Module Interface Specification for TTE RecSys

Yinying Huo

March 13, 2025

1 Revision History

Date	Version	Notes
March 2 2025	1.0	First Draft

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://github.com/V-AS/Two-tower-recommender-system/blob/main/docs/SRS/SRS.pdf

Contents

3 Introduction

The following document details the Module Interface Specifications for TTE RecSys

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/V-AS/Two-tower-recommender-system

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 following table summarizes the primitive data types used by TTE RecSys.

Data Type	Notation	Description
character	char	A sequence of characters
Array	$[T]$	A sequence of elements of type T
Matrix	$[T]^{m \times n}$	A 2D array of type T with m rows and n columns
Boolean	$\mathbb{B}$	True or False value
Integer	$\mathbb{Z}$	A number without a fractional component in (-∞, ∞)
real	$\mathbb{R}$	Any number in (-∞, ∞)

TTE RecSys 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.

The specification also uses derived data types:

- Embedding: A vector of real numbers
- Tensors: Multi-dimensional arrays
- User: A type representing user features
- Item: A type representing item features

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	Data Processing Module Model Training Module Embedding Generation Module Recommendation Module
Software Decision Module	Neural Network Architecture Module ANN Search Module Vector Operations Module

Table 1: Module Hierarchy

6 MIS of Hardware-Hiding Module

6.1 Module

SystemInterface

6.2 Uses

None

6.3 Syntax

6.3.1 Exported Constants

None

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
save_model	model: Model, path: String	success: $\mathbb{B}$	IOError
load_model	path: String	model: Model	IOError, FormatError
save_emds	embeddings: [Embedding], path: String	success: $\mathbb{B}$	IOError
load_emds	path: String	embeddings: [Embedding]	IOError

6.4 Semantics

6.4.1 State Variables

None

6.4.2 Environment Variables

FileSystem: The file system where models and embeddings are stored

6.4.3 Assumptions

- The file system is accessible and has sufficient space
- The paths provided are valid

6.4.4 Access Routine Semantics

save_model(model, path):

- output: success = true if operation succeeds
- exception: IOError if file cannot be written

load_model(path):

- output: model
- exception: IOError if file cannot be read, FormatError if file format is invalid

save_embeddings(embeddings, path):

- output: success = true if operation succeeds
- exception: IOError if file cannot be written

oad_embeddings(path):

- output: embeddings
- exception: IOError if file cannot be read, FormatError if file format is invalid

7 MIS of Data Processing Module

7.1 Module

DataProcessor

7.2 Uses

SystemInterface

7.3 Syntax

7.3.1 Exported Constants

None

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
load_data	path: String	data: DataSet	IOError, FormatError
validate_data	data: DataSet	is_valid: $\mathbb{B}$	-
prep_data	data: DataSet	processed_dataset	-
split_data	data: DataSet, train_ratio: $\mathbb{R}$	train_data: DataSet, test_data: DataSet	IOError

7.4 Semantics

7.4.1 State Variables

None

7.4.2 Environment Variables

None

7.4.3 Assumptions

- Input data follows the expected schema

7.4.4 Access Routine Semantics

load_data(path):

- data = parsed data from file at path
- exception: IOError if file cannot be read, FormatError if file format is invalid

validate_data(data)::

- output: is_valid = true if data meets all validation criteria

preprocess_data(data):

- output: processed_data = dataset after applying preprocessing transformations

split_data(data, train_ratio):

- output: (train_data, test_data) where:
  - train_data = subset of data for training (size $\approx$ train_ratio $*|data|$)
  - test_data = subset of data for training (size $\approx$ (1-train_ratio) $*|data|$)
- exception: ValueError if train_ratio is not in $(0, 1)$

8 MIS of Model Training Module

8.1 Module

ModelTrainer

8.2 Uses

DataProcessor, NeuralNetworkArchitecture, VectorOperations

8.3 Syntax

8.3.1 Exported Constants

DEFAULT_LEARNING_RATE = 0.01
DEFAULT_BATCH_SIZE = 128
DEFAULT_REGULARIZATION = 0.01

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
initialize	config: TrainingConfig	-	ValueError
train	train_data: DataSet, epochs: $\mathbb{Z}$	model: Model	IOError, FormatError
save_emds	embeddings: [Embedding], path: String	success: $\mathbb{B}$	IOError
load_emds	path: String	embeddings: [Embedding]	IOError

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Environment Variables

FileSystem: The file system where models and embeddings are stored

8.4.3 Assumptions

- The file system is accessible and has sufficient space
- The paths provided are valid

8.4.4 Access Routine Semantics

save_model(model, path):

- output: success = true if operation succeeds
- exception: IOError if file cannot be written

load_model(path):

