A Multi-objective Loss Function for Neighbor-Based Recommender Systems

Ho Thi Hoang Vy[1,3], Hien D. Nguyen[2,3], and Le Nguyen Hoai Nam[1,3(✉)]

[1] Faculty of Information Technology, University of Science, Ho Chi Minh City, Vietnam
{hthvy,lnhnam}@fit.hcmus.edu.vn
[2] University of Information Technology, Ho Chi Minh City, Vietnam
hiennd@uit.edu.vn
[3] Vietnam National University, Ho Chi Minh City, Vietnam

Abstract. Neighbor-based recommendation models exploit users who are similar to an active user, referred to as neighbors, to predict individual preferences for recommendation tasks. These models are typically optimized through a loss function to estimate the influence of neighbors on the active user. Existing studies commonly design the loss function by minimizing the discrepancy between predicted preferences and observed numerical ratings. However, modern recommender systems can also collect textual comments from users, which provide additional signals about their preferences. Motivated by this observation, this paper extends the conventional loss function by incorporating textual comments into the optimization process. The proposed multi-objective loss function enables the model to jointly learn from both numerical ratings and textual comments. As a result, the recommendation model becomes more robust, particularly in scenarios where observed rating data are sparse. Experimental results demonstrate that the proposed approach improves recommendation accuracy, especially for users with limited rating histories.

Keywords: Neighbor-based model · Collaborative filtering · Recommendation systems

1 Introduction

Recommender systems have been developed to personalize customer experiences across various application domains [1]. In such systems, user preferences are predicted using machine learning models, enabling the delivery of items that best match individual needs. Among the various approaches, one of the most widely adopted is based on the assumption that an active user's preference can be inferred from the preferences of other users who are similar to them, commonly referred to as neighbors [2]. This intuition also enhances the transparency and interpretability of recommendation models.

In neighbor-based recommendation models, the key parameters capture the influence of each neighbor on the predicted preference of the active user. Based on these parameters, the model aggregates the preferences of neighbors through a weighted averaging

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026, LNCS 16768, pp. 277–296, 2027.
https://doi.org/10.1007/978-3-032-30488-9_18

mechanism [3, 4]. The resulting aggregated preference is then used to infer the active user's preference for subsequent recommendation tasks.

To estimate these parameters, loss functions are typically formulated to minimize the discrepancy between observed feedback and the preferences predicted by the model. In most existing studies, this feedback is represented by explicit ratings, which provide a direct numerical indication of user satisfaction. Consequently, the loss functions are primarily designed to align predicted preferences with observed numerical ratings [5, 6]. However, in modern recommender systems, user feedback is also expressed through textual comments, which may appear alongside or even in the absence of ratings [7, 8]. This raises a natural question: how should the loss function be designed and optimized so that predicted preferences are jointly aligned with both observed numerical ratings and observed textual comments?

Based on this research question, the contributions of this paper are threefold:

- We propose a multi-objective loss function for neighbor-based recommender systems to ensure that predicted preferences are jointly aligned with multiple forms of observed feedback.
- We develop a detailed optimization procedure for the proposed objective using stochastic gradient descent (SGD).
- We conduct extensive experiments on benchmark datasets containing both numerical and textual feedback to demonstrate the effectiveness of the proposed approach.

Throughout this paper, the notations used are summarized in Table 1.

Table 1. List of Notations.

Denote	Description
$\mathbb{U}$	Set of users
$\mathbb{I}$	Set of items
$r_{u,i}$	Observed numerical rating given by user u to item i
$t_{u,i}$	Observed textual comment written by user u for item i
$c_{u,i}$	Comment-derived rating inferred from the textual comment $t_{u,i}$
$p_{u,i}$	Predicted preference of user u for item i
$\text{sim}(v, u)$	Similarity between users u and v
$w_{v,u,i}$	Influence weight of user v on user u for item i
$w_{j,i,u}$	Influence weight of item j on item i for user u
$\mathbb{N}_u^i$	Set of users similar to user u who have rated item i
$\mathbb{N}_i^u$	Set of items similar to item i that user u has rated
θ_u	Parameter vector associated with user u
θ_i	Parameter vector associated with item i
α_u	Bias term of user u
α_i	Bias term of item i

(*continued*)

Table 1. (*continued*)

Denote	Description
$\mathbb{R}$	Set of observed numerical ratings
$\mathbb{C}$	Set of observed textual comments

2 Related Work

Let $\mathbb{U}$ denote the set of users and $\mathbb{I}$ denote the set of items. The system collects explicit ratings provided by users, where each rating $r_{u,i} \in \{1,2,3,4,5\}$ represents the preference of user $u \in \mathbb{U}$ for item $i \in \mathbb{I}$. A higher rating indicates a stronger level of satisfaction, while a lower rating reflects weaker preference. The primary objective is to predict the unknown preference $p_{u,i}$ of an active user u for an unseen item i based on the observed ratings. In neighbor-based recommendation models, preference prediction is typically performed through the following steps [1]:

- First, the similarity between the active user u and another user v is computed: $\text{sim}(v, u)$;
- Next, the set of users who have rated item i is identified. From this set, the top users who are most similar to u are selected as neighbors: $\mathbb{N}_u^i$;
- The predicted preference of user u for item i ($p_{u,i}$) is then estimated by aggregating the observed ratings of these neighbors ($r_{v,i}, v \in \mathbb{N}_u^i$) using influence weights ($w_{v,u}, v \in \mathbb{N}_u^i$), as follows:

$$p_{u,i} = \sum_{v \in \mathbb{N}_v^i} w_{v,u}.r_{v,i} \tag{1}$$

Most studies on neighbor-based recommendation models primarily focus on user similarity computation. User similarity can be characterized from multiple perspectives. The most common approach relies on users' past item experiences, typically reflected through numerical ratings [9] or textual comments [7, 8]. Several studies further exploit user interaction behaviors before item consumption, such as viewing items, adding items to the cart, and purchasing activities [10]. Although these behaviors do not explicitly express user preferences, they provide valuable implicit signals that reveal underlying interests. Other approaches leverage item descriptions associated with users' item experiences by applying advanced natural language processing and deep learning techniques [3]. Item categories have also been incorporated as an additional source of similarity information [11]. Furthermore, some studies consider contextual factors, such as temporal patterns of user interactions, to refine the similarity measure [12]. Accurate estimation of user similarity plays a critical role in neighbor-based recommendation models, as it directly determines the quality of the neighbor set and, consequently, the effectiveness of preference prediction.

