INCORPORATING USER-ITEM SIMILARITY IN HYBRID NEIGHBORHOOD-BASED RECOMMENDATION SYSTEM

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern hybrid recommendation systems require a sufficient amount of data. However, several internet privacy issues make users skeptical about sharing their personal information with online service providers. This work introduces various novel methods utilizing the baseline estimate to learn user interests from their 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 can boost the accuracy of hybrid neighborhood-based systems by at least 2.11% while minimizing the need for tracking users' digital footprints.

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 such as movies, music, news, and articles, creating billion-dollar industries. Following this evolution, recommendation systems (RSs) have been widely developed to automatically help users 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 (Ricci et al., 2015).

RSs are commonly classified into three main types (Adomavicius & Tuzhilin, 2005): the content-based technique, the collaborative filtering technique, and the hybrid technique. The *content-based* approach, as in Lang (1995); Pazzani & Billsus (1997); Lops et al. (2011); Narducci et al. (2016), learns to recommend items that are similar to the ones that a user liked based on items' features. The main weakness of this approach is the lack of available and reliable metadata associated with item (Mooney & Roy, 2000). Meanwhile, the *collaborative filtering* (CF) approach does not require product information but 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 suggest items that are most similar to the item that a user is interested in (Herlocker et al., 2000; Tintarev & Masthoff, 2007), and ii) matrix factorization models which could explore the latent factors connecting items to users in order to make accurate recommendations (Ricci et al., 2015; Koren et al., 2009; Koren, 2008). 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 so-called *cold-start* problem, *hybrid methods* are proposed by combining auxiliary information into CF models (Singh & Gordon, 2008; Agarwal & Chen, 2009; Wang & Blei, 2011; Li et al., 2011; Rendle, 2010).

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 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 degree 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 on neighborhood-based CF systems, including hybrid models. Detail descriptions of our proposed methods are presented in Section 4. Section 5 gives experimental results and in-depth analysis. At last, we conclude this study in Section 6.

2 PRELIMINARIES

In this paper, u, v denote users and i, j denote items. r_{ui} denotes the preference by user u for item i, also known as the rating, where high values indicate strong preference, and all the (u, i) pairs are stored in the set $\mathbb{K} = \{(u, i)|r_{ui} \text{ is known}\}$. Meanwhile, $\mathbb{R}(u)$ denotes the set of all items rated by user u. In 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 currently two methods for implementing neighborhood-based CF models: i) user-oriented (or user-user) model which predicts a user's preference based on similar users, and ii) item-oriented (or item-item) model which finds similar items to the item a user liked and recommends these items to her. Of the two methods, the latter introduced by Sarwar et al. (2001) has become dominant. This is due to the fact that the number of users in real-life systems is often orders of magnitude bigger than of items, which makes the user-oriented model inefficient. Furthermore, the item-oriented model is capable of providing a rational explanation for recommendations (Ricci et al., 2015). Therefore, our implementations in this work adopt the item-item approach as the base model.

The fundamental of neighborhood-based models is similarity measure. 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. Then, $\hat{r}_{ui}$ can be predicted as a weighted average of the ratings of similar items:

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

Even though Equation (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. kNNBaseline model proposed by Koren (2010) adjusts the above formula through a baseline estimate which accounts for the user and item effects as follows.

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

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).

2.2 INTEGRATING CONTENT-BASED INFORMATION INTO NEIGHBORHOOD-BASED MODELS

A number of problems regarding kNN models using similarity measure on the rating information were noticed by Duong et al. (2019). The first problem is the sparsity of the rating matrix, which

might yield an inaccurate similarity score between two items that share only a few common users. Secondly, 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, a novel similarity measure was proposed using item content-based information. Assuming that each item i is characterized by a feature vector $\boldsymbol{q}_i = \{q_{i1}, q_{i2}, ..., q_{if}\} \in \mathbb{R}^f$ 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.

