JST: Smart Systems and Devices
Volume 33, Issue 1, January 2023, 034-042

Incorporating User Item Similarity in Hybrid Neighborhood-Based Recommendation Systems

Tan Nghia Duong*[*], *Truong Giang Do*, *Tuan Nghia Cao
School of Electronics and Telecommunications, Hanoi University of Science and Technology, Ha Noi, Vietnam
[*]Corresponding author email: nghia.duongtan@hust.edu.vn

Abstract

Recommendation systems have been developed in many domains to help users with information overload from the large volume of online multimedia content by providing them with appropriate options. Recently developed hybrid recommendation systems require a large amount of data to understand users' interests and give appropriate suggestions. However, several internet privacy issues make users skeptical about sharing their personal information with online service providers, limiting the potential of these systems. The study in this paper introduces various novel methods utilizing the baseline estimate to learn user interests in specific item features from their past interactions. Subsequently, extracted user feature vectors are implemented to estimate the user-item correlations, providing an additional fine-tuning factor for neighborhood-based collaborative filtering systems. Comprehensive experiments show that utilizing the user-item similarity scores in the rating prediction task can improve the accuracy of hybrid neighborhood-based systems by at least 2.11% compared to traditional methods while minimizing the need for tracking users' digital footprints.

Keywords: Collaborative filtering, data mining, neighborhood-based, recommendation system.

1. Introduction

The continuously accelerated growth of communication technology and data storage in the past decades has benefited customers with an enormous amount of online multimedia content, creating billion-dollar industries. Following this evolution, recommendation systems (RSs) have been widely developed to automatically help users to filter redundant information and suggest only suitable products that fit their needs. Such systems are used in a variety of domains and have become a part of our daily online experience [1].

RSs are commonly classified into three main types: the content-based technique, the collaborative filtering technique, and the hybrid technique. The *content-based* approach learns to recommend items that are similar to the ones that a user liked based on the item features. The main weakness of this approach is the lack of available and reliable metadata associated with items. Meanwhile, the *collaborative filtering* (CF) approach only relies on users' interaction history which can be either explicit or implicit feedback. CF systems can be divided into two major categories: i) neighborhood-based models which focus on computing the correlation between items or users using rating information [2], and ii) matrix factorization models which could explore the latent factors connecting items to users in order to make accurate recommendations [1, 3, 4]. Recently, deep learning has also been proven as a potential approach for implementing CF system by learning the hidden relationships in user interactions [5, 6]. However, it is often the case that there is not enough transaction data to make accurate recommendations for a new user or item. To tackle this *cold-start* problem, *hybrid methods* are proposed by combining auxiliary information into CF models [7].

In the interest of the hybrid approach and its advantages, our study attempts to improve typical neighborhood-based RSs utilizing available content-related knowledge. The main contributions of this work are summarized as follows:

- Introducing new methods to represent user preference via combining the user's interaction data and item's content-based information, which helps to estimate the similarity between a user and an item;
- Integrating the user-item similarity into the baseline estimate of neighborhood-based RSs to provide more precise recommendations, surpassing competitive hybrid models.

The remainder of this paper is organized as follows. Section 2 reviews the basic knowledge of neighborhood-based CF systems including hybrid models. Detailed descriptions of our proposed methods are presented in Section 3. Section 4 gives experimental results and in-depth analysis. At last, we conclude this study in Section 5.

ISSN: 2734-9373
https://doi.org/10.51316/jst.163.ssad.2023.33.1.5
Received: January 4, 2022; accepted: October 20, 2022

2. Preliminaries

In this paper, u, v denote users and i, j denote items. r_{ui} denotes the rating by user u for item i where high values indicate a strong preference and all the (u, i) pairs are stored in the set $\mathbb{K} = \{(u, i) | r_{ui} \text{ is known}\}$. Meanwhile, $R(u)$ denotes the set of all items rated by user u. In the rating prediction task, the objective is to predict unknown rating $\hat{r}_{ui}$ where user u has not rated item i yet.

Popular neighborhood-based CF techniques for the rating prediction task and an existing hybrid variant are briefly reviewed as follows.

2.1. Neighborhood-Based Models

The neighborhood-based approach is one of the most popular techniques in CF, which is only based on the similarity between users or items to give recommendations. There are two methods for implementing neighborhood-based CF models: i) user-oriented model which predicts a user's preference based on similar users, and ii) item-oriented model which finds similar items to the item a user liked and recommends these items to her.

Of the two methods, the latter introduced in [2] has become dominant due to its superior accuracy and its capability of providing a rational explanation for recommendations [1]. Therefore, our implementations in this work adopt the item-oriented approach as the baseline model.

The fundamental of neighborhood-based models is similarity measure. As illustrated in Fig. 1, by computing the similarity degree s_{ij} between all pairs of items i and j using popular similarity measures such as Cosine similarity (Cos) or Pearson Correlation Coefficients (PCC), we can identify the set of k neighbors $\mathbb{S}^k(i, u)$ which consists of k most similar items to i rated by user u.