To estimate the influence weights between neighbors and the active user ($w_{v,u}, v \in \mathbb{N}_u^i$), some studies directly assign weights based on similarity scores, as follows [7, 8,

10–12]:

$$p_{u,i} = \sum_{v \in \mathbb{N}_u^i} \text{sim}(v, u).r_{v,i} \tag{2}$$

However, a more accurate estimation can be achieved by learning these influence weights through a loss function that minimizes the discrepancy between predicted preferences ($p_{u,i}$) and observed ratings ($r_{u,i}$), as follows [3, 13]:

$$p_{u,i} = \sum_{v \in \mathbb{N}_u^i} w_{v,u}.r_{v,i} \underset{w_{v,u}, \forall v \in \mathbb{N}_u^i, \forall u \in \mathbb{U}, \forall i \in \mathbb{I}}{min} \frac{1}{2} \sum_{(u,i) \in \mathbb{R}} \left(r_{u,i} - p_{u,i}\right)^2 \tag{3}$$

where $\mathbb{R}$ denotes the set of observed ratings.

Building upon this formulation, several studies further extend the loss function to estimate item-specific influence weights $w_{v,u,i}$ (instead of $w_{v,u}$), motivated by the observation that a neighbor's influence on the active user may vary across items. However, modeling such fine-grained, item-dependent influence substantially increases the number of parameters in the loss function. To address this issue, [5, 6] proposes a factorized parameterization of influence weights. Specifically, the influence weight associated with a neighbor-user-item triplet $w_{v,u,i}$ is modeled as the inner product between a triplet feature vector $\phi_{v,u,i}$ and a user-specific parameter vector θ_u, as follows:

$$p_{u,i} = \sum_{v \in \mathbb{N}_u^i} w_{v,u,i}.r_{v,i} \; with \; w_{v,u,i} = \phi_{v,u,i}^T.\theta_u \underset{\theta_u, \forall u \in \mathbb{U}}{min} \frac{1}{2} \sum_{(u,i) \in \mathbb{R}} \left(r_{u,i} - p_{u,i}\right)^2 \tag{4}$$

3 Motivation

In real-world recommendation scenarios, different forms of user feedback may be observed after a user experiences an item. Specifically, three common situations can occur as follows:

- A user provides both a numerical rating and a textual comment;
- A user provides only a numerical rating;
- A user provides only a textual comment.

Existing neighbor-based recommendation models typically construct their loss functions solely based on observed numerical ratings. As a result, the optimization process focuses on aligning predicted preferences with rating signals, implicitly assuming that ratings are always available and serve as the sole learning signal. However, this assumption does not hold in many practical settings, where users may choose to leave only textual comments or provide textual comments alongside ratings. Consequently, optimizing prediction models using rating-based objectives alone becomes insufficient. In such cases, predicted preferences should also be consistent with the opinions reflected in users' comments.

Motivated by this limitation, this study develops a Multi-Objective loss function for Neighbor-Based recommendation models, referred to as **MONB**. The proposed formulation enables influence weights to be learned by jointly considering multiple forms of user feedback within a unified optimization framework. Specifically, Sect. 4.1 introduces a neighbor-based recommendation model that integrates both user-oriented and item-oriented perspectives. Building upon this model, Sect. 4.2 formulates the proposed multi-objective loss function for learning influence weights. Finally, Sect. 4.3 presents the optimization procedure in detail.

4 Our Approach

4.1 Neighbor-Based Recommendation Model from User and Item Perspectives

In many studies, neighbor-based recommendation models are implemented from either the user-oriented perspective or the item-oriented perspective. The item-based approach estimates the preference of an active user for an item based on the preferences of similar items that the user has previously interacted with, referred to as item neighbors [3, 11, 13]. Analogous to the user-oriented formulation in Eq. (4), the preference prediction of user u for item i using item neighbors can be formulated as follows [3, 11, 13]:

$$p_{u,i} = \sum_{j \in \mathbb{N}_i^u} w_{j,i,u}.r_{u,j} \; with \; w_{j,i,u} = \phi_{j,i,u}^T.\theta_i \tag{5}$$

where $\mathbb{N}_i^u$ denotes the set of items similar to item i that have been rated by user u, $w_{j,i,u}$ represents the influence weight between item i and its item neighbor j with respect to user u, $\phi_{j,i,u}$ is the feature vector representing the triplet (item neighbor j, item i, user u), and θ_i is the parameter vector associated with item i.

With the development of hybrid recommender systems, multiple perspectives can be combined within a unified framework. A common strategy is to integrate user-based and item-based predictions through an aggregation mechanism. Accordingly, the preference prediction can be expressed as [5, 6]:

$$p_{u,i} = \sum_{v \in \mathbb{N}_u^i} \phi_{v,u,i}^T.\theta_u.r_{v,i} + \sum_{j \in \mathbb{N}_i^u} \phi_{j,i,u}^T.\theta_i.r_{u,j} \tag{6}$$

To further improve prediction accuracy, user and item biases (α_u and α_i) relative to the global average (μ) are incorporated following common practices [3, 5, 6, 12, 13]:

$$\begin{aligned} p_{u,i} = \alpha_u + \alpha_i + \mu + \sum_{v \in \mathbb{N}_u^i} \phi_{v,u,i}^T.\theta_u.\left(r_{v,i} - \alpha_v - \alpha_i - \mu\right) \\ + \sum_{j \in \mathbb{N}_i^u} \phi_{j,i,u}^T.\theta_i.\left(r_{u,j} - \alpha_u - \alpha_j - \mu\right) \end{aligned} \tag{7}$$

In this model, the parameters to be learned include α_u, α_i, θ_u, and θ_i. The feature vectors $\phi_{v,u,i}$ và $\phi_{j,i,u}$ are constructed according to [5], as described in Tables 2, 3.

