified constraints. (index of the target user must be in the range of user indices)
- A3: The users' preferences remain consistent over time. In simpler terms, once they rate a movie, their ratings for it remain unchanged over time.

4.2.2 Theoretical Models

This section focuses on the general equations and laws that Movie Recommender is based on.

RefName: **TM1**

Label: Euclidean Similarity

Equation: $d(u,v) = \sqrt{\sum_{i=1}^{n}(u_i - v_i)^2}$

Description: Euclidean similarity, also known as Euclidean distance, is a measure of similarity between two vectors in a multidimensional space. It calculates the distance between two points in the space, where a smaller distance implies greater similarity. Euclidean similarity measures the actual geometric distance between the points represented by the vectors.

Notes: In the context of user-user collaborative filtering, you can use Euclidean similarity to measure the distance between users based on their ratings for items. Users with smaller Euclidean distances are considered more similar to each other, while users with larger distances are less similar..

Source: https://en.wikipedia.org/wiki/Euclidean$_{d}istance$

Ref. By: IM1

Preconditions for TM1: None

Derivation for TM1: Not Applicable

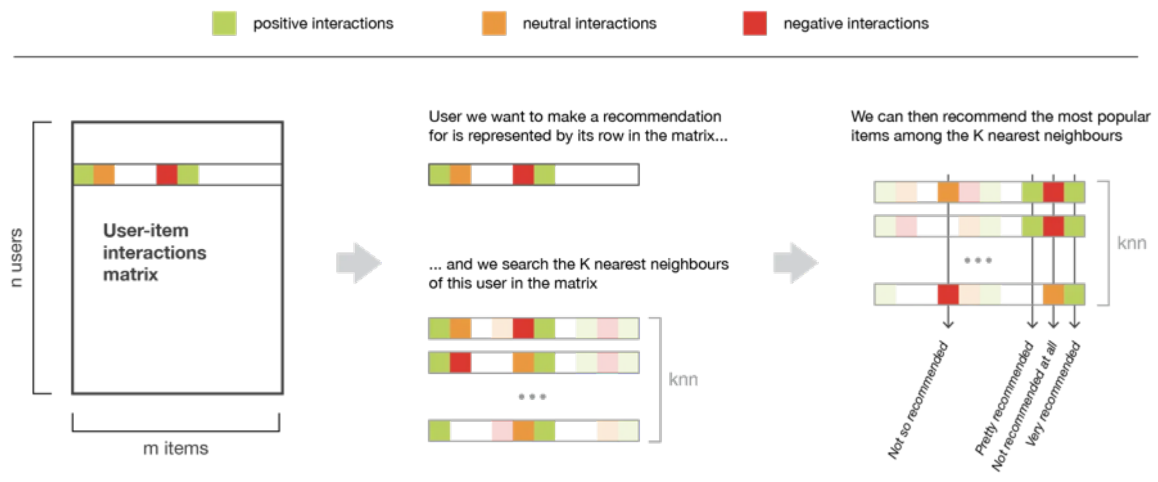
4.2.3 General Definitions

This section collects the laws and equations that will be used in building the instance models.

Number	GD1
Label	**User-Item interaction matrix**
Type	**pandas DataFrame**
Symbol	$X_{m \times n}$ where m is number of users and n is number of items
Description	Collaborative methods for recommender systems are methods that are based solely on the past interactions recorded between users and items to produce new recommendations. These interactions are stored in the so-called "user-item interactions matrix". The rows in the matrix represent users indices and the columns represent items indices (movies, in our case). So the entry ij of the matrix represents the rating of the user i for the movie j (an integer number between 1 to 5)
Source	TowardsDataScience/UserItemInteractionMatrix

4.2.4 Instance Models

This section transforms the problem defined in Section 4.1 into one which is expressed in mathematical terms. It uses concrete symbols defined in Section **??** to replace the abstract symbols in the models identified in Sections 4.2.2 and 4.2.3.



positive interactions
neutral interactions
negative interactions
n users
User-item interactions matrix
m items
User we want to make a recommendation for is represented by its row in the matrix...
... and we search the K nearest neighbours of this user in the matrix
knn
We can then recommend the most popular items among the K nearest neighbours
knn
Not so recommended
Pretty recommended
Not recommended at all
Very recommended

Figure 2: A schematic of Movie Recommender Collaborative model

Number	IM1
Label	**The Movie Recommender model**
Input	User-item interaction matrix
Output	Recommended items (Movies)
Description	Assume that we want to make a recommendation for a given user. First, every user can be represented by its vector of interactions with the different items ("its line" in the interaction matrix). Then, we can compute some kind of "similarity" between our user of interest and every other users. That similarity measure is such that two users with similar interactions on the same items should be considered as being close. Once similarities to every users have been computed, we can keep the k-nearest-neighbours to our user and then suggest the most popular items among them (only looking at the items that our reference user has not interacted with yet).
Sources	TowardsDataScience/RecommenderSystems
Ref. By	-

5 Requirements

This section provides the functional requirements, the business tasks that the software is expected to complete, and the nonfunctional requirements, the qualities that the software is expected to exhibit.

5.1 Functional Requirements

R1: Check the validity of the input data. Like the size and the format of ratings.

R2: The system shall calculate users close to the target user in terms of vector similarities using KNN.

R3: The system shall calculate popular items among close users and provide it as a list of recommended movies.

5.2 Nonfunctional Requirements

NFR1: **Accuracy** The level of accuracy for Movie Recommender will be specified in Verification and Validation Plan.

NFR2: **Understandability** The codebase must be readable and understandable for facilitating maintenance, collaboration among developers, and onboarding of new team members.

NFR3: **Maintainability** the end user of the recommender system is only expected to provide the user of interest index number. Hence there is no scientific requirement for the user.

6 Likely Changes

LC1: Maybe, I use some additional information about users and/or movies.

7 Unlikely Changes

None

8 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that are marked with an "X" may have to be modified as well. Table 1 shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table 2 shows the dependencies of instance models, requirements, and data constraints on each other. Table 3 shows the dependencies of theoretical models, general definitions, data definitions, instance models, and likely changes on the assumptions.

	TM1	GD1	IM1
TM1			
GD1			
IM1	X	X	

Table 1: Traceability Matrix Showing the Connections Between Items of Different Sections

	TM1	GD1	IM1
R1		X	
R2	X		X
R3		X	X

Table 2: Traceability Matrix Showing the Connections Between Requirements and Instance Models

	TM1	GD1	IM1
A1		X	
A2			X
A3	X		X

Table 3: Traceability Matrix Showing the Connections Between Assumptions and Other Items