Software Requirements Specification for TTE RecSys: A Two Tower Embeddings Recommendation System

Yinying Huo

March 2, 2025

Contents

Revision History

Date	Version	Notes
Feb 6 2025	1.0	Fisrt draft
Feb 17 2025	1.1	Addressed comments from Yuanqi

1 Reference Material

This section records information for easy reference.

1.1 Table of Units

Not Applicable

1.2 Abbreviations and Acronyms

symbol	description
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
TTE RecSys	Two-tower embending recommendation system
TM	Theoretical Model
DNN	Deep Neural Network
ANN	Approximate Nearest Neighbor

1.3 Mathematical Notation

symbol	description
$\mathbb{R}$	Set of real numbers
θ	Model parameter, a vector of real numbers
$u(\cdot)$	A function that maps model parameter θ and features of a user to the embedding space
$v(\cdot)$	A function that maps model parameter θ and features of an item to the embedding space
x_i	User feature
y_i	Item feature
k	Dimension of embedding space
r_i	The associated reward for each user and item pair
$J(\cdot)$	Regularization function to prevent overfitting
$\mathcal{L}(\theta)$	Loss function
η	Learning rate
B	Randomly selected subset of the training data used in one iteration of model training
γ	Hyperparameter to control the regularization term

2 Introduction

In the digital era, recommendation systems play a crucial role in enhancing user experiences by providing personalized content, products, and services. These systems are widely used in industries such as e-commerce, streaming platforms, and social media to help users discover relevant items from vast catalogs. Traditional recommendation approaches, including collaborative filtering and content-based methods, rely on explicit user-item interactions or manually engineered features. However, as datasets grow in complexity and size, these methods face challenges in scalability and efficiency.

To overcome these limitations, modern recommendation systems leverage deep learning-based techniques, such as Two-Tower Embeddings (TTE). This project implements a recommendation system using a TTE model, where user and item features are mapped into a shared embedding space. To efficiently retrieve relevant items, the project will use FAISS as a tool to perform the Approximate Nearest Neighbor (ANN) search. FAISS is a vector similarity search library developed by Meta.

The training dataset used for this project is from Kaggle: https://www.kaggle.com/datasets/arashnic/book-recommendation-dataset.

2.1 Purpose of Document

The purpose of this document is to define the software requirements for the recommendation system based on Two-Tower Embeddings (TTE). It serves as a reference for the system's

functionality, design considerations, and constraints.

2.2 Scope of Requirements

This project assumes that users and items are fully represented by the features available in the dataset. External factors, such as social influences, or temporal patterns, are not considered. The recommendation model will rely solely on the provided user-item interaction data to generate recommendations.

2.3 Characteristics of Intended Reader

Readers of this documentation are expected to have completed an introductory course on machine learning, including fundamental concepts in linear algebra, statistics and probability, and basic optimization techniques (e.g., gradient descent). Familiarity with neural networks and nearest neighbor search methods is also recommended.

2.4 Organization of Document

This document starts with reference materials, such as abbreviations and mathematical notation, followed by the problem, objectives, assumptions, theoretical model, and instance models. It also defines the project's functional and nonfunctional requirements.

3 General System Description

This section provides general information about the system. It identifies the interfaces between the system and its environment, describes the user characteristics and lists the system constraints.

3.1 System Context

The recommendation system follows a structured workflow consisting of two main phases: **training** and **inference**. The system leverages Two-Tower Embeddings and ANN search to generate personalized recommendations efficiently.

Training Phase

During the training phase, the system learns two embedding functions:

- **User Embedding Function:** Maps user features (e.g., age=20, gender=male) into a fixed-dimensional vector representation (e.g., $[1, 2.3, 4]$) in the embedding space.
- **Item Embedding Function:** Maps item features into the same embedding space as users.