$$s_{ij}^{\text{Cos}^{\text{content}}} = \frac{\sum_{k=1}^{f} q_{ik} q_{jk}}{\sqrt{\sum_{k=1}^{f} q_{ik}^2} \sqrt{\sum_{k=1}^{f} q_{jk}^2}} \tag{3}$$

or

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

where $\bar{\boldsymbol{q}}_i$ and $\bar{\boldsymbol{q}}_j$ are the mean of feature vectors $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$, respectively. Experiments showed that the item-oriented CF models using $\text{Cos}^{\text{content}}$ and $\text{PCC}^{\text{content}}$ provide equivalent accuracy to the state-of-the-art CF models using rating information whilst performing at least 2 times faster. Hereafter, kNNBaseline model using one of these similarity measures is referred to as kNNContent.

3 Experimental Setting

3.1 MovieLens Dataset and Evaluation Criteria

In this work, the MovieLens 20M dataset is used as a benchmark. This is a widely used dataset in the study of RSs which originally contains 20,000,263 ratings and 465,564 tag applications across 27,278 movies. The ratings are float values ranging from 0.5 to 5.0 with a step of 0.5. Tag Genome data, which is computed on user-contributed content including tags, ratings, and textual reviews, is firstly introduced in this version of MovieLens dataset (Harper & Konstan, 2016). To utilize this kind of data, a cleaning process is applied to the dataset. Specifically, the movies without genome tags are excluded from the dataset. Then, only movies and users with at least 20 ratings are kept. Table 1 summarizes the result of the cleaning stage.

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

Dataset	# Ratings	# Users	# Movies	Sparsity
Original	20,000,263	138,493	27,278	99.47%
Preprocessed	19,793,342	138,185	10,239	98.97%

The preprocessed dataset is split into 2 distinct parts: 80% as the training set and the remaining 20% as the testing set. To evaluate the performance of the proposed models, three commonly used indicators in the field of rating prediction are used: 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. Here, RMSE and MAE are defined as follows.

$$\text{RMSE} = \sqrt{\sum_{u,i \in \text{TestSet}} \left(\hat{r}_{ui} - r_{ui}\right)^2 / |\text{TestSet}|} \tag{5}$$

$$\text{MAE} = \sum_{u,i \in \text{TestSet}} \left|\hat{r}_{ui} - r_{ui}\right| / |\text{TestSet}| \tag{6}$$

where $|\text{TestSet}|$ is the size of the testing set. The total duration of the model's learning process on the training set and predicting all samples in the testing set is measured as Time [s]. All experiments are carried out using Google Colaboratory with $25GB$ RAM and no GPU.

3.2 BASELINE MODELS

In this paper, several popular models are selected as baselines to evaluate the proposed methods. Firstly, we implement two competitive neighborhood-based models including kNNBaseline (Koren, 2010) and kNNContent (Duong et al., 2019). Besides, SVD (Funk, 2006) and SVD++ (Koren, 2008), two well-known representatives of matrix factorization technique, are also experimented due to their superior accuracy and flexible scalability with extremely sparse data (Funk, 2006).

Error rates and the time to make predictions are measured for comparison. The optimal hyper-parameters for each baseline model are carefully chosen using 5-fold cross-validation. In particular, the error rates of the neighborhood-based models are calculated with the neighborhood size $k \in \{10, 15, 20, 25, 30, 35, 40\}$. Due to better performance compared to Cos and $\text{Cos}^{\text{content}}$, PCC and $\text{PCC}^{\text{content}}$ are chosen as the similarity measures in kNNBaseline and kNNContent models. SVD and SVD++ models are trained using 40 hidden factors with 100 iterations and the step size of 0.002.

4 PROPOSED SYSTEMS

So far, kNNBaseline models have successfully applied the item-item similarity exploiting rating information and the available metadata representing item features provided by users. 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 (Ricci et al., 2015). Besides, to our best knowledge, the interest of a user in individual characteristics of an item also lacks careful consideration, which is a major problem restricting the growth of RSs. 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 her personal preferences as much as possible for accurate recommendations. In reality, it is impractical due to a variety of data privacy concerns (Jeckmans et al., 2013).