The most straightforward method of predicting the rating of user u for item i is a weighted average of the ratings of similar items as follows:

$$\hat{r}_{ui} = \frac{\sum_{j \in \mathbb{S}^k(i,u)} s_{ij}\, r_{uj}}{\sum_{j \in \mathbb{S}^k(i,u)} s_{ij}} \tag{1}$$

Even though (1) can capture the user-item interactions, much of the observed ratings are due to the bias effects associated with either users or items, independently of their interactions. In detail, some items usually receive higher ratings than others, and some users tend to give higher ratings than others. To predict $\hat{r}_{ui}$, kNNBaseline model [8] also takes the bias effect associated with either users or items into account by adding the baseline estimate to the weighted average of the ratings of similar items as follows:

$$\hat{r}_{ui}^{kNNBaseline} = b_{ui} + \frac{\sum_{j \in \mathbb{S}^k(i,u)} s_{ij}(r_{uj} - b_{uj})}{\sum_{j \in \mathbb{S}^k(i,u)} s_{ij}} \tag{2}$$

where $b_{ui} = \mu + b_u + b_i$ denotes the baseline estimate, μ denotes the mean of overall ratings, b_u and b_i correspond to the bias of user u and item i, respectively, which can be trained using popular optimization algorithms such as Stochastic Gradient Descent (SGD) or Alternating Least Squares (ALS).

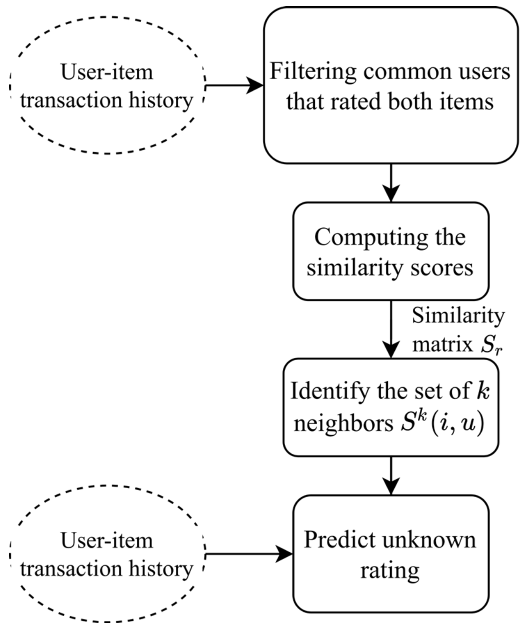
User-item transaction history
Filtering common users that rated both items
Computing the similarity scores
Similarity matrix S_r
Identify the set of k neighbors $S^k(i, u)$
User-item transaction history
Predict unknown rating

Fig. 1. The simplified flow diagram of a kNNBaseline system

2.2. Integrating Content-Based Information into Neighborhood-Based Models

The study in [7] showed that the sparsity of the rating matrix could yield an inaccurate similarity score between two items that share only a few common users. Furthermore, filtering common users who rated both items to calculate the similarity score is a time-consuming task due to a large number of users. To address these problems, novel similarity measures were proposed using item content-based information to modify the conventional kNNBaseline model into a hybrid system. The flow graph of an item-based hybrid kNNBaseline system is illustrated in Fig. 2, where the "Filtering common users that rated both items" step in conventional kNNBaseline system (Fig. 1) is no longer necessary.

Assume that each item i is characterized by a feature vector $\boldsymbol{q}_i = \{q_{i1}, q_{i2}, \ldots, q_{if}\} \in \mathbb{R}^f$ where f is the number of features, which is stored in matrix $\boldsymbol{Q} \in \mathbb{R}^{n \times f}$. The value of each element encodes how strong an item exhibits particular properties. The similarity score s_{ij} between movies i and j is calculated as follows [7, 9]:

$$s_{ij}^{\mathrm{Cos}^{\mathrm{content}}} = \frac{\sum_{k=1}^{f} q_{ik} q_{jk}}{\sqrt{\sum_{k=1}^{f} q_{ik}^2}\sqrt{\sum_{k=1}^{f} q_{jk}^2}} \quad (3)$$

or

$$s_{ij}^{\mathrm{PCC}^{\mathrm{content}}} = \frac{\sum_{k=1}^{f}(q_{ik} - \overline{\boldsymbol{q}}_i)(q_{jk} - \overline{\boldsymbol{q}}_j)}{\sqrt{\sum_{k=1}^{f}(q_{ik} - \overline{\boldsymbol{q}}_i)^2}\sqrt{\sum_{k=1}^{f}(q_{jk} - \overline{\boldsymbol{q}}_j)^2}} \quad (4)$$

where $\overline{\boldsymbol{q}}_i$ and $\overline{\boldsymbol{q}}_j$ are the mean of feature vectors $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$, respectively. Hereafter, the hybrid kNN-Baseline model using one of these similarity measures is referred to as kNNContent.

