Module Interface Specification for Movie Recommender

Seyed Ali Mousavi

April 15, 2024

1 Revision History

Date	Version	Notes
2024-03-19	1.0	Initial Release
2024-04-15	2.0	Final Release

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at
https://github.com/alimousavi1997/RecommSys/blob/main/docs/SRS/SRS.pdf

Contents

3 Introduction

The following document details the Module Interface Specifications for Movie Recommender

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/alimousavi1997/RecommSys.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Movie Recommender.

Data Type	**Notation**	**Description**
character	char	a single symbol or digit
string	char*	a sequence of characters
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)

The specification of Movie Recommender uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, Movie Recommender uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Input Format Module Movie Selector Module
Software Decision Module	KNN Module

Table 1: Module Hierarchy

6 MIS of Input Format Module

6.1 Module

Input_Format

6.2 Uses

None

6.3 Syntax

6.3.1 Exported Access Programs

1. load_dataset

   - **Description**: It loads the raw dataset, including user ratings for movies row by row. number of rows in this dataset is the total number of feedbacks.
   - **Input**: A path representing the dataset location. (datatype: char*)
   - **Output**: success or fail (datatype: boolean).
   - **Exceptions**: "FileNotFound" if there is no file in the specified path. "InvalidFileFormat" if the file format doesn't match what it should be (".csv") or if the size of the file is larger than expected.

2. convert_dataset

   - **Description**: Converts the stored raw dataset to a user-item interaction matrix suitable for the KNN algorithm (Pandas DataFrame).
   - **Input**: raw rating dataset. (datatype: Pandas DataFrame)
   - **Output**: user-item interaction matrix in desired form and shape. (datatype: Pandas DataFrame)
   - **Exceptions**: -

6.4 Semantics

6.4.1 State Variables

None

6.4.2 Environment Variables

file: A ".csv" file.

6.4.3 Assumptions

None

6.4.4 Access Routine Semantics

1. load_dataset

   - Transition: Receives and stores the raw dataset.
   - Exceptions: ”FileNotFound” will be raised if no file exists at the provided path. ”InvalidFileFormat” will be raised if the file format does not match the expected format (”.csv”), or if the size of the dataset file exceeds the expected size.

2. convert_dataset

   - Transition: Transforms the stored raw dataset into a compatible format for the k-nearest neighbors (KNN) module, represented as a NumPy array.
   - Exceptions: -

7 MIS of Movie Selector

7.1 Module

Movie_Selector

7.2 Uses

None

7.3 Syntax

7.3.1 Exported Access Programs

1. rating_prediction
   - **Description**: It returns the predicted rating from a target user on a target movie based on the method described in SRS.
   - **Input**: target userId(datatype: Integer) , target movieId(datatype: Integer)
   - **Output**: Predicted rating (datatype: Integer)
   - **Exceptions**: -
2. recommended_movies
   - **Description**: It selects movies(items) for the ”target user” based on popular items among nearby users. Popular items mean movies with the highest rankings.
   - **Input**: target userId. (datatype: Integer)
   - **Output**: Recommended movies. (datatype: a list of strings)
   - **Exceptions**: -

7.4 Semantics

7.4.1 State Variables

None

7.4.2 Environment Variables

None

7.4.3 Assumptions

None

7.4.4 Access Routine Semantics

- Transition: -
- Output: It returns movies for the ”target user” by considering popular items among users who are nearby in similarity.
- Exceptions: -

8 MIS of KNN

8.1 Module

KNN

8.2 Uses

None

8.3 Syntax

8.3.1 Exported Access Programs

1. fit

   - **Description**: Here we follow the conventions of the other machine learning libraries. This will fit the training samples to the model. However, In the KNN method, doesn't involve a training step.
   - **Input**: Training data (datatype: NumPy array)
   - **Output**: -
   - **Exceptions**: -

2. nearest_neighbor_indices

   - **Description**: It first computes Euclidean distances of the target datapoint (target user profile) to all the other data points (other users' profiles) then It returns k number of nearest neighbors indices.
   - **Input**: target user profile which is a data point (datatype: NumPy array)
   - **Output**: k nearest indices. (datatype: Integer Numpy array)
   - **Exceptions**: -

3. _predict

   - **Description**: It returns the predicted rating from a target user for a movie. First, it gets the nearest neighbor indices then it searches and returns the most common class label. (i.e., most common rating from close users on the target movie)
   - **Input**: trained model. Here is the training dataset (datatype: NumPy array)
   - **Output**: predicted rating (datatype: Integer)
   - **Exceptions**: -

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Environment Variables

None

8.4.3 Assumptions

None

8.4.4 Access Routine Semantics

- Transition: -
- Output: It returns the nearest users to the target user based on vector similarities.
- Exceptions: -

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.