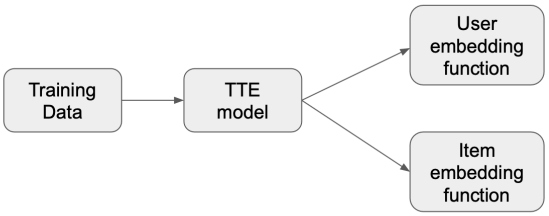

Figure 1: Training phase

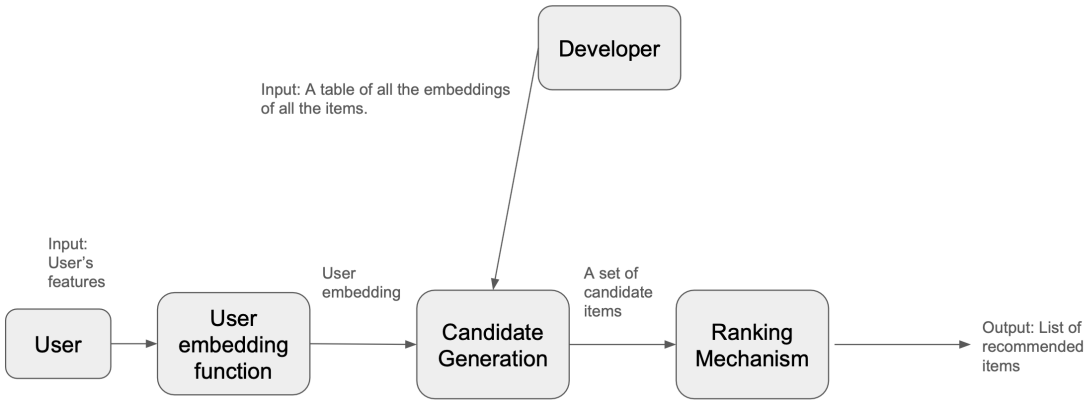

Figure 2: Inference phase

During the training phase, the objective is to make the dot product of the outputs of the two embedding functions approximately equal to the associated rating r_i. This can be achieved by minimizing the loss function.

Once training is complete, the item embedding function is used to precompute embeddings for all items. These embeddings are stored in an ANN index for efficient retrieval during inference.

Inference Phase

When a user interacts with the system, the following steps occur:

1. The user's features are processed by the user embedding function to generate an embedding.

2. This embedding is used as a query in the ANN index to retrieve a set of candidate items.
3. Candidates are ranked by similarity (via dot product).
4. The final ranked list is presented as recommendations to the user.

- User Responsibilities:
  - Provide correct inputs.
- Developer Responsibilities:
  - Precompute item embeddings before the user interacts with the system.
- TTE RecSys Responsibilities:
  - Detect data type mismatches.
  - Generate recommendations as output.

3.2 User Characteristics

The end user are not required to have any technical knowledge. The only responsibility of the user is to provide their information in the required format.

3.3 System Constraints

There is no system constrain for this software.

4 Specific System Description

This section first presents the problem description, which gives a high-level view of the problem to be solved. This is followed by the solution characteristics specification, which presents the assumptions, theories, definitions and finally the instance models.

4.1 Problem Description

TTE RecSys is intended to build a recommendation system based on the Two-Tower Embeddings architecture.

4.1.1 Terminology and Definitions

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements:

- **Approximate Nearest Neighbor (ANN) Search**: A method for efficiently retrieving the most similar items to a query embedding by approximating the nearest neighbor search
- **Embedding**: A dense, low-dimensional vector representation of a user or item in a continuous vector space. Embeddings capture semantic relationships and are learned by the two-tower model.
- **Two-Tower Embedding (Model)**: A neural network architecture consisting of two separate towers (user and item) that generate embeddings. The model is trained to predict user-item relevance.
- **Regularization**: A technique to prevent overfitting by adding a penalty term to the loss function.
- **Batch**: A subset of the training dataset used during a single optimization step.

4.1.2 Goal Statements

GS1: Learn the embedding functions from a training dataset.

GS2: Return a list of recommended items.

4.2 Solution Characteristics Specification

4.2.1 Modelling Decisions

This project will be written in Python due to the availability of various machine learning libraries in Python.

4.2.2 Assumptions

This section simplifies the original problem and helps in developing the theoretical model by filling in the missing information for the physical system. The numbers given in the square brackets refer to the theoretical model [TM], general definition [GD], instance model [IM], or likely change [LC], in which the respective assumption is used.