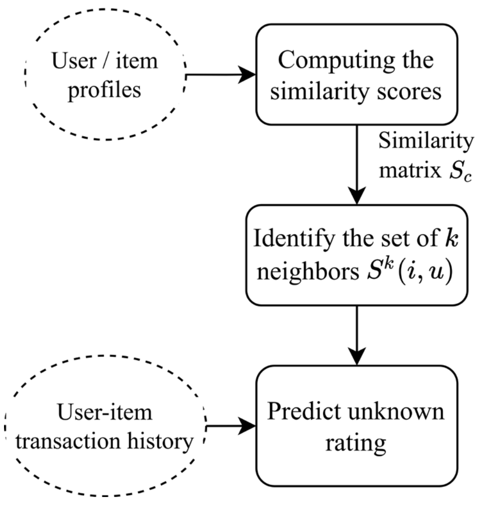
User / item profiles
Computing the similarity scores
Similarity matrix S_c
Identify the set of k neighbors $S^k(i, u)$
User-item transaction history
Predict unknown rating

Fig. 2. The simplified flow diagram of a kNNContent system.

While also sharing the same predicting method compared to both kNNBaseline and kNNContent [7] models, our proposed model described in Section 3.2 extends the baseline estimate formula b_{ui} in (2). The content-based information is also not limited to item profiles as in [7] but is enhanced with the user's interest estimated using several novel methods.

3. Proposed Systems

So far, neighborhood-based CF models have successfully applied the item-item similarity exploited via rating information and the available item features. In contrast, the knowledge about user-user correlation finds it difficult to be deployed in practical applications due to its modest performance and high memory requirement [1]. Nonetheless, the interest of a user in an individual characteristic of each item, to our best knowledge, lacks careful consideration. One of the main reasons is that the user-item correlation is commonly defined as a similarity degree between a user's interest in individual item features and an item feature vector, which requires a customer to provide his personal preferences. In reality, it is impractical due to a variety of data privacy concerns.

This study first tackles this problem by introducing various novel methods to represent a user preference in the form of a vector, utilizing her past interactions with items in the system and the feature vectors of those items. After gathering reliable information about users, we then propose a modification to the baseline estimate of kNNBaseline model and its variants by integrating the user-item similarity scores, which boosts the precision of the conventional kNNBaseline model.

3.1. Estimating User Interests for User-Item Similarity Measure

In RSs, there are two main sources of information to learn user interests and give recommendations: the interaction records of users on the system and the item content information. User personal data, however, is not included in public datasets for research due to the risk of exposing user identities. Therefore, there is rarely any data or statistic that directly specifies user interest in each item feature. This lack of information has limited the potential performance of RS in practice. In this section, we present 3 different methods to characterize a user's interest in item features based on given ratings of the user and metadata of the items that he rated.

The most straightforward approach to estimate a user's interest is via a weighted average of the feature vectors $\boldsymbol{q}_i$ of items that he rated by the normalized ratings as follows:

$$\boldsymbol{p}_u^{\mathrm{norm}} = \frac{\sum_{i \in R(u)} r_{ui}^{\mathrm{norm}} \cdot \boldsymbol{q}_i}{|R(u)|} \quad (5)$$

where r_{ui}^{norm} is the rating of user u for item i which has been normalized to the range of $[0,1]$. As a result, the *normalized* feature vector $\boldsymbol{p}_u^{\mathrm{norm}}$ of user u has the same dimension and range of element values as an item feature vector $\boldsymbol{q}_i$. More importantly, by using (5), each user is currently described in an explainable way: elements with higher values indicate that the user has a greater preference for the corresponding item attributes and vice versa.

Although this method creates a simple shortcut to understand user preferences, all users are treated in the same way: all users' ratings are normalized using the same minimum and maximum rating values of the system while, in practice, users have a variety of tendencies of rating an item. For example, easy-going people often rate movies a little higher than the average, and conversely, strict users often give lower scores than the others. That means if two users have conflicting views after watching a movie but accept to give a 3-star rating for that movie, for example, then the system described in (5) will implicitly assume they have the same weight of opinion. This problem leads to several researches taking the user and item biases into account, which have a considerable impact on CF systems [4, 8].

Therefore, a modification of (5) incorporating the effect of biases is proposed as follows:

$$\boldsymbol{p}_u^{\text{biased}} = \frac{\sum_{i\in R(u)} z_{ui}\cdot \boldsymbol{q}_i}{\sum_{i\in R(u)} |z_{ui}|} \quad (6)$$

where $z_{ui} = r_{ui} - b_{ui}$ denotes the residual rating.

In more detail, (6) applies z_{ui} as weighting factors to the corresponding item feature vectors, which helps to eliminate the restrictions of r_{ui}^{norm}. The resulting *biased* user feature vector $\boldsymbol{p}_u^{\text{biased}}$ has its elements in the value range of $[-1, +1]$, where -1 / $+1$ indicates that he hates/loves the respective item attribute, and 0 is neutral preference. It is expected that $\boldsymbol{p}_u^{\text{biased}}$ could measure the user interest in each item attribute more precisely than its normalized version $\boldsymbol{p}_u^{\text{norm}}$.

However, both of the above methods treat all items equally in profiling a user's interest. For example, the scores of "Titanic" and "Mad Max" for the *romantic* genre are 0.90 and 0.05, respectively. Assume that Janet's normalized ratings for these movies are $\tilde{r}_{\text{Janet,Titanic}} = 0.7$ and $\tilde{r}_{\text{Janet,Madmax}} = 0.72$, which are almost identical. Thus, the *romantic* genre score of Janet calculated by (5) is quite low: $(0.7 \times 0.9 + 0.72 \times 0.05)/2 = 0.333$. The fact that "Mad Max" has almost no romantic element in the movie does not mean that Janet doesn't like *romantic* movies since she also loves "Titanic", one of the most epic romance movies in history. Equation (6) also encounters the same problem of taking features that the movie does not exhibit (indicated by low scores) into account. This might lead to severe misunderstanding on learning the interests of users in a variety of circumstances.