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 and the feature vectors of those items. We then propose a modification to the baseline estimate of kNNBaseline model and its variants by integrating the user-item similarity score, which boosts the precision of the conventional kNNBaseline model.

4.1 ESTIMATING USER INTERESTS FOR USER-ITEM SIMILARITY MEASURE

In RSs, there are two main sources of information: the interaction records (such as transactions history, ratings, ...) and the item content information (item catalog, movie genres, or the Tag Genome data in the MovieLens 20M dataset). User personal data, however, is not stored or included in publicly available datasets due to the risk of exposing user identities. Therefore, in most datasets for research, there is rarely any data or statistic that directly specifies user interest in each item feature. In this section, we present 3 different methods to characterize a user's interest based on the ratings and metadata of the movies she watched.

The most straightforward approach to estimate a user interest is via a weighted average of the feature vectors $\boldsymbol{q}_i$ of items that she rated as follows.

$$\boldsymbol{p}_u^{\text{norm}} = \frac{\sum_{i \in \mathbb{R}(u)} r_{ui}^{\text{norm}} \cdot \boldsymbol{q}_i}{|\mathbb{R}(u)|} \tag{7}$$

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 $\boldsymbol{p}_u^{\text{norm}}$ of user u has the same dimension and range of element values as an item vector $\boldsymbol{q}_i$. More importantly, 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 helps to create a simple shortcut to understand user preferences, all users are treated in the same way. Specifically, users' ratings are all normalized using the minimum and maximum values of the system's rating scale. Whereas, in practice, different users have a variety

tendencies of rating an item according to their characters. For example, easy-going people often rate movies a little higher than they really feel, 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 will implicitly assume they have the same weight of opinion. This fact leads to the researches taking into account the user and item biases, which have a considerable impact on kNNBaseline and biased SVD models (Koren et al., 2009; Koren, 2010). Therefore, a modification of Equation (7) incorporating the effect of biases is proposed as follows.

$$\boldsymbol{p}_u^{\text{biased}} = \frac{\sum\limits_{i \in \mathbb{R}(u)} z_{ui} \cdot \boldsymbol{q}_i}{\sum\limits_{i \in \mathbb{R}(u)} |z_{ui}|} \tag{8}$$

where the residual rating $z_{ui} = r_{ui} - b_{ui}$ denotes how much extra rating a user u gives more or less than her expectation to a movie i. In more detail, this formula applies the residual ratings 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 $\boldsymbol{p}_u^{\text{biased}}$ now has its elements in the value range of $[-1, +1]$, where -1 / $+1$ indicates that she totally 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 interest. For example, consider user Janet and two movies "Titanic" and "Mad Max". The scores of "Titanic" and "Mad Max" for the *romantic* genre are 0.90 and 0.05, respectively, which means "Titanic" is a romantic movie while "Mad Max" has almost no romantic scene. Assume that Janet's normalized ratings for these movies are $\tilde{r}_{\text{Janet,Titanic}} = 0.7$ and $\tilde{r}_{\text{Janet,Mad Max}} = 0.72$, which are almost identical. Thus, the *romantic* genre score of Janet calculated by Equation (7) is quite low: $(0.7 \times 0.9 + 0.72 \times 0.05)/2 = 0.333$. The fact that "Mad Max" has no romantic element does not mean that Janet doesn't like *romantic* movies. Equation (8) also encounters the same problem. This might lead to misunderstanding the character of a user in some cases.