Table 2. Feature vector for the triplet $\phi_{v,u,i}$ (user neighbor v, user u, item i) [5]

No	Feature description
1	Number of items rated by the user neighbor v
2	Number of users who have rated item i
3	Similarity between the user u and the user neighbor v
4	Average observed rating of item i
5	Average observed rating given by the user neighbor v
6	Standard deviation of observed ratings for item i
7	Standard deviation of observed ratings given by the user neighbor v

Table 3. Feature vector for the triplet $\phi_{j,i,u}$ (item neighbor j, item i, user u) [5]

No	Feature description
1	Number of users who have rated the item neighbor j
2	Number of items rated by the user u
3	Similarity between the item i and the item neighbor j
4	Average observed rating given by the user u
5	Average observed rating of the item neighbor j
6	Standard deviation of observed ratings given by the user u
7	Standard deviation of observed ratings for the item neighbor j

4.2 Multi-objective Loss Function: MONB

Figure 1 illustrates the main idea of our proposed MONB model. Specifically, to learn the parameters above while accounting for multiple feedback scenarios (rating and/or comment), we formulate a multi-objective loss function, referred to as **MONB**, that jointly aligns the predicted preferences $p_{u,i}$ with observed numerical ratings $r_{u,i}$ in the

observed rating set $\mathbb{R}$ and comment-derived ratings $c_{u,i}$ in the observed comment set $\mathbb{C}$:

$$\min_{\theta_u,\theta_i,\alpha_u,\alpha_i,\forall u\in\mathbb{U},\forall i\in\mathbb{I}} \frac{1}{2}\sum_{(u,i)\in\mathbb{R}}\left(r_{u,i} - p_{u,i}\right)^2 + \frac{1}{2}\sum_{(u,i)\in\mathbb{C}}\left(c_{u,i} - p_{u,i}\right)^2$$

$\Leftrightarrow$

$$\begin{aligned}\min_{\theta_u,\theta_i,\alpha_u,\alpha_i,\forall u\in\mathbb{U},\forall i\in\mathbb{I}} \frac{1}{2}\sum_{(u,i)\in\mathbb{R}}\Bigg(& r_{u,i} - \alpha_u - \alpha_i - \mu - \sum_{v\in\mathbb{N}_u^i}\phi_{v,u,i}^T.\theta_u.\left(r_{v,i} - \alpha_v - \alpha_i - \mu\right)\\ &- \sum_{j\in\mathbb{N}_i^u}\phi_{j,i,u}^T.\theta_i.\left(r_{u,j} - \alpha_u - \alpha_j - \mu\right)\Bigg)^2\\ &+\frac{1}{2}\sum_{(u,i)\in\mathbb{C}}\Bigg(c_{u,i} - \alpha_u - \alpha_i - \mu - \sum_{v\in\mathbb{N}_u^i}\phi_{v,u,i}^T.\theta_u.\left(r_{v,i} - \alpha_v - \alpha_i - \mu\right)\\ &- \sum_{j\in\mathbb{N}_i^u}\phi_{j,i,u}^T.\theta_i.\left(r_{u,j} - \alpha_u - \alpha_j - \mu\right)\Bigg)^2\end{aligned} \tag{8}$$

where $c_{u,i}$ is obtained by mapping a user's textual comment into a rating score. This mapping can be implemented with low computational cost using sentiment dictionaries, as described in [14]. With higher computational cost, more accurate scores can be produced by using large language models.

To prevent overfitting, Tikhonov regularization is incorporated:

$$\begin{aligned}\min_{\theta_u,\theta_i,\alpha_u,\alpha_i,\forall u\in\mathbb{U},\forall i\in\mathbb{I}} \frac{1}{2}\sum_{(u,i)\in\mathbb{R}}\Bigg(& r_{u,i} - \alpha_u - \alpha_i - \mu - \sum_{v\in\mathbb{N}_u^i}\phi_{v,u,i}^T.\theta_u.\left(r_{v,i} - \alpha_v - \alpha_i - \mu\right)\\ &- \sum_{j\in\mathbb{N}_i^u}\phi_{j,i,u}^T.\theta_i.\left(r_{u,j} - \alpha_u - \alpha_j - \mu\right)\Bigg)^2\\ &+\frac{1}{2}\sum_{(u,i)\in\mathbb{C}}\Bigg(c_{u,i} - \alpha_u - \alpha_i - \mu - \sum_{v\in\mathbb{N}_u^i}\phi_{v,u,i}^T.\theta_u.\left(r_{v,i} - \alpha_v - \alpha_i - \mu\right)\\ &- \sum_{j\in\mathbb{N}_i^u}\phi_{j,i,u}^T.\theta_i.\left(r_{u,j} - \alpha_u - \alpha_j - \mu\right)\Bigg)^2\\ &+\frac{\lambda}{2}\left(\sum_{u\in\mathbb{U}}\left(\|\theta_u\|^2 + \alpha_u^2\right) + \sum_{i\in\mathbb{I}}\left(\|\theta_i\|^2 + \alpha_i^2\right)\right)\end{aligned} \tag{9}$$

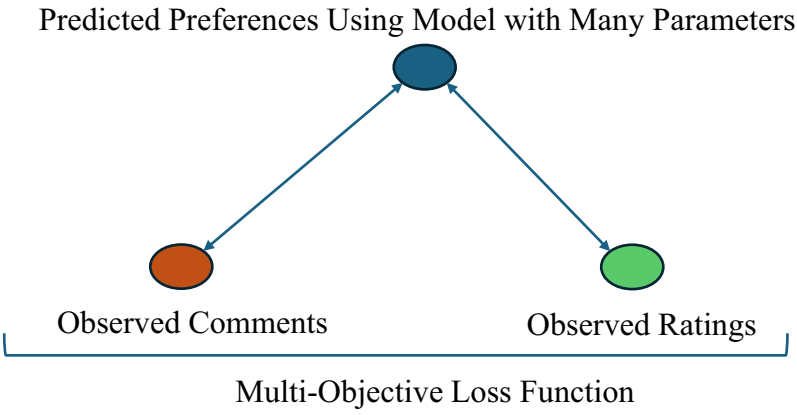


Fig. 1. The proposed MONB loss function

4.3 Optimization via Stochastic Gradient Descent

The loss function is optimized using stochastic gradient descent (SGD), as outlined in Algorithm 1. Specifically, for each observed user–item pair (u, i), the prediction error $e_{u,i}$ is defined according to the available feedback:

- Only a rating is available:

$$e_{u,i} = r_{u,i} - \alpha_u - \alpha_i - \mu - \sum_{v \in \mathbb{N}_u^i} \phi_{v,u,i}^T . \theta_u . \left(r_{v,i} - \alpha_v - \alpha_i - \mu\right) - \sum_{j \in \mathbb{N}_i^u} \phi_{j,i,u}^T . \theta_i . \left(r_{u,j} - \alpha_u - \alpha_j - \mu\right) \tag{10}$$

- Only a comment is available:

$$e_{u,i} = c_{u,i} - \alpha_u - \alpha_i - \mu - \sum_{v \in \mathbb{N}_u^i} \phi_{v,u,i}^T . \theta_u . \left(r_{v,i} - \alpha_v - \alpha_i - \mu\right) - \sum_{j \in \mathbb{N}_i^u} \phi_{j,i,u}^T . \theta_i . \left(r_{u,j} - \alpha_u - \alpha_j - \mu\right) \tag{11}$$

- Both rating and comment are available:

$$\begin{aligned} e_{u,i} = & \left(r_{u,i} - \alpha_u - \alpha_i - \mu - \sum_{v \in \mathbb{N}_u^i} \phi_{v,u,i}^T . \theta_u . \left(r_{v,i} - \alpha_v - \alpha_i - \mu\right) \right. \\ & \left. - \sum_{j \in \mathbb{N}_i^u} \phi_{j,i,u}^T . \theta_i . \left(r_{u,j} - \alpha_u - \alpha_j - \mu\right) \right) \\ & + \left(c_{u,i} - \alpha_u - \alpha_i - \mu - \sum_{v \in \mathbb{N}_u^i} \phi_{v,u,i}^T . \theta_u . \left(r_{v,i} - \alpha_v - \alpha_i - \mu\right) \right. \\ & \left. - \sum_{j \in \mathbb{N}_i^u} \phi_{j,i,u}^T . \theta_i . \left(r_{u,j} - \alpha_u - \alpha_j - \mu\right) \right) \end{aligned} \tag{12}$$

- The model parameters are updated by computing the partial derivatives of the loss function with respect to each parameter:

$$\theta_u \leftarrow \theta_u + \delta\left(e_{u,i}.\sum_{v\in\mathbb{N}_u^i}\phi_{v,u,i}.\left(r_{v,i}-\alpha_v-\alpha_i-\mu\right)-\lambda\theta_u\right) \tag{13}$$

$$\theta_i \leftarrow \theta_i + \delta\left(e_{u,i}.\sum_{j\in\mathbb{N}_i^u}\phi_{j,i,u}\left(r_{u,j}-\alpha_u-\alpha_j-\mu\right)-\lambda\theta_i\right) \tag{14}$$

$$\alpha_u \leftarrow \alpha_u + \delta\left(e_{u,i}-e_{u,i}\sum_{j\in\mathbb{N}_i^u}\phi_{j,i,u}^T.\theta_i-\lambda\alpha_u\right) \tag{15}$$

$$\alpha_i \leftarrow \alpha_i + \delta\left(e_{u,i}-e_{u,i}\sum_{v\in\mathbb{N}_u^i}\phi_{v,u,i}^T.\theta_u-\lambda\alpha_i\right) \tag{16}$$

$$\forall v\in\mathbb{N}_u^i:\alpha_v \leftarrow \alpha_v-\delta\left(\phi_{v,u,i}^T.\theta_u+\lambda\alpha_v\right) \tag{17}$$

$$\forall j\in\mathbb{N}_i^u:\alpha_j \leftarrow \alpha_j-\delta\left(\phi_{j,i,u}^T.\theta_i+\lambda\alpha_j\right) \tag{18}$$

where δ is the learning rate

Algorithm 1: SGD for the Multi-Objective Neighbor-based Model

1. Initialize $\theta_u, \theta_i, \alpha_u, \alpha_i \ \forall u \in \mathbb{U}, \forall i \in \mathbb{I}$
2. Construct the set of observed user–item pairs $\mathbb{H}$, where $\mathbb{R}$ denotes the set of observed ratings and $\mathbb{C}$ denotes the set of observed comments.
3. Repeat until convergence (i.e., the change in the objective function is less than 10^{-6} or the number of iterations reaches 500)
4. Randomly shuffle the observed set $\mathbb{H}$
5. For each observed pair (u,i) in $\mathbb{H}$:
   - If (u,i) belongs to both $\mathbb{R}$ and $\mathbb{C}$
     - Compute $e_{u,i}$ using Eq. (12)
   - If (u,i) belongs only to $\mathbb{R}$:
     - Compute $e_{u,i}$ using Eq. (10)
   - If (u,i) belongs only to $\mathbb{C}$:
     - Compute $e_{u,i}$ using Eq. (11)
   - Update model parameters:
     - θ_u using Eq. (13)
     - θ_i using Eq. (14)
     - α_u using Eq. (15)
     - α_i using Eq. (16)
     - $\forall v \in \mathbb{N}_u^i$ using Eq. (17)
     - $\forall j \in \mathbb{N}_i^u$ using Eq. (18)

Although the model parameters are learned through the optimization procedure described above, the effectiveness of the optimization process can be influenced by the initial values of the user and item parameter vectors θ_u and θ_i (Step 1 in Agorithm 1). Let $\theta_u^{(0)}$ and $\theta_i^{(0)}$ represent their initial representations. In real-world recommender systems, textual comments contain rich semantic information that reflects user preferences and item characteristics. Therefore, instead of using random initialization, we introduce a review-aware strategy to initialize the parameter vectors of users and items based on textual comments associated with positive user preferences, as shown in Fig. 2.