This problem can be solved by alleviating the influence of low score features whilst primarily focusing on features with high values. Accordingly, the simplest method is to use the scores themselves as the weights in parallel with normalized ratings to estimate user feature vectors so that low score features will equal themselves out of the final user feature vectors. The biased feature vector of user u weighted by the item feature vector can be formulated as follows:

$$\boldsymbol{p}_u^{\text{w-biased}} = \frac{\sum_{i\in R(u)} z_{ui}\cdot \boldsymbol{q}_i^2}{\sum_{i\in R(u)} |z_{ui}|\cdot \boldsymbol{q}_i} \quad (7)$$

Specifically for the above example, the interest score of Janet for the *romantic* genre calculated using (7) is equal to 0.854, which is much more reasonable than measuring the affection of a user for a specific kind of genre based on items that are not relevant to that genre.

From the user feature vector $\boldsymbol{p}_u$ calculated using one of the three above methods, the relevance between a user and an item can be effectively evaluated using common similarity measures such as Cos or PCC as follows:

$$s_{ui}^{\text{Cos}} = \frac{\sum_{k=1}^{f} p_{uk} q_{jk}}{\sqrt{\sum_{k=1}^{f} p_{uk}^2}\sqrt{\sum_{k=1}^{f} q_{jk}^2}} \quad (8)$$

$$s_{ui}^{\text{PCC}} = \frac{\sum_{k=1}^{f}(p_{uk} - \overline{\boldsymbol{p}}_u)(q_{jk} - \overline{\boldsymbol{q}}_j)}{\sqrt{\sum_{k=1}^{f}(p_{uk} - \overline{\boldsymbol{p}}_u)^2}\sqrt{\sum_{k=1}^{f}(q_{jk} - \overline{\boldsymbol{q}}_j)^2}} \quad (9)$$

In the following section, we demonstrate the effectiveness of the newly proposed representations of a user by integrating user-time similarity s_{ui} into popular neighborhood-based systems, namely kNN-Baseline model and its variants.

3.2. Integrating the User-Item Correlations into the Baseline Estimate

In kNNBaseline, the baseline estimate b_{ui} takes the main role of predicting the coarse ratings while the analogy between items serves as a fine-tuning term to improve the accuracy of the final predictions. Subsequently, the more precise the baseline estimate is to the target rating, the better the kNNBaseline model gets in terms of prediction accuracy as demonstrated in several previous studies [8, 9]. However, we realize that a conventional baseline estimate only considers the biases of users and items separately and ignores the user-item correlations, which might lead to an incomplete evaluation.

For example, an RS needs to estimate the ratings of user James for two movies "Titanic" and "Mad Max". Assume the average rating denoted by μ is 3.7 stars. "Titanic" is considerably more well-received by the general audience than an ordinary movie, so it tends to be rated 0.5 stars above the average. Meanwhile, James is a critical user, who usually rates 0.3 stars lower than a moderate user. Thus, the baseline estimate of "Titanic" rated by James would be 3.9 stars $(= 3.7 - 0.3 + 0.5)$. On the other hand, "Mad Max" is also highly popular and tends to be rated 0.6 stars higher than the mean rating; hence, the baseline estimate of James for "Mad Max" would be 4.0 stars $(= 3.7 - 0.3 + 0.6)$, which is similar to the baseline rating for the "Titanic" movie.

However, from James's past interactions with other movies on the system, the recommendation system estimates James's interests using one of the methods described in Section 3.1 and discovers that a *romantic* and *drama* movie like "Titanic" seems to be more suitable for James, while his personal preference is contradictory compared to an *action* and *thriller* movie like "Mad Max". Consequently, the above predicted ratings of James turn out to be rather irrational.

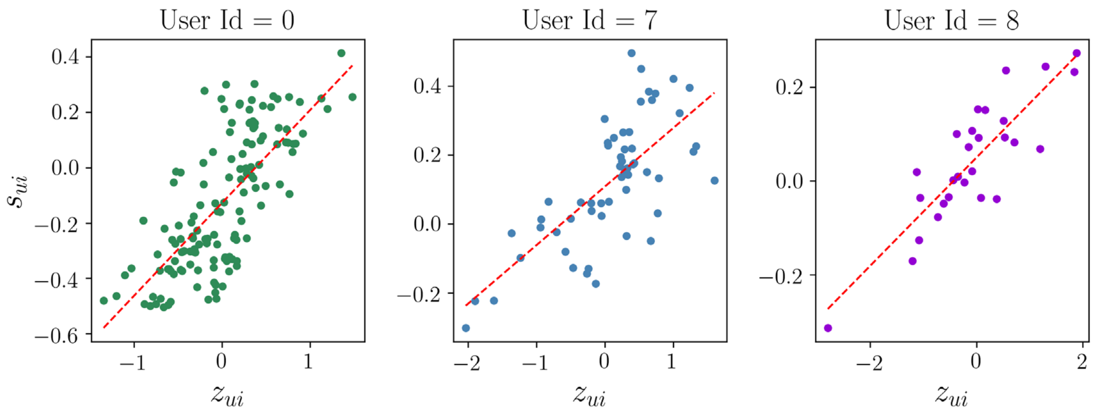

Fig. 1: The residual rating of several users with respect to the user-item similarity degree.