- output: model
- exception: IOError if file cannot be read, FormatError if file format is invalid

save_embeddings(embeddings, path):

- output: success = true if operation succeeds
- exception: IOError if file cannot be written

oad_embeddings(path):

- output: embeddings
- exception: IOError if file cannot be read, FormatError if file format is invalid

9 MIS of Hardware-Hiding Module

9.1 Module

SystemInterface

9.2 Uses

None

9.3 Syntax

9.3.1 Exported Constants

None

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
save_model	model: Model, path: String	success: $\mathbb{B}$	IOError
load_model	path: String	model: Model	IOError, FormatError
save_emds	embeddings: [Embedding], path: String	success: $\mathbb{B}$	IOError
load_emds	path: String	embeddings: [Embedding]	IOError

9.4 Semantics

9.4.1 State Variables

None

9.4.2 Environment Variables

FileSystem: The file system where models and embeddings are stored

9.4.3 Assumptions

- The file system is accessible and has sufficient space
- The paths provided are valid

9.4.4 Access Routine Semantics

save_model(model, path):

- output: success = true if operation succeeds
- exception: IOError if file cannot be written

load_model(path):

- output: model
- exception: IOError if file cannot be read, FormatError if file format is invalid

save_embeddings(embeddings, path):

- output: success = true if operation succeeds
- exception: IOError if file cannot be written

oad_embeddings(path):

- output: embeddings
- exception: IOError if file cannot be read, FormatError if file format is invalid

10 MIS of Hardware-Hiding Module

10.1 Module

SystemInterface

10.2 Uses

None

10.3 Syntax

10.3.1 Exported Constants

None

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
save_model	model: Model, path: String	success: $\mathbb{B}$	IOError
load_model	path: String	model: Model	IOError, FormatError
save_emds	embeddings: [Embedding], path: String	success: $\mathbb{B}$	IOError
load_emds	path: String	embeddings: [Embedding]	IOError

10.4 Semantics

10.4.1 State Variables

None

10.4.2 Environment Variables

FileSystem: The file system where models and embeddings are stored

10.4.3 Assumptions

- The file system is accessible and has sufficient space
- The paths provided are valid

10.4.4 Access Routine Semantics

save_model(model, path):

- output: success = true if operation succeeds
- exception: IOError if file cannot be written

load_model(path):

- output: model
- exception: IOError if file cannot be read, FormatError if file format is invalid

save_embeddings(embeddings, path):

- output: success = true if operation succeeds
- exception: IOError if file cannot be written

oad_embeddings(path):

- output: embeddings
- exception: IOError if file cannot be read, FormatError if file format is invalid

11 MIS of Hardware-Hiding Module

11.1 Module

SystemInterface

11.2 Uses

None

11.3 Syntax

11.3.1 Exported Constants

None

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
save_model	model: Model, path: String	success: $\mathbb{B}$	IOError
load_model	path: String	model: Model	IOError, FormatError
save_emds	embeddings: [Embedding], path: String	success: $\mathbb{B}$	IOError
load_emds	path: String	embeddings: [Embedding]	IOError

11.4 Semantics

11.4.1 State Variables

None

11.4.2 Environment Variables

FileSystem: The file system where models and embeddings are stored

11.4.3 Assumptions

- The file system is accessible and has sufficient space
- The paths provided are valid

11.4.4 Access Routine Semantics

save_model(model, path):

- output: success = true if operation succeeds
- exception: IOError if file cannot be written

load_model(path):

- output: model
- exception: IOError if file cannot be read, FormatError if file format is invalid

save_embeddings(embeddings, path):

- output: success = true if operation succeeds
- exception: IOError if file cannot be written

oad_embeddings(path):

- output: embeddings
- exception: IOError if file cannot be read, FormatError if file format is invalid

11.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

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.

12 Appendix

[Extra information if required —SS]

Appendix — Reflection

[Not required for CAS 741 projects —SS]

The information in this section will be used to evaluate the team members on the graduate attribute of Problem Analysis and Design.

1. What went well while writing this deliverable?

2. What pain points did you experience during this deliverable, and how did you resolve them?

3. Which of your design decisions stemmed from speaking to your client(s) or a proxy (e.g. your peers, stakeholders, potential users)? For those that were not, why, and where did they come from?

4. While creating the design doc, what parts of your other documents (e.g. requirements, hazard analysis, etc), it any, needed to be changed, and why?

5. What are the limitations of your solution? Put another way, given unlimited resources, what could you do to make the project better? (LO_ProbSolutions)

6. Give a brief overview of other design solutions you considered. What are the benefits and tradeoffs of those other designs compared with the chosen design? From all the potential options, why did you select the documented design? (LO_Explores)