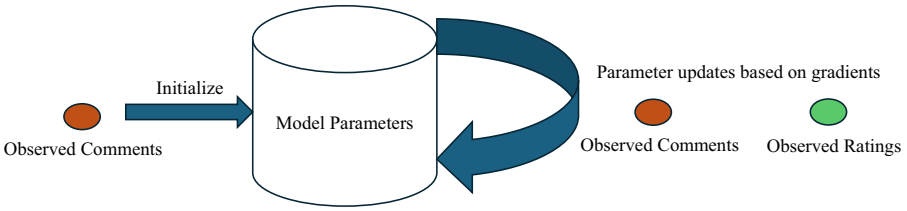


Fig. 2. Initialization strategy based on textual comments for users and items

Specifically, for each user u, we identify the set of items that the user positively prefers, denoted by $\mathbb{S}_u$. An item i is considered preferred if the corresponding textual comment expresses positive sentiment, i.e., the comment-derived rating $c_{u,i}$ exceeds a liking threshold δ. Formally,

$$\mathbb{S}_u = \{i | (u, i) \in \mathbb{C} \wedge c_{u,i} > \delta\} \tag{19}$$

where $\mathbb{C}$ denotes the set of user–item pairs for which textual comments are available. Let $t_{u,i}$ denote the embedding vector extracted from the textual comment written by user u for item i. This embedding can be obtained using large language models. The initial representation of user u is then constructed by aggregating the embeddings of the comments associated with the preferred items:

$$\theta_u^{(0)} = \frac{1}{|\mathbb{S}_u|} \sum_{i \in \mathbb{S}_u} t_{u,i} \tag{20}$$

Similarly, for each item i, we consider the set of users who express positive preference toward that item, denoted by $\mathbb{S}_i$:

$$\mathbb{S}_i = \{u | (u, i) \in \mathbb{C} \wedge c_{u,i} > \delta\} \tag{21}$$

The initial representation of item i is obtained by aggregating the embeddings of the corresponding textual comments:

$$\theta_i^{(0)} = \frac{1}{|\mathbb{S}_i|} \sum_{i \in \mathbb{S}_i} t_{u,i} \tag{22}$$

4.4 System Implementation

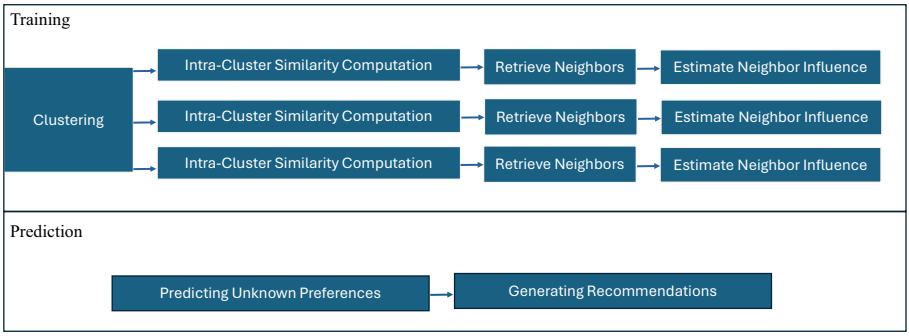


Fig. 3. The training and prediction phases in the proposed system

Based on the above design, the proposed system is implemented in two main phases, namely training and prediction, as shown in Fig. 3:

- In the training phase, the similarities between user pairs and between item pairs are first computed to construct the corresponding neighbor sets for recommendation. Next, the proposed objective function is optimized to estimate the parameter vectors associated with users and items, as well as the corresponding bias terms.
- In the prediction phase, the preferences of an active user for candidate items are inferred by combining the learned parameters with the user and item neighbor sets obtained during training.

However, when the number of users and items is large, the computational cost increases substantially in both similarity computation and parameter optimization. In particular, computing pairwise similarities for all users and all items leads to a large search space, while optimizing the objective function requires repeated evaluation of the error terms $e_{u,i}$ for observed user-item pairs (u, i). To address the first implementation issue, users and items are clustered before similarity computation so that the similarity search can be restricted to a smaller and more relevant subset. This clustering strategy takes advantage of the initial representations of users and items, denoted by $\theta_u^{(0)}$ and $\theta_i^{(0)}$, which are learned from review embeddings and are expected to capture meaningful semantic characteristics of users and items.

For the optimization stage, the most expensive computation lies in evaluating the error term $e_{u,i}$ (Eqs. (10–12)) for each observed user-item pair (u, i), especially when both rating-based and comment-based feedback are incorporated into the learning process. According to the objective function and the SGD-based optimization procedure in Sect. 4.3, each error term $e_{u,i}$ depends not only on the observed feedback of the pair (u, i) itself, but also on two aggregated neighbor components: one from the user neighbor set $\mathbb{N}_u^i$ and one from the item neighbor set $\mathbb{N}_i^u$. As a result, repeatedly evaluating these summation terms over large datasets becomes a major computational bottleneck. To improve scalability, this computation is parallelized using the MapReduce model on

the Hadoop platform. More specifically, for a user-item pair (u, i), the corresponding error term can be decomposed into one constant component $a_{u,i}$ and two summation components $f_{u,i}$ and $b_{u,i}$:

$$a_{u,i} = \left(r_{u,i} - \alpha_u - \alpha_i - \mu\right) \tag{23}$$

$$f_{u,i} = \sum_{v \in \mathbb{N}_u^i} \phi_{v,u,i}^T . \theta_u . \left(r_{v,i} - \alpha_v - \alpha_i - \mu\right)$$

$$b_{u,i} = \sum_{j \in \mathbb{N}_i^u} \phi_{j,i,u}^T . \theta_i . \left(r_{u,j} - \alpha_u - \alpha_j - \mu\right)$$

This decomposition is suitable for distributed computation because each component can be evaluated independently before being aggregated to form the final error value. To compute this error efficiently, we design a MapReduce procedure consisting of three main stages: input split, map, and reduce. In the input split stage, for each target pair (u, i), the required data are transformed into neighbor records. Two types of records are generated:

- The first type corresponds to each user neighbor $v \in \mathbb{N}_u^i$, where each record contains $(u, i, v, \phi_{v,u,i}, \theta_u, r_{v,i}, \alpha_v, \alpha_v, \mu)$.
- The second type corresponds to each item neighbor $j \in \mathbb{N}_i^u$, where each record contains $(u, i, j, \phi_{j,i,u}, \theta_i, r_{u,j}, \alpha_u, \alpha_j, \mu)$.