Fig. 3 illustrates the correlations between the residual ratings and the user-item similarity scores of some users, where s_{ui} values are calculated using PCC similarity measure to compare the original Tag Genome data of the movies in the MovieLens 20M dataset and the user feature vectors $\boldsymbol{p}_u^{\text{w-biased}}$. The red trendlines determined by linear regression show that there is an approximately linear relationship between the user-item correlations and the residual ratings: the more interested a user is in a movie (i.e., the higher user-item similarity score), the higher he tends to rate that movie.

In order to take the analogy between a user and an item into account, we propose a revised version of the baseline estimate by integrating the user-item similarity score as follows:

$$b_{ui} = \mu + b_u + b_i + \omega \times s_{ui} \tag{10}$$

where s_{ui} is the similarity degree between user u and item i calculated by a similarity measure such as Cos in (8) or PCC in (9), and ω serves as the weight to adjust the contribution of the user-item correlation term to fit the rating information.

By introducing ω, the least squares problem of the enhanced baseline estimate term is now updated to the following function:

$$b_u^*, b_i^*, \omega^* = \underset{b_u, b_i, \omega}{\arg\min} \sum_{u,i\in\mathbb{K}} \left(r_{ui} - (\mu + b_u + b_i + \omega s_{ui})\right)^2 + \lambda\left(\sum_u b_u^2 + \sum_i b_i^2 + \sum_{u,i\in\mathbb{K}} \omega^2\right) \tag{11}$$

In this paper, two common optimization techniques, namely SGD and ALS, are experimented to solve this problem. An SGD optimizer minimizes the sum of the squared errors in (11) using the following update rule:

$$\begin{aligned} b_u &\leftarrow b_u + \alpha(e_{ui} - \lambda. b_u) \\ b_i &\leftarrow b_i + \alpha(e_{ui} - \lambda. b_i) \\ \omega &\leftarrow \omega + \alpha(e_{ui}. s_{ui} - \lambda. \omega) \end{aligned} \tag{12}$$

where $e_{ui} = r_{ui} - \hat{r}_{ui}$ denotes the predicting error, α denotes the learning rate, and λ denotes L2 regularization term.

Different from SGD, the ALS technique decouples the calculation of one parameter from the others [8]. Each iteration of ALS can be described as follows. First, for each item i, the optimizer fixes the b_u's and ω to solve for the b_i's:

$$b_i = \frac{\sum_{u|(u,i)\in\mathbb{K}} r_{ui} - \mu - b_u - \omega s_{ui}}{\lambda_i + |\{u|(u,i) \in \mathbb{K}\}|} \tag{13}$$

Then, for each user u, the optimizer fixes the b_i's and ω to solve for the b_u's:

$$b_u = \frac{\sum_{i|(u,i)\in\mathbb{K}} r_{ui} - \mu - b_i - \omega s_{ui}}{\lambda_u + |\{i|(u,i) \in \mathbb{K}\}|} \tag{14}$$

Finally, the optimizer fixes both the b_u's and the b_i's to solve for ω:

$$\omega = \frac{\sum_{u,i\in\mathbb{K}} s_{ui}(r_{ui} - \mu - b_u - b_i)}{\lambda_\omega + |\mathbb{K}|} \tag{15}$$

Here, the regularization terms λ_i, λ_u, and λ_ω are the shrinkage and vary due to the number of ratings that affect each parameter. Therefore, each parameter of b_u's, b_i's, and ω needs a distinct value of λ. By applying a *learnable* weighting factor ω to the user-item similarity term, the new kNNBaseline model using the same predicting method in (2) is capable of exploiting auxiliary information to achieve more precise predictions.

4. Performance Evaluation

4.1. MovieLens Dataset and Evaluation Criteria

In this work, one of the most popular datasets for RS research named MovieLens 20M [10] is used as a benchmark. The dataset contains 20,000,263 ratings given by 138,493 users to 27,278 movies. Additionally, there are 465,564 tag applications across 27,278 movies. Especially, Tag Genome data, which is computed using a machine learning system on user-contributed content including tags, ratings, and textual reviews, encodes how strongly movies exhibit particular properties represented by tags [10].

To utilize this kind of data, a cleaning process is applied to the dataset. Specifically, the movies without genome tags are excluded, then, only movies and users with at least 20 ratings are kept. Table 1 summarizes the result of the cleaning stage, where the sparsity of the preprocessed dataset is reduced from 99.47% to 98.97%. The final preprocessed dataset is split into 2 distinct parts: 80% as the training set and the remaining 20% as the testing set.

Table 1. Summary of the original MovieLens 20M dataset and the preprocessed one.