A1: User features can be effectively represented as low-dimensional dense vectors (embeddings). [IM1][IM2][IM3][LC1]

A2: Item features can be effectively represented as low-dimensional dense vectors (embeddings). [IM1][IM2][IM3][LC1]

A3: The dot product between user and item embeddings is a valid measure of user interest in items. [IM1][IM2][IM3]

A4: User-item interactions in the training data reflect true user preferences and are free from systemic bias. [IM1][IM2][IM3]

4.2.3 Theoretical Models

This section focuses on the general equations and laws that TTE RecSys is based on.

RefName: **TM:Mean Squared Error**

Label: Regularized Mean Squared Error (MSE) Loss

Equation: $$\mathcal{L}(\theta) = \min_{\theta} \frac{1}{|B|} \sum_{(x_i, y_i, r_i) \in B} \left(r_i - \langle u(x_i; \theta), v(y_i; \theta) \rangle\right)^2 + \gamma \left[J(u) + J(v)\right]$$

Description: The two-tower model is trained to minimize the Mean Squared Error (MSE) between predicted relevance scores $\langle u(x_i; \theta), v(y_i; \theta) \rangle$ and observed rewards r_i. The regularization term $\gamma \left[J(u) + J(v)\right]$ penalizes model complexity using L_1 or L_2 norms to prevent overfitting.

Notes: - $u(x_i; \theta), v(y_i; \theta)$: User and item embeddings generated by deep neural networks. - $J(u) = \|\theta_u\|_p$, $J(v) = \|\theta_v\|_p$: Regularization on network parameters ($p = 1$ or 2). - γ: Hyperparameter controlling regularization strength.

Source: Wikipedia (2025c)

Ref. By: GS1, A1, A2

Preconditions for TM:Mean Squared Error: Training batch $B \subset \mathcal{T}$ is sampled uniformly from the dataset.

Derivation for TM:Mean Squared Error: Not Applicable

RefName: **TM:OPT**

Label: Mathematical Optimization

Equation: $\theta^* = \arg\min_{\theta \in \Theta} \mathcal{L}(\theta)$

Description: The general problem of finding parameter values θ^* within a feasible set Θ that minimize a loss function $\mathcal{L}(\theta)$.

Notes: Θ: Parameter space
$\mathcal{L}$: Real-valued functions to be minimized
Solution existence and uniqueness dependent on properties of the functions and parameter space

Source: Wikipedia (2025b)

Ref. By: IM3

Preconditions for TM:OPT: No specific requirements on the functions or parameters, though solvability depends on their properties

Derivation for TM:OPT: Not Applicable

4.2.4 Instance Models

This section transforms the problem defined in Section 4.1 into one which is expressed in mathematical terms.

The goals GS1 and GS2 are addressed by the following mathematical models:

Number	IM1
Label	**ANN-based Candidate Generation**
Input	• User embedding $u(x_i;\theta) \in \mathbb{R}^k$ (from user tower). • Precomputed item embeddings $\{v(y_i;\theta)\}_{i=1}^N \in \mathbb{R}^k$ (from item tower). • K: Number of candidates to return.
Output	Top-K candidate indices $I \subset \{1,\ldots,N\}$
Description	• Builds an ANN index (e.g., FAISS-IVFPQ) over $\{v(y_i;\theta)\}_{i=1}^N$. • Queries the index with $u(x_i;\theta)$ to retrieve candidates: $I = \text{ANN-Search}\left(v(y_i;\theta), \{v(y_i;\theta)\}_{i=1}^N, K\right)$
Sources	Douze et al. (2024)
Ref. By	R2, R3, GS2

Number	IM2
Label	**Dot Product Ranking**
Input	• User embedding $u(x_i;\theta)$. • Candidate item embeddings $\{v(y_i;\theta)\}_{i\in I}$. • N: Number of items to return.
Output	Ranked list I_{ranked} sorted by $s(x_i,y_i) = \langle u(x_i;\theta), v(y_i;\theta)\rangle$.
Description	• Computes exact dot product scores: $s(x_i,y_i) = \langle u(x_i;\theta), v(y_i;\theta)\rangle$. • Sorts candidates in descending order of $s(x_i,y_i)$. • Returns top-N items.
Sources	Wikipedia (2025a)
Ref. By	R2, R3, GS2