The constant values $(r_{u,i}, \alpha_u, \alpha_i, \mu)$ are also made available to the processing nodes, either directly within the input metadata or through a distributed cache. By splitting the computation into these fine-grained records, the system enables each worker node to process one local summand independently.

In the map stage, each mapper processes one neighbor record and computes one local contribution to the error term. If the input record corresponds to a user neighbor v, the mapper computes:

$$m_v = \phi_{v,u,i}^T . \theta_u . \left(r_{v,i} - \alpha_v - \alpha_i - \mu\right) \tag{24}$$

and emits the key-value pair:

$$key = (u, i); value = (user, m_v) \tag{25}$$

Similarly, if the input record corresponds to an item neighbor j, the mapper computes:

$$m_j = \phi_{j,i,u}^T . \theta_i . \left(r_{u,j} - \alpha_u - \alpha_j - \mu\right) \tag{26}$$

and emits:

$$key = (u, i); value = (\text{"}item\text{"}, m_j) \tag{27}$$

Optionally, one mapper may also emit the constant term $a_{u,i}$ (see Eq. (23)) using the same key.

During the shuffle and sort stage, all intermediate values having the same key (u, i) are grouped and routed to the same reducer. Consequently, all partial contributions associated with the target pair are collected at one location for final aggregation. In the reduce stage, the reducer initializes two accumulators, namely $f_{u,i}$ and $b_{u,i}$ (See Eq. (23)), to zero. It then scans all received values associated with the key (u, i). If an incoming value is labeled `"user"`, its contribution is added to $f_{u,i}$. If it is labeled `"item"`, its contribution is added to $b_{u,i}$. After all partial values have been aggregated, the reducer obtains the final error value as follows:

$$e_{u,i} = a_{u,i} - f_{u,i} - b_{u,i} \tag{28}$$

5 Experiment

5.1 Experiment Setup

In this section, we conduct experiments using a series of neighbor-based recommendation models, in which each stage is implemented as follows:

- Similarity computation: RaReSi [8], MFPS [9]
- Influence weight estimation:
- Using similarity as influence weights: SBP [8, 9]
- Constructing a loss function optimized for ratings: OSRP [5] and ROSRP [6]
- Constructing a joint loss function optimized for both ratings and comments: MONB

For hyperparameter settings, the neighbor set size (for both user neighbors and item neighbors) is varied from 30 to 100. The learning rate (δ) and regularization coefficient (λ) are set to 0.001 and 0.01, respectively. To incorporate textual comments into the learning process, user comments are converted into rating scores ($c_{u,i}$ in Eq. (8)) and into embedding vectors ($t_{u,i}$ in Eq. (20)) using GPT-4. The liking threshold (δ in Eqs. (19, 21)) is defined as the midpoint of the rating scale, which is 3 on a scale from 1 to 5.

5.2 Dataset

To align with the characteristics of this study, we selected publicly available benchmark datasets that contain both numerical ratings and textual comments, as summarized in Table 4. The datasets were split into training and test sets using a 70–30 ratio. The splitting process was conducted randomly three times to generate three different train–test partitions. The final results reported in the following sections correspond to the average performance across these three splits.

Table 4. Dataset description.

Dataset	#Users	#Items	#Comments and ratings
Amazon Baby	19,445	7,050	160,792
Amazon Beauty	22,365	12,101	198,502
Amazon Gourmet Foods	14,601	8,713	150,289

5.3 Measure

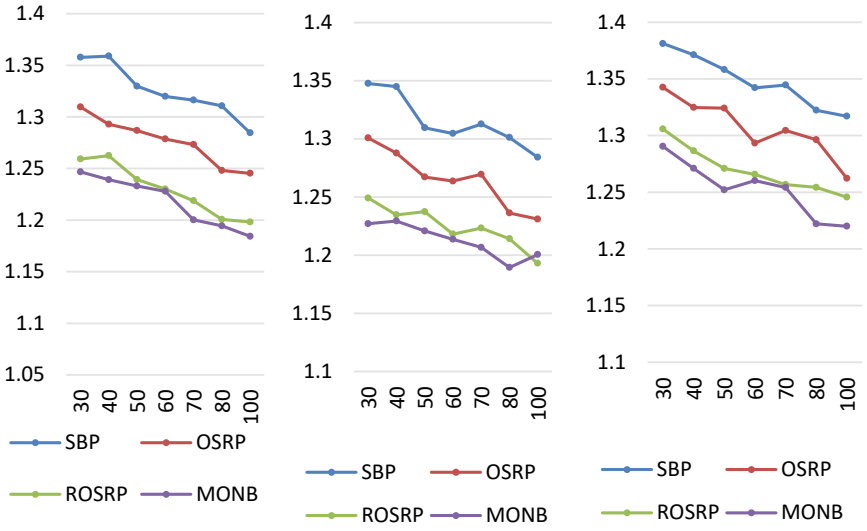
A recommender system is considered effective when it can accurately estimate user preferences, resulting in lower RMSE values. The RMSE is computed as follows:

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in\mathbb{T}} \left(p_{u,i} - r_{u,i}\right)^2}{|\mathbb{T}|}} \tag{29}$$

where $\mathbb{T}$ is the test set.