Dataset	**#Ratings**	**#Users**	**#Movies**
Original	20,000,263	138,493	27,278
Preprocessed	19,793,342	138,185	10,239

To evaluate the performance of the proposed models compared to related works, this paper uses three commonly used indicators in the rating prediction task, including RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error) for accuracy evaluation where smaller values indicate better performance and Time [s] for timing evaluation.

In more detail, RMSE and MAE are calculated using the following equations:

$$\text{RMSE} = \sqrt{\sum_{u,i \in \text{TestSet}} \frac{(\hat{r}_{ui} - r_{ui})^2}{|TestSet|}} \tag{16}$$

$$\text{MAE} = \sum_{u,i \in \text{TestSet}} \frac{|\hat{r}_{ui} - r_{ui}|}{|TestSet|} \tag{17}$$

where $|\text{TestSet}|$ is the size of the testing set. The total duration of the learning process on the training set and predicting all samples in the testing set of the model is measured as Time [s].

All experiments are carried out using Google Colaboratory with high RAM configuration (25GB RAM) and no GPU.

4.2. Baseline Models

In this paper, several popular CF models are selected as baselines to evaluate the proposed methods. Firstly, we implement two competitive neighborhood-based models including kNNBaseline [8] and kNNContent [7]. SVD [3] and SVD++ [3] are implemented as the two representatives of the matrix factorization technique. Besides, I-RBM [6] and I-Autorec [5], two deep-learning CF models, are also implemented for comparison.

The optimal hyperparameters for each baseline model are carefully tuned. In particular, the error rates of neighborhood-based models are calculated with the neighborhood size $k \in \{10, 15, 20, 25, 30, 35, 40\}$. Cos$^{\text{content}}$, PCC and PCC$^{\text{content}}$ are chosen as the similarity measures in kNNBaseline and kNNContent models. SVD and SVD++ models are trained using 40 factors with 100 iterations and a step size of 0.002.

To assess the new methods of characterizing users and the proposed baseline estimate in Section 3, the Tag Genome data in the MovieLens 20M dataset is used to construct the feature vector for each movie i as $\boldsymbol{q}_i = \{g_{i1}, g_{i2}, \dots, g_{ik}, \dots\}$ where g_{ik} is the genome score of genome tag k^{th}. In our experiments, p_u^{norm}, p_u^{biased}, and $p_u^{\text{w-biased}}$ are first integrated into the original baseline estimate to find the optimal technique for profiling user interest. The enhanced baseline estimate is then implemented into several neighborhood-based models to comprehensively evaluate its impact on the final prediction.

4.3. Accuracy of the Baseline Estimate Utilizing the User-Item Correlations

In this section, the enhanced baseline estimates are learned using both SGD and ALS optimization algorithms for a more thorough comparison. For SGD, the baseline estimates are trained using the typical learning rate value $\alpha = 0.005$ and the typical regularization value $\lambda = 0.02$. For ALS, the typical values for regularization terms λ_u and λ_i in the MovieLens dataset are 15 and 10, respectively [11]. However, the number of training points in set $\mathbb{K}$ is much larger than the number of appearances of each user or item, which completely differs from the value of λ_ω in (13) from λ_u and λ_i in (11) and (12). Therefore, a grid search is performed on λ_ω, which finds that $\lambda_\omega = -9{,}500{,}000$ provides the system with good performance.

Table 2shows that utilizing the user-item correlations helps to improve the predicting accuracy of the original baseline estimate at the price of increased complexity. Empirical results also prove the superior of $p_u^{\text{w-biased}}$ over its counterparts for both similarity measures being used. Specifically, calculating the user-item similarity with PCC achieves the coarse rating prediction with 6.46% lower RMSE and 6.71% lower MAE but takes approximately 3.6 times as much time as the original baseline estimate (optimized via ALS).

A noteworthy point here is that ALS achieves consistently lower error rates than SGD for all cases at the expense of requiring an additional hyperparameter tuning process (and thus a further computational complexity). However, this trade-off is acceptable at this stage because the absolute time to determine the baseline estimate compared to the total time to make the final prediction is negligible. Hence, ALS is selected as the optimizer for the proposed baseline estimate hereafter.

Table 2. Performance of the enhanced baseline estimates with different types of user feature vectors.

User feature vectors	**Similarity measure**	**SGD**			**ALS**		
		RMSE	**MAE**	**Time [s]**	**RMSE**	**MAE**	**Time [s]**
None		0.8593	0.6595	24	0.8576	0.6590	34
p_u^{norm}	Cos	0.8553 *(-0.47%)*	0.6567 *(-0.42%)*	71 *(x3.0)*	0.8351 *(-2.62%)*	0.6348 *(-3.67%)*	114 *(x3.4)*
	PCC	0.8432 *(-1.87%)*	0.6474 *(-1.83%)*	75 *(x3.1)*	0.8184 *(-4.80%)*	0.6274 *(-4.79%)*	121 *(x3.6)*
p_u^{biased}	Cos	0.8153 *(-5.12%)*	0.6239 *(-5.40%)*	73 *(x3.3)*	0.8129 *(-5.21%)*	0.6228 *(-5.49%)*	117 *(x3.4)*
	PCC	0.8096 *(-5.78%)*	0.6201 *(-5.97%)*	79 *(x3.3)*	0.8072 *(-5.88%)*	0.6186 *(-6.13%)*	126 *(x3.7)*
$p_u^{\text{w-biased}}$	Cos	0.8149 *(-5.17%)*	0.6235 *(-5.46%)*	74 *(x3.1)*	0.8057 *(-6.05%)*	0.6172 *(-6.34%)*	119 *(x3.5)*
	PCC	0.8069 *(-6.10%)*	0.6171 *(-6.43%)*	80 *(x3.3)*	**0.8022** *(-6.46%)*	**0.6148** *(-6.71%)*	**122** *(x3.6)*

