Module Interface Specification for TTE RecSys

Yinying Huo

March 30, 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

1 Revision History **i**

2 Symbols, Abbreviations and Acronyms **ii**

3 Introduction **1**

4 Notation **1**

5 Module Decomposition **2**

6 MIS of Hardware-Hiding Module **4**
6.1 Module . . . 4
6.2 Uses . . . 4
6.3 Syntax . . . 4
6.3.1 Exported Constants . . . 4
6.3.2 Exported Access Programs . . . 4
6.4 Semantics . . . 4
6.4.1 State Variables . . . 4
6.4.2 Environment Variables . . . 4
6.4.3 Assumptions . . . 4
6.4.4 Access Routine Semantics . . . 5

7 MIS of Data Processing Module **5**
7.1 Module . . . 5
7.2 Uses . . . 5
7.3 Syntax . . . 5
7.3.1 Exported Constants . . . 5
7.3.2 Exported Access Programs . . . 5
7.4 Semantics . . . 6
7.4.1 State Variables . . . 6
7.4.2 Environment Variables . . . 6
7.4.3 Assumptions . . . 6
7.4.4 Access Routine Semantics . . . 6

8 MIS of Model Training Module **6**
8.1 Module . . . 6
8.2 Uses . . . 6
8.3 Syntax . . . 7
8.3.1 Exported Constants . . . 7
8.3.2 Exported Access Programs . . . 7
8.4 Semantics . . . 7
8.4.1 State Variables . . . 7

8.4.2 Environment Variables . . . 7
8.4.3 Assumptions . . . 7
8.4.4 Access Routine Semantics . . . 7
8.4.5 Local Functions . . . 8

9 MIS of Embedding Generation Module 8
9.1 Module . . . 8
9.2 Uses . . . 9
9.3 Syntax . . . 9
9.3.1 Exported Constants . . . 9
9.3.2 Exported Access Programs . . . 9
9.4 Semantics . . . 9
9.4.1 State Variables . . . 9
9.4.2 Environment Variables . . . 9
9.4.3 Assumptions . . . 9
9.4.4 Access Routine Semantics . . . 9

10 MIS of Recommendation Module 10
10.1 Module . . . 10
10.2 Uses . . . 10
10.3 Syntax . . . 10
10.3.1 Exported Constants . . . 10
10.3.2 Exported Access Programs . . . 10
10.4 Semantics . . . 11
10.4.1 State Variables . . . 11
10.4.2 Environment Variables . . . 11
10.4.3 Assumptions . . . 11
10.4.4 Access Routine Semantics . . . 11
10.4.5 Local Functions . . . 11

11 MIS of Neural Network Architecture Module 12
11.1 Module . . . 12
11.2 Uses . . . 12
11.3 Syntax . . . 12
11.3.1 Exported Constants . . . 12
11.3.2 Exported Access Programs . . . 12
11.4 Semantics . . . 12
11.4.1 State Variables . . . 12
11.4.2 Environment Variables . . . 12
11.4.3 Assumptions . . . 12
11.4.4 Access Routine Semantics . . . 12

12 MIS of ANN Search Module **13**
12.1 Module . . . 13
12.2 Uses . . . 13
12.3 Syntax . . . 13
12.3.1 Exported Constants . . . 13
12.3.2 Exported Access Programs . . . 13
12.4 Semantics . . . 14
12.4.1 State Variables . . . 14
12.4.2 Environment Variables . . . 14
12.4.3 Assumptions . . . 14
12.4.4 Access Routine Semantics . . . 14

13 MIS of Vector Operations Module **14**
13.1 Module . . . 14
13.2 Uses . . . 15
13.3 Syntax . . . 15
13.3.1 Exported Constants . . . 15
13.3.2 Exported Access Programs . . . 15
13.4 Semantics . . . 15
13.4.1 State Variables . . . 15
13.4.2 Environment Variables . . . 15
13.4.3 Assumptions . . . 15
13.4.4 Access Routine Semantics . . . 15

3 Introduction

The following document details the Module Interface Specifications for TTE RecSys, a two-tower recommendation system. The system leverages deep learning to create both the user tower and item tower, mapping inputs to a shared embedding space. Then, an effective algorithm is used to select a large number of candidate items. Finally, the dot product is applied for a refined ranking of the candidate items, returning the final recommendations accordingly.