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. Specifically for the above example, the score of Janet for the *romantic* genre is equal to $\frac{(0.7 \times 0.9 \times 0.9) + (0.72 \times 0.05 \times 0.05)}{(0.7 \times 0.9) + (0.72 \times 0.05)} = 0.854$, which is much more reasonable than measuring the affection of a user for a kind of genre based on items that are not relevant to that genre. The biased feature vector of user u weighted by item feature vector $\boldsymbol{p}_u^{w-biased}$ can be formulated as follows.

$$\boldsymbol{p}_u^{w-biased} = \frac{\sum\limits_{i \in \mathbb{R}(u)} z_{ui} \cdot \boldsymbol{q}_i^2}{\sum\limits_{i \in \mathbb{R}(u)} |z_{ui}| \cdot \boldsymbol{q}_i} \tag{9}$$

From the user feature vectors calculated using one of the above methods, it is noticeable that each value describing user interests has comparatively the same meaning as the corresponding value in the item feature vector. Therefore, the strength of the relevance between a user and an item can be evaluated using common similarity measures such as Cos or PCC (Section 2.2), which eventually calculates a user-item similarity matrix. In the next section, we demonstrate the effectiveness of these vector representations by integrating user-item similarity into the popular kNNBaseline model and its variants.

4.2 Integrating the user-item correlations into the baseline estimate

In item-oriented kNNBaseline, the baseline estimate 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 predicted ratings. Furthermore, it also means that the more precise the baseline estimate is to the targeted rating, the better the kNNBaseline models get in terms of prediction accuracy (Duong Tan et al., 2020). However, we realize that a conventional baseline estimate only considers

the biases of users and items separately, ignoring the user-item correlations, which might lead to a rudimentary evaluation approach.

For example, an RS needs to estimate the ratings of user James to two movies "Titanic" and "Mad Max". Assuming that the average rating, μ, is 3.7 stars. Furthermore, "Titanic" is better 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" 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 $(= 3.7 - 0.3 + 0.6)$. However, from James's past interactions with other movies, the system estimates James's interests using one of the methods described in Section 4.1 and discovers that a *romantic* and *drama* movie like "Titanic" seems to be very suitable for James while his personality is contradictory compared to an *action* and *thriller* movie like "Mad Max". Consequently, the above predicted ratings of James now turn out to be rather irrational.

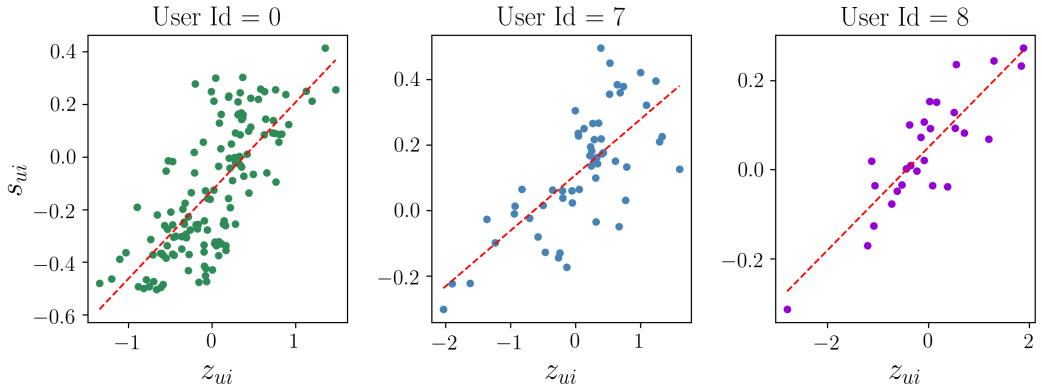

Figure 1: The residual rating of several users with respect to the user-item similarity degree. s_{ij} values are calculated using PCC similarity measure to compare between Tag Genome data of the movies in the MovieLens 20M dataset and the $\boldsymbol{p}_u^{\text{w-biased}}$ matrix. The red trendlines are determined using linear regression.

As illustrated in Figure 1, there is an approximate-linear relationship between the user-item correlations and the residual ratings: the more interested a user is in a movie (i.e., the larger user-item similarity score), the higher she tends to rate that movie. In order to take the analogy between user and 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, 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.

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

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

Different from SGD, the ALS technique decouples the calculation of one parameter from the others (Koren, 2010). In one iteration, the ALS process 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 the ratings that affect each parameter. Therefore, each parameter of b_u's, b_i's, and ω needs a distinct value of λ, which can be determined by cross-validation. By applying a *learnable* weighting factor ω to the user-item similarity term, the new kNNBaseline model is capable of exploiting auxiliary information to achieve more precise predictions.

5 Performance Evaluation

To assess the new methods of characterizing user preferences and the proposed baseline estimate in Section 4, Tag Genome in the MovieLens 20M dataset is used to construct a movie feature vector: $\boldsymbol{q}_i = \{g_{i1}, g_{i2}, ..., g_{ik}, ...\}$ where g_{ik} is the genome score of genome tag k^{th}. In our experiments, $\boldsymbol{p}_u^{\text{norm}}$, $\boldsymbol{p}_u^{\text{biased}}$, and $\boldsymbol{p}_u^{\text{w-biased}}$ are first integrated into the traditional baseline estimate to find the optimal technique of profiling user interest in terms of predicting accuracy. The enhanced baseline estimate is then implemented into several neighborhood-based models to comprehensively evaluate its impact on the final rating prediction.

5.1 Accuracy of the baseline estimate utilizing the user-item correlation

The enhanced baseline estimates are learned using both optimization algorithms SGD and ALS for comparison. For SGD, the baseline are trained using the learning rate $\alpha = 0.005$ and the regularization $\lambda = 0.02$. For ALS, typical values for λ_u and λ_i in the MovieLens dataset are 15 and 10, respectively (Hug, 2020). 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 the value of λ_ω from λ_u and λ_i. Therefore, a grid search is performed on λ_ω, which finds that $\lambda_\omega = -9,500,000$ is the best choice.

Table 2 shows that utilizing the user-item correlation helps to improve the accuracy of the traditional baseline estimate at the price of increased complexity. Empirical results also prove the superior of $\boldsymbol{p}_u^{\text{w-biased}}$ 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. The conventional baseline estimate without the user-item similarity is included for comparison.

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*
$\boldsymbol{p}_u^{\text{norm}}$	Cos	0.8553 *(-0.47%)*	0.6567 *(-0.42%)*	71 (x*3.0*)	0.8351 *(-2.62%)*	0.6348 *(-3.67%)*	114 (x*3.4*)
	PCC	0.8432 *(-1.87%)*	0.6474 *(-1.83%)*	75 (x*3.1*)	0.8184 *(-4.80%)*	0.6274 *(-4.79%)*	121 (x*3.6*)
$\boldsymbol{p}_u^{\text{biased}}$	Cos	0.8153 *(-5.12%)*	0.6239 *(-5.40%)*	73 (x*3.3*)	0.8129 *(-5.21%)*	0.6228 *(-5.49%)*	117 (x*3.4*)
	PCC	0.8096 *(-5.78%)*	0.6201 *(-5.97%)*	79 (x*3.3*)	0.8072 *(-5.88%)*	0.6186 *(-6.13%)*	126 (x*3.7*)
$\boldsymbol{p}_u^{\text{w-biased}}$	Cos	0.8149 *(-5.17%)*	0.6235 *(-5.46%)*	74 (x*3.1*)	0.8057 *(-6.05%)*	0.6172 *(-6.34%)*	119 (x*3.5*)
	PCC	0.8069 *(-6.10%)*	0.6171 *(-6.43%)*	80 (x*3.3*)	**0.8022** *(-6.46%)*	**0.6148** *(-6.71%)*	**122** (x*3.6*)

5.2 Performance of the unified neighborhood-based system

Finally, the advanced baseline estimates are integrated into kNNBaseline and kNNContent models to refine the ultimate rating predictions. For calculating the item-item similarity in kNNBaseline model, two common measures Cos and PCC are examined. The same goes for kNNContent model, where $\text{Cos}^{\text{content}}$ and $\text{PCC}^{\text{content}}$ are both implemented for comparison.