4.4. Performance of the Unified Neighborhood-Based System

Finally, the advanced baseline estimates are integrated into kNNBaseline and kNNContent models to refine the process of predicting unknown ratings. For calculating the item-item similarity in kNNBaseline model, two common measures in implementing neighborhood-based RS, namely Cos and PCC, are examined. The same goes for kNNContent model, where Cos$^{\text{content}}$ and PCC$^{\text{content}}$ are both implemented for comparison.

As shown in Fig. 4, the outperformance of the modified baseline estimates over their original shown in Table 2 makes significant accuracy improvements in the rating prediction task: the newly proposed neighborhood-based models, totally surpass their initial versions for all cases by a large margin. Moreover, the empirical results again confirm that incorporating the user and item biases, along with eliminating the effect of low score features, can produce a much more reliable version of user feature vectors.

Another noticeable observation from Fig. 4 is that even though incorporating s_{ui} calculated by PCC is clearly better than Cos when using p_u^{norm}, the difference between these two similarity measures gets much smaller in the case of p_u^{biased} and insignificant with $p_u^{\text{w-biased}}$. This is because the user feature vector generated by (6) or (7) has the original ratings subtracted by the baseline estimate, which makes the mean of the resulting vector come close to 0. Therefore, applying Cos or PCC to the approximately zero-mean vectors produces nearly identical results.

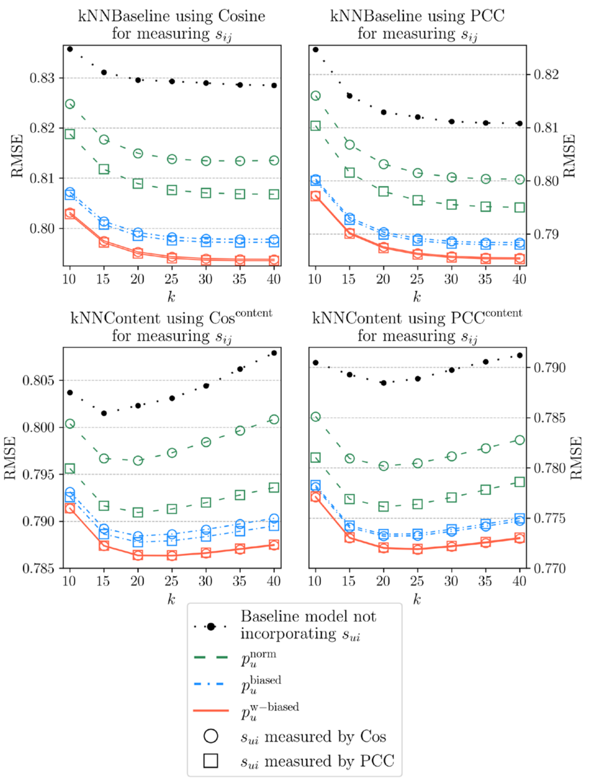

Fig. 4: Error rates of kNNBaseline and kNNContent models when incorporating the user-item similarity scores.

Table 3. Performance of the neighborhood-based models utilizing the user-item correlations against popular CF models.

Model	**RMSE**	**MAE**	**Time [s]**
kNNBaseline ($k = 40$)	0.8108	0.6167	565
kNNContent ($k = 20$)	0.7885	0.5988	293
SVD (40 factors)	0.7922	0.6042	292
SVD++ (40 factors)	0.7894	0.5992	27,387
I-RBM	0.7951	0.6065	96,455
I-AutoRec	0.7808	0.5931	69,860
kNNBaseline incorporating s_{ui} ($k = 40$)	0.7853	0.5981	659
kNNContent incorporating s_{ui} ($k = 25$)	**0.7719**	**0.5866**	**392**
kNN with hybrid similarity matrix incorporating s_{ui} ($k = 25$)	**0.7690**	**0.5830**	**420**

Due to the discussed reasons, in the following experiments, $p_u^{\text{w-biased}}$ and PCC are thus adopted to calculate s_{ui} for best accuracy. Table 3 shows a comparison between the neighborhood-based models incorporating the user-item correlations and several common CF ones, including deep learning systems. Note that for a fair comparison, the number of neighbors k of each neighborhood-based model is chosen for the best performance. In this study, the hybrid similarity matrix proposed in [12] is also integrated into kNN with s_{ui} to boost the accuracy of the unified system.