5.4 Experiment Result and Discussion

Figure 4 and Table 5 respectively present the comparison results of the experimental models on each dataset and the overall average performance. The proposed model (MONB) achieves better results across most neighbor set sizes, indicating that the influence weights are estimated more accurately than in the competing models. On average across all datasets, the proposed model reduces RMSE by 1.48% compared with the best-performing baseline (ROSRP).



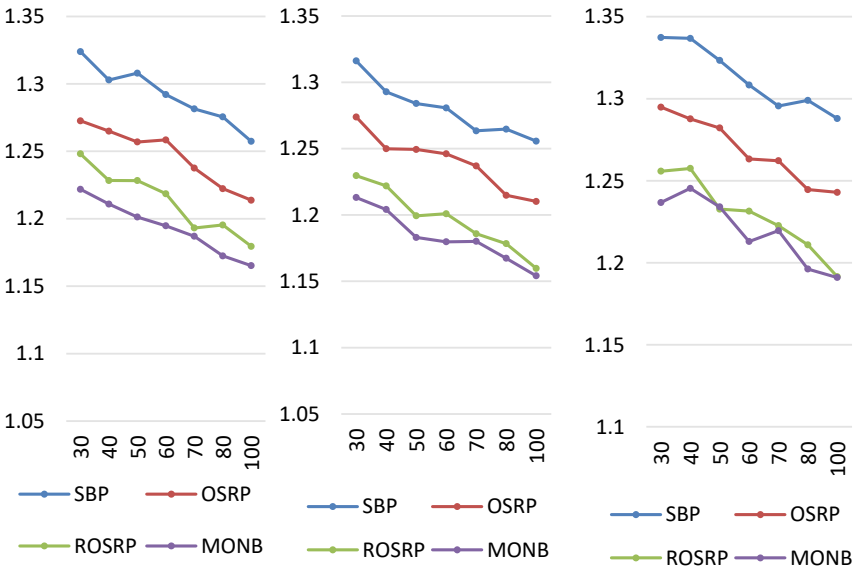


Fig. 4. RMSE comparison of SBP, OSRP, ROSRP, and MONB across varying neighbor set sizes using MFPS and RaReSi for user similarity computation. The horizontal axis denotes the number of selected neighbors (30–100), while the vertical axis denotes RMSE.

Table 5. Average RMSE across all datasets using two similarity measures, MFPS and RaReSi.

# Neighbors	30	40	50	60	70	80	100
SBP	1,33399	1,29461	1,28811	1,27959	1,27802	1,26557	1,26111
OSRP	1,27907	1,28467	1,27778	1,26724	1,26399	1,25373	1,23424
ROSRP	1,25794	1,25857	1,23467	1,22747	1,21675	1,21190	1,21095
MONB	1,23934	1,23331	1,22072	1,21487	1,20796	1,19032	1,18359

In addition to evaluating RMSE over the entire user population, we further focus on a subset consisting of one-third of users with the smallest number of observed ratings in each dataset. As illustrated in Tables 6, 7 and 8, the proposed model (MONB) demonstrates a clear performance advantage over the competing models in this setting. This improvement can be explained as follows. When the number of observed ratings is limited, users may not have sufficient experience to consistently express their preferences through rating scales. In such cases, optimizing the model solely based on ratings, as in prior studies, may lead to noisy estimates of influence weights. By incorporating textual comments into the optimization process, the proposed model provides additional preference signals, thereby improving prediction accuracy under sparse-rating conditions.

To strengthen the findings, we conducted statistical testing using the t-test. The samples were constructed from user-level RMSE values, resulting in a sufficiently large sample size to ensure the reliability of the test. As shown in Table 9, most comparisons produced p-values lower than 0.05, while the mean RMSE of the proposed model was better than those of the competing models. These results confirm that the proposed model achieves statistically significant improvements over the baseline models.

Table 6. Average RMSE under two user similarity measures (RaReSi and MFPS) for one-third of users with the smallest number of observed ratings on the Amazon Baby dataset.

# Selected Neighbors	30	40	50	60	70	80	100
SBP	1,49661	1,47696	1,47881	1,45828	1,44328	1,43825	1,43482
OSRP	1,43940	1,42754	1,43389	1,41330	1,41364	1,38601	1,38271
ROSRP	1,40893	1,40112	1,38418	1,37516	1,36225	1,3544	1,32815
MONB	1,37702	1,38532	1,36566	1,35621	1,3522	1,33957	1,31706

Table 7. Average RMSE under two user similarity measures (RaReSi and MFPS) for one-third of users with the smallest number of observed ratings on the Amazon Beauty dataset.

# Selected Neighbors	30	40	50	60	70	80	100
SBP	1,49378	1,45385	1,45273	1,44372	1,42879	1,42886	1,42021
OSRP	1,44198	1,41511	1,40561	1,41102	1,40380	1,37132	1,37100
ROSRP	1,39677	1,38782	1,36225	1,36331	1,33484	1,33781	1,30488
MONB	1,37219	1,35404	1,34101	1,33215	1,33151	1,31756	1,29738

Table 8. Average RMSE under two user similarity measures (RaReSi and MFPS) for one-third of users with the smallest number of observed ratings on the Amazon Gourmet Foods dataset.

# Selected Neighbors	30	40	50	60	70	80	100
SBP	1,51115	1,51142	1,49702	1,47706	1,46615	1,46510	1,46023
OSRP	1,46452	1,44913	1,44839	1,42943	1,41968	1,40750	1,41021
ROSRP	1,41375	1,41501	1,39253	1,39016	1,37588	1,37448	1,35498
MONB	1,40320	1,40270	1,39763	1,37211	1,37957	1,34602	1,34522

Table 9. Statistical comparison using the t-test on user-level RMSE across all experimental datasets.

