Module Interface Specification for Movie Recommender

Seyed Ali Mousavi

March 29, 2024

1 Revision History

Date	Version	Notes
2024-03-19	1.0	Initial 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 $(-\infty, \infty)$
natural number	$\mathbb{N}$	a number without a fractional component in $[1, \infty)$
real	$\mathbb{R}$	any number in $(-\infty, \infty)$

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 Item Selector Module
Software Decision Module	KNN Module

Table 1: Module Hierarchy

6 MIS of Input Format Module

6.1 Module

InputFormat

6.2 Uses

None

6.3 Syntax

6.3.1 Exported Access Programs

1. load_dataset

   - **Description**: Receives and stores the UIM (User-Item-Interaction matrix).
   - **Input**: A path representing the UIM 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. (”.npz”) or if the size of the UIM file is larger than expected

2. convert_dataset

   - **Description**: Converts the stored UIM to a suitable format for knn module (Numpy array).
   - **Input**: raw UIM (datatype: ”.npz”)
   - **Output**: UIM in desired format. (datatype: Numpy array)
   - **Exceptions**: -

6.4 Semantics

6.4.1 State Variables

None

6.4.2 Environment Variables

file: A ”.npz” file.

6.4.3 Assumptions

None

6.4.4 Access Routine Semantics

1. load_dataset
   - Transition: Receives and stores the UIM (User-Item-Interaction matrix).
   - 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 (".npz"), or if the size of the UIM file exceeds the expected size.
2. convert_dataset
   - Transition: Transforms the stored UIM into a compatible format for the k-nearest neighbors (KNN) module, represented as a NumPy array.
   - Exceptions: -

7 MIS of Item Selector

7.1 Module

Itemselector

7.2 Uses

None

7.3 Syntax

7.3.1 Exported Access Programs

1. select_items
   - **Description**: it selects items(movies) for the "target user" based on popular items among nearby users.
   - **Input**: near users indices. (datatype: a list of integers)
   - **Output**: selected 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

1. select_items
   - 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. predict

   - **Description**: It returns k number of nearest neighbors.
   - **Input**: trained model. Here is the training dataset (datatype: NumPy array)
   - **Output**: list of nearest neighbors. (datatype: list of integers)
   - **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.