Specifically, the kNNContent model with s_{ui}, gains:

- 4.80% lower RMSE and 4.88% lower MAE than the original kNNBaseline.
- 2.11% lower RMSE and 2.03% lower MAE than the original kNNContent.
- 2.56% lower RMSE and 2.91% lower MAE than SVD.
- 2.22% lower RMSE and 2.10% lower MAE than SVD++.
- 2.91% lower RMSE and 3.28% lower MAE than I-RBM
- 1.14% lower RMSE and 1.10% lower MAE than I-AutoRec.

On the other hand, by combining the hybrid similarity matrix [12] and using user interests in specific item's features into kNN, the prediction accuracy gains 0.29% lower RMSE and 0.36% lower MAE than the kNNContent with s_{ui}. This improvement again validates the effectiveness of incoporating s_{ui} into neighborhood-based RS.

These improvements in predicting accuracy are achieved at the expense of the additional complexity. However, in practice evaluating the user-item similarity matrix from fixed-length vectors could be performed in parallel with a low computational cost. Hence, we consider that this trade-off is worth in real-life applications.

5. Conclusion

Conventional neighborhood-based CF methods have shown remarkable success in real-life systems. However, due to the lack of user information as a result of several personal privacy concerns on the internet, user-item correlations are not well-considered in implementing RS despite their usefulness in describing users' interests. In this paper, we first introduced various techniques to characterize user preferences utilizing both rating data and item content information. The new user representations not only help the system understand user interests in each item attribute but also make it possible to measure reliable user-item correlations. Consequently, an innovative method was proposed to adjust the baseline estimate of kNN-based that takes user-item similarity scores into account. Thereby, the resulting hybrid models achieve at least 2.11% lower RMSE and 2.03% lower MAE compared to their neighborhood-based counterparts, whilst performing at least 1.10% better compared to other deep learning and matrix factorization systems. This leads to the conclusion that neighborhood-based RSs could be greatly improved by integrating both the item-item and user-item correlations in the predicting model. In addition, we have combined this model with a previously proposed publication, the prediction results have improved but not significantly compared to the idea presented in this paper.

These promising results with user-item similarity have opened us up to several potential research directions. Our future work will focus on extending the application of user-item correlations in other CF systems, not limited to neighborhood-based ones. We are also interested in applying modern neural networks to learn hidden relationships between a user's ratings and the content of those items that he rated to learn more about him and providing efficient user-item similarity scores.

References

[1] F. Ricci, L. Rokach, B. Shapira, Recommender systems: introduction and challenges, in Recommender Systems Handbook, Springer, 2015, p. 1–34. https://doi.org/10.1007/978-1-4899-7637-6_1

[2] Sarwar, B., Karypis, G., Konstan, J., Riedl, J. Item-based collaborative filtering recommendation

algorithms. In Proceedings of the 10th international conference on World Wide Web, pp. 285-295, April 2001. https://doi.org/10.1145/371920.372071.

[3] Y. Koren, Factorization meets the neighborhood: A multifaceted collaborative filtering model, in Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, https://doi.org/10.1145/1401890.1401944.

[4] Y. Koren, R. Bell and C. Volinsky, Matrix factorization techniques for recommender systems, Computer, vol. 42, p. 30–37, 2009, https://doi.org/10.1109/MC.2009.263.

[5] Sedhain, S., Menon, A.K., Sanner, S., Xie, L., Autorec: Autoencoders meet collaborative filtering, in Proceedings of the 24th International Conference on World Wide Web, 2015, https://doi.org/10.1145/2740908.2742726

[6] Salakhutdinov, R., Mnih, A., Hinton, G., Restricted Boltzmann machines for collaborative filtering, in Proceedings of the 24th International Conference on Machine Learning, ACM, 2007, https://doi.org/10.1145/1273496.1273596

[7] T. N. Duong, V. D. Than, T. A. Vuong, T. H. Tran, Q. H. Dang, D. M. Nguyen, H. M. Pham, A novel hybrid recommendation system integrating content-based and rating information, in International Conference on Network-Based Information Systems, 2019, https://doi.org/10.1007/978-3-030-29029-0_30.

[8] Y. Koren, Factor in the neighbors: Scalable and accurate collaborative filtering, ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 4, p. 1, 2010, https://doi.org/10.1145/1644873.1644874.

[9] N. Duong Tan, T. A. Vuong, D. M. Nguyen, Q. H. Dang, Utilizing an autoencoder-generated item representation in hybrid recommendation system, IEEE Access, vol. PP, pp. 1-1, April 2020, https://doi.org/10.1109/ACCESS.2020.2989408.

[10] F. M. Harper, J. A. Konstan, The movielens datasets: History and context, ACM Transactions on Interactive Intelligent Systems (TIIS), vol. 5, p. 19, 2016. https://doi.org/10.1145/2827872

[11] N. Hug, Surprise: A python library for recommender systems, Journal of Open Source Software, vol. 5, p. 2174, 2020, https://doi.org/10.21105/joss.02174.

[12] T. N. Duong, T. G. Do, N. N. Doan, T. N. Cao, T. D. Mai, Hybrid similarity matrix in neighborhood-based recommendation system, in 2021 8th NAFOSTED Conference on Information and Computer Science (NICS), pp. 475-480 2021, https://doi.org/10.1109/NICS54270.2021.9701524.