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 single character
String	string	A sequence of characters representing text
Array	$[T]$	A sequence of elements of type T
Dictionary/Map	dict	A collection of key-value pairs
Vector	$\mathbb{R}^n$	An ordered collection of n real numbers
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 (-∞, ∞)
Tuple	$(T_1, T_2, ...)$	An ordered collection of elements with possibly different types

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
- DataSet: A collection of data records containing user, item, and interaction information
- Model: A neural network model
- ANNIndex: An index structure for approximate nearest neighbor search
- TrainingConfig: A dictionary containing configuration parameters for model training (learning rate, batch size, etc.)
- LayerConfig: A dictionary describing the configuration of neural network layers
- EvaluationMetrics: A dictionary of evaluation metric names and values

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
evaluate	test_data: DataSet,	metrics: EvaluationMetrics	-
get_user_model	-	user_model: Model	NotInitializedError
get_item_model	-	item_model: Model	NotInitializedError

8.4 Semantics

8.4.1 State Variables

- UserModel: The neural network model for the user tower
- ItemModel: The neural network model for the item tower
- IsInitialized: Boolean indicating if the module has been initialized

8.4.2 Environment Variables

None

8.4.3 Assumptions

- The training data is preprocessed and valid
- The model configuration is valid

8.4.4 Access Routine Semantics

initialize(config):

- transition:
  - UserModel = create_user_model(config.user_architecture)

- ItemModel = create_item_model(config.item_architecture)
- IsInitialized = true

- exception: ValueError if config contains invalid parameters

train(train_data, epochs):

- transition:
  - Update UserModel and ItemModel parameters through training
- output: history = record of loss values and metrics during training
- exception: NotInitializedError if IsInitialized is false

evaluate(test_data):

- output: metrics = evaluation metrics on test data
- exception: NotInitializedError if IsInitialized is false

get_user_model():

- output: user_model = UserModel
- exception: NotInitializedError if IsInitialized is false

get_item_model():

- output: item_model = UserModel
- exception: NotInitializedError if IsInitialized is false

8.4.5 Local Functions

compute_loss(user_embeddings, item_embeddings, ratings):

- Type: $[\mathbb{R}^k] \times \mathbb{R}^k \times \mathbb{R}$
- Description: Computes MSE loss between predicted and actual ratings

9 MIS of Embedding Generation Module

9.1 Module

EmbeddingGenerator

9.2 Uses

NeuralNetworkArchitecture, VectorOperations

9.3 Syntax

9.3.1 Exported Constants

EMBEDDING_DIMENSION = 128

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
initialize	user_model: Model,item_model: Model	-	ValueError
generate_user_embedding	users: [ProcessedUser]	embeddings: [Embedding]	NotInitializedError
generate_item_embedding	items: [ProcessedItem]	embeddings: [Embedding]	NotInitializedError

9.4 Semantics

9.4.1 State Variables

- UserModel: The neural network model for the user tower
- ItemModel: The neural network model for the item tower
- IsInitialized: Boolean indicating if the module has been initialized

9.4.2 Environment Variables

None

9.4.3 Assumptions

- The models have been trained

9.4.4 Access Routine Semantics

initialize(user_model, item_model):

- transition:
  - UserModel = user_model

- ItemModel = item_model
- IsInitialized = true

- exception: ValueError if models are incompatible

generate_user_embedding(users):

- output: embeddings := [UserModel(user) for user in users]
- exception: NotInitializedError if IsInitialized is false

generate_item_embedding(items):

- output: embeddings := [ItemModel(item) for item in items]
- exception: NotInitializedError if IsInitialized is false

10 MIS of Recommendation Module

10.1 Module

Recommender

10.2 Uses

EmbeddingGenerator, ANNSearch, VectorOperations

10.3 Syntax

10.3.1 Exported Constants

DEFAULT_NUM_RECOMMENDATIONS = 10 SIMILARITY_THRESHOLD = 0.5

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
initialize	ann_index : ANNIndex	-	ValueError
get_recommendations	user: ProcessedUser, num_results: $\mathbb{Z}$	recommendations: [Recommendation]	NotInitializedError, FormatError
evaluate_rec	test_data: DataSet, path: String	metrics: RecommendationMetrics	NotInitializedError

10.4 Semantics

10.4.1 State Variables

ANNIndex: The index for approximate nearest neighbor search EmbeddingGen: Reference to the embedding generator IsInitialized: Boolean indicating if the module has been initialized

