

# Module Interface Specification for TTE RecSys

Yinying Huo

March 14, 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

<b>1</b>	<b>Revision History</b>	<b>i</b>
<b>2</b>	<b>Symbols, Abbreviations and Acronyms</b>	<b>ii</b>
<b>3</b>	<b>Introduction</b>	<b>1</b>
<b>4</b>	<b>Notation</b>	<b>1</b>
<b>5</b>	<b>Module Decomposition</b>	<b>1</b>
<b>6</b>	<b>MIS of Hardware-Hiding Module</b>	<b>3</b>
6.1	Module . . . . .	3
6.2	Uses . . . . .	3
6.3	Syntax . . . . .	3
6.3.1	Exported Constants . . . . .	3
6.3.2	Exported Access Programs . . . . .	3
6.4	Semantics . . . . .	3
6.4.1	State Variables . . . . .	3
6.4.2	Environment Variables . . . . .	3
6.4.3	Assumptions . . . . .	3
6.4.4	Access Routine Semantics . . . . .	4
<b>7</b>	<b>MIS of Data Processing Module</b>	<b>4</b>
7.1	Module . . . . .	4
7.2	Uses . . . . .	4
7.3	Syntax . . . . .	4
7.3.1	Exported Constants . . . . .	4
7.3.2	Exported Access Programs . . . . .	4
7.4	Semantics . . . . .	5
7.4.1	State Variables . . . . .	5
7.4.2	Environment Variables . . . . .	5
7.4.3	Assumptions . . . . .	5
7.4.4	Access Routine Semantics . . . . .	5
<b>8</b>	<b>MIS of Model Training Module</b>	<b>5</b>
8.1	Module . . . . .	5
8.2	Uses . . . . .	5
8.3	Syntax . . . . .	6
8.3.1	Exported Constants . . . . .	6
8.3.2	Exported Access Programs . . . . .	6
8.4	Semantics . . . . .	6
8.4.1	State Variables . . . . .	6

8.4.2	Environment Variables . . . . .	6
8.4.3	Assumptions . . . . .	6
8.4.4	Access Routine Semantics . . . . .	6
<b>9</b>	<b>MIS of Hardware-Hiding Module</b>	<b>7</b>
9.1	Module . . . . .	7
9.2	Uses . . . . .	7
9.3	Syntax . . . . .	7
9.3.1	Exported Constants . . . . .	7
9.3.2	Exported Access Programs . . . . .	7
9.4	Semantics . . . . .	8
9.4.1	State Variables . . . . .	8
9.4.2	Environment Variables . . . . .	8
9.4.3	Assumptions . . . . .	8
9.4.4	Access Routine Semantics . . . . .	8
<b>10</b>	<b>MIS of Hardware-Hiding Module</b>	<b>8</b>
10.1	Module . . . . .	8
10.2	Uses . . . . .	9
10.3	Syntax . . . . .	9
10.3.1	Exported Constants . . . . .	9
10.3.2	Exported Access Programs . . . . .	9
10.4	Semantics . . . . .	9
10.4.1	State Variables . . . . .	9
10.4.2	Environment Variables . . . . .	9
10.4.3	Assumptions . . . . .	9
10.4.4	Access Routine Semantics . . . . .	9
<b>11</b>	<b>MIS of Hardware-Hiding Module</b>	<b>10</b>
11.1	Module . . . . .	10
11.2	Uses . . . . .	10
11.3	Syntax . . . . .	10
11.3.1	Exported Constants . . . . .	10
11.3.2	Exported Access Programs . . . . .	10
11.4	Semantics . . . . .	11
11.4.1	State Variables . . . . .	11
11.4.2	Environment Variables . . . . .	11
11.4.3	Assumptions . . . . .	11
11.4.4	Access Routine Semantics . . . . .	11
11.4.5	Local Functions . . . . .	11
<b>12</b>	<b>Appendix</b>	<b>13</b>

### 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 $(-\infty, \infty)$
real	$\mathbb{R}$	Any number in $(-\infty, \infty)$

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	
	Data Processing Module
	Model Training Module
	Embedding Generation Module
Behaviour-Hiding 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: TrainingCon- fig	-	ValueError
train	train_data: DataSet, epochs: $\mathbb{Z}$	model: Model	IOError, FormatError
save_emds	embeddings: [Embed- ding], path: String	success: $\mathbb{B}$	IOError
load_emds	path: String	embeddings: [Embed- ding]	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

load\_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

load\_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]