Number	IM3
Label	**Stochastic Gradient Descent**
Input	• Training dataset $\mathcal{D} = \{(x_i, y_i, r_i)\}_{i=1}^{T}$ • Initial parameters θ_0 • Learning rate η • Mini-batch size $\|B\|$
Output	Learned parameters θ^*
Description	• For each iteration: 1. Sample mini-batch $B \subset \mathcal{D}$ 2. Compute gradient estimate: $\nabla_\theta \mathcal{L}_B(\theta_t)$ 3. Update parameters: $\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}_B(\theta_t)$ • Loss function with regularization: $\mathcal{L}_B(\theta) = \frac{1}{\|B\|} \sum_{(x_i,y_i)\in B} (r_i - \langle u(x_i;\theta), v(y_i;\theta)\rangle)^2 + \gamma(J(u) - J(v))$
Sources	Wikipedia (2025d)
Ref. By	GS1

4.2.5 Input Data Constraints

Table 1: Input Variables

Var	Physical Constraints	Software Constraints	Typical Value	Uncertainty
r_i	$r_i \in \mathbb{R}$	$0 \leq r_i \leq 10$	8	0%

(*) r_i is the rating that user x_i assigns to the item y_i.

5 Requirements

This section provides the functional requirements, the business tasks that the software is expected to complete, and the nonfunctional requirements, the qualities that the software is expected to exhibit.

5.1 Functional Requirements

R1: The system shall accept user features, item features, and training data with observed rewards.

R2: Identical user/item embeddings shall produce identical rankings.

R3: The system shall return a ranked list of recommended items with similarity scores.

5.2 Nonfunctional Requirements

NFR1: **Scalability** The system shall support incremental updates to trained models when new data becomes available, with updates completing within a time proportional to the dataset size.

NFR2: **Reliability:** The reliability is measured and verified in the VnV Plan

NFR3: **Portability** TTE RecSys shall operate on Windows, Linux, and macOS environments

5.3 Rationale

The scope requirements are designed to simplify the model architecture and reduce computational complexity.

Assumptions A1 and A2 strike a balance between expressiveness and computational efficiency. Additionally, A3 eliminates the need for normalization, preserving magnitude information in embeddings. Furthermore, A4 ensures that the model learns meaningful patterns.

Modeling Decision Rationale: Python is chosen for its extensive machine learning ecosystem, including libraries such as PyTorch and FAISS, as well as its rapid prototyping capabilities and strong community support.

6 Likely Changes

LC1: The system may adopt more advanced ANN algorithms (e.g., ScaNN or HNSW) to improve retrieval efficiency or accuracy.

7 Unlikely Changes

None

8 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that are marked with an "X" may have to be modified as well. Table 4 shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table 5 shows the dependencies of instance models, requirements, and data constraints on each other. Table 3 shows the dependencies of theoretical models, general definitions, data definitions, instance models, and likely changes on the assumptions.

	A1	A2	A3	A4
TM??	X	X		
TM4.2.3			X	X
IM1	X	X	X	X
IM2	X	X	X	X
IM3	X	X		
LC1	X	X		X

Table 3: Traceability Matrix Showing the Connections Between Assumptions and Other Items

	TM??	TM4.2.3
IM3	X	X
IM2		X
IM1		X

Table 4: Traceability Matrix Showing the Connections Between Items of Different Sections

References

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library, 2024. URL https://arxiv.org/abs/2401.08281.

	IM1	IM2	IM3
R1	X		
R2		X	
R3	X	X	X

Table 5: Traceability Matrix Showing the Connections Between Requirements and Instance Models

Wikipedia. Dot product — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Dot%20product&oldid=1268352915, 2025a. [Online; accessed 17-February-2025].

Wikipedia. Gradient descent — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Gradient%20descent&oldid=1274464503, 2025b. [Online; accessed 17-February-2025].

Wikipedia. Mean squared error — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Mean%20squared%20error&oldid=1276072434, 2025c. [Online; accessed 17-February-2025].

Wikipedia. Stochastic gradient descent — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Stochastic%20gradient%20descent&oldid=1265558819, 2025d. [Online; accessed 17-February-2025].