10.4.2 Environment Variables

None

10.4.3 Assumptions

- The ANN index has been built with item embeddings
- The embedding generator has been initialized with trained models

10.4.4 Access Routine Semantics

initialize(ann_index):

- transition:
  - ANNIndex = ann_index
  - IsInitialized = true
- exception: ValueError if ann_index is invalid

get_recommendations(user, num_results):

- output: recommendations = ranked list of recommended items with similarity scores
- exception: NotInitializedError if IsInitialized is false

evaluate_recommendations(test_data):

- output: metrics = evaluation metrics for recommendations
- exception: NotInitializedError if IsInitialized is false

10.4.5 Local Functions

rank_candidates(user_embedding, candidate_embeddings):

- Description: Ranks candidates by similarity score (dot product)
- Type: $\mathbb{R}^k \times [\mathbb{R}^k] \rightarrow [(\mathbb{Z}, \mathbb{R})]$

11 MIS of Neural Network Architecture Module

11.1 Module

NeuralNetworkArchitecture

11.2 Uses

VectorOperations

11.3 Syntax

11.3.1 Exported Constants

DEFAULT_HIDDEN_LAYERS = [256, 128] DEFAULT_ACTIVATION = "relu"

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
create_user_tower	input_dim: $\mathbb{Z}$, hidden_layers: [$\mathbb{Z}$] , embedding_dim: $\mathbb{Z}$	model: Model	ValueError
create_item_tower	input_dim: $\mathbb{Z}$, hidden_layers: [$\mathbb{Z}$] , embedding_dim: $\mathbb{Z}$	model: Model	ValueError
get_layer_config	model: Model	config: LayerConfig	ValueError

11.4 Semantics

11.4.1 State Variables

None

11.4.2 Environment Variables

None

11.4.3 Assumptions

None

11.4.4 Access Routine Semantics

create_user_tower(input_dim, hidden_layers, embedding_dim):

- output: model = neural network model for user tower

- exception: ValueError if dimensions are invalid

create_item_tower(input_dim, hidden_layers, embedding_dim):

- output: model = neural network model for item tower
- exception: ValueError if dimensions are invalid

get_layer_config(model):

- output: config := configuration of layers in the model
- exception: ValueError if model is invalid

12 MIS of ANN Search Module

12.1 Module

ANNSearch

12.2 Uses

VectorOperations

12.3 Syntax

12.3.1 Exported Constants

DEFAULT_SEARCH_NPROBE := 10 DEFAULT_INDEX_TYPE := "IVF"

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
build_index	embeddings: [Embedding] , item_ids: [$\mathbb{Z}$], index_type: String	index: ANNIndex	ValueError
search	index: ANNIndex, query: Embedding, k: $\mathbb{Z}$	results: [($\mathbb{Z}, \mathbb{R}$)]	ValueError
save_index	index: ANNIndex, path: String	success: $\mathbb{B}$	IOError
load_index	path: String	index: ANNIndex	IOError, FormatError

12.4 Semantics

12.4.1 State Variables

None

12.4.2 Environment Variables

None

12.4.3 Assumptions

- Embeddings are of consistent dimension
- Query vector is of same dimension as indexed vectors

12.4.4 Access Routine Semantics

build_index(embeddings, item_ids, index_type):

- output: index = ANN index built from embeddings and associated item IDs
- exception: ValueError if parameters are invalid

search(index, query, k):

- output: results = list of (item_id, similarity_score) tuples for k nearest neighbors
- exception: ValueError if parameters are invalid

save_index(index, path):

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

load_index(path):

- output: index := ANNIndex loaded from file
- exception: IOError if file cannot be read, FormatError if file format is invalid

13 MIS of Vector Operations Module

13.1 Module

VectorOperations

13.2 Uses

None

13.3 Syntax

13.3.1 Exported Constants

EPSILON := 1e-8 (small value to prevent division by zero)

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
dot_product	v1: [$\mathbb{R}$], v2: [$\mathbb{R}$]	result: $\mathbb{R}$	DimensionMismatchError

13.4 Semantics

13.4.1 State Variables

None

13.4.2 Environment Variables

None

13.4.3 Assumptions

None

13.4.4 Access Routine Semantics

dot_product(v1, v2):

- output: result = $\sum_i^{len(v1)} v1[i] * v2[i]$
- exception: DimensionMismatchError if len(v1) $\neq$ len(v2)

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.