Experiment model	p-value	Sample mean
MONB $>>$ SBP	0,0002	1,22778 $>>$ 1,32950
MONB $>>$ OSRP	0,0007	1,22778 $>>$ 1,29276
MONB $>>$ ROSRP	0,0106	1,22778 $>>$ 1,24125
MFPS is used for user similarity computation		
MONB $>>$ SBP	0,0001	1,19861 $>>$ 1,29460
MONB $>>$ OSRP	0,0004	1,19861 $>>$ 1,24769
MONB $>>$ ROSRP	0,0083	1,19861 $>>$ 1,21286
RaReSi is used for user similarity computation		

Table 10. Ablation analysis based on average RMSE across all experimental datasets

Method	Average RMSE across experimental datasets	p-value
Full MONB	1,22778	
DeepSeekScore	↑0,02252	0,0004
GeminiScore	↑0,02033	0,0058
SentiDicScore	↑0,07380	0,0051
RemoveUser	↑0,06539	0,0093
RemoveItem	↑0,00313	0,0169
DeepSeekEmbedding	↑0,00922	0,0001
GeminiEmbedding	↑0,00801	0,0079
Random	↑0,00409	0,0001
MFPS is used for user similarity computation		
Full MONB	1,98616	
DeepSeekScore	↑0,02664	0,0009
GeminiScore	↑0,00338	0,0036
SentiDicScore	↑0,07024	0,0018
RemoveUser	↑0,02992	0,0104
RemoveItem	↑0,01524	0,0207
DeepSeekEmbedding	↑0,00450	0,0006
GeminiEmbedding	↑0,00211	0,0039
Random	↑0,01700	0,0001
RaReSi is used for user similarity computation		

Finally, we conducted ablation analyses to examine the contribution of each component in the proposed model. Specifically, we removed the user-oriented module (RemoveUser) and the item-oriented module (RemoveItem) to evaluate their individual impacts on recommendation performance. In addition, we experimented with several alternative approaches for converting textual comments into rating scores, replacing the original GPT-based scoring mechanism. These alternatives include sentiment dictionaries (SentiDicScore) [14], Gemini (GeminiScore), and DeepSeek (DeepSeekScore).

For the parameter initialization stage, the GPT-based embeddings used in the original model were also replaced with random initialization (RanDom), as well as embeddings generated by Gemini (GeminiEmbedding) and DeepSeek (DeepSeekEmbedding), to assess the influence of different language models on the initialization of user and item representations. As presented in Table 10, all modified variants resulted in higher RMSE compared with the original model, with p-values below 0.05, indicating statistically significant performance degradation. These findings demonstrate that the components

of the proposed model complement each other effectively and that their integration plays a crucial role in achieving the overall performance improvements.

6 Conclusion

This paper proposes a multi-objective loss function for optimizing neighbor-based recommender systems by jointly learning from numerical ratings and textual comments. The proposed formulation extends conventional rating-based objectives, enabling the model to better estimate influence weights. To learn the model parameters effectively, we design a detailed stochastic gradient descent (SGD) procedure that accommodates multiple feedback scenarios, including rating-only, comment-only, and combined cases. Experiments conducted on three benchmark datasets containing both ratings and textual comments show that the proposed model achieves lower RMSE than existing models. The improvements are particularly evident for users with sparse rating histories, where textual comments provide complementary preference signals. In future work, we plan to investigate adaptive weighting mechanisms that dynamically balance the contributions of rating and comment objectives instead of treating them equally. We also aim to extend the proposed model to incorporate implicit feedback and temporal dynamics to model evolving user preferences. In addition, we will explore more advanced approaches for transforming textual comments into rating scores to further enhance the robustness of the model.

Acknowledgments. This research is funded by Vietnam National University, Ho Chi Minh City (VNU-HCM) under grant number B2026–18-13.

References

1. Raza, S., et al.: A comprehensive review of recommender systems: transitioning from theory to practice. Comput. Sci. Rev. **59**, 100849 (2026)
2. Zhang, Y., Zhang, Y., Zhang, Y., Sang, L., Yang, Y.: Unveiling contrastive learning's capability of neighborhood aggregation for collaborative filtering. In: Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1985–1994 (2025)
3. Ha, D.T.T., Nam, L.N.H.: A content-aware collaborative filtering framework for video recommender systems. In: IFIP International Conference on Artificial Intelligence Applications and Innovations, pp. 352–366. Springer, Cham (2025)
4. Xie, Z., et al.: Neighborhood-based collaborative filtering for conversational recommendation. In: Proceedings of the 18th ACM Conference on Recommender Systems, pp. 1045–1050 (2024)
5. Nam, L.N.H.: Towards comprehensive approaches for the rating prediction phase in memory-based collaborative filtering recommender systems. Inf. Sci. **589** (2022)
6. Vy, H.T.H., Pham-Nguyen, C.: Integrating textual reviews into neighbor-based recommender systems. Expert Syst. Appl. **249**, 123648 (2024)
7. Kuo, R.J., Li, S.S.: Applying particle swarm optimization algorithm-based collaborative filtering recommender system considering rating and review. Appl. Soft Comput. 110038 (2023)

8. Vy, H.T.H., et al.: RaReSi: an approach combining ratings and reviews to measure user similarity in neighbor-based recommender systems. In: International Conference on Computational Collective Intelligence, pp. 17–29. Springer, Cham (2023)
9. Vy, H.T.H., Gia, H.T., Hang, V.T.M., Pham-Nguyen, C., Hoai Nam, L.N.: A multi-factor approach to measure user preference similarity in neighbor-based recommender systems. In: DATA, pp. 532–539 (2023a)
10. Park, S.H., Kim, K.: Collaborative filtering recommendation system based on improved Jaccard similarity. J. Ambient. Intell. Humaniz. Comput. **14**(8), 11319–11336 (2023)
11. Nilashi, M., Ibrahim, O., Bagherifard, K.: A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques. Expert Syst. Appl. **92**, 507–520 (2018)
12. Nhung, N., Nguyen, D., Hong, T., Vu, T.: Approaches for extending recommendation models for food choices in meals. In: Proceedings of the 16th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, pp. 113–121 (2024)
13. Aggarwal, C.C.: Neighborhood-based collaborative filtering. In: Recommender Systems: The Textbook, pp. 29–70. Springer, Cham (2016)
14. Shen, R.P., Zhang, H.R., Yu, H., Min, F.: Sentiment based matrix factorization with reliability for recommendation. Expert Syst. Appl. **135**, 249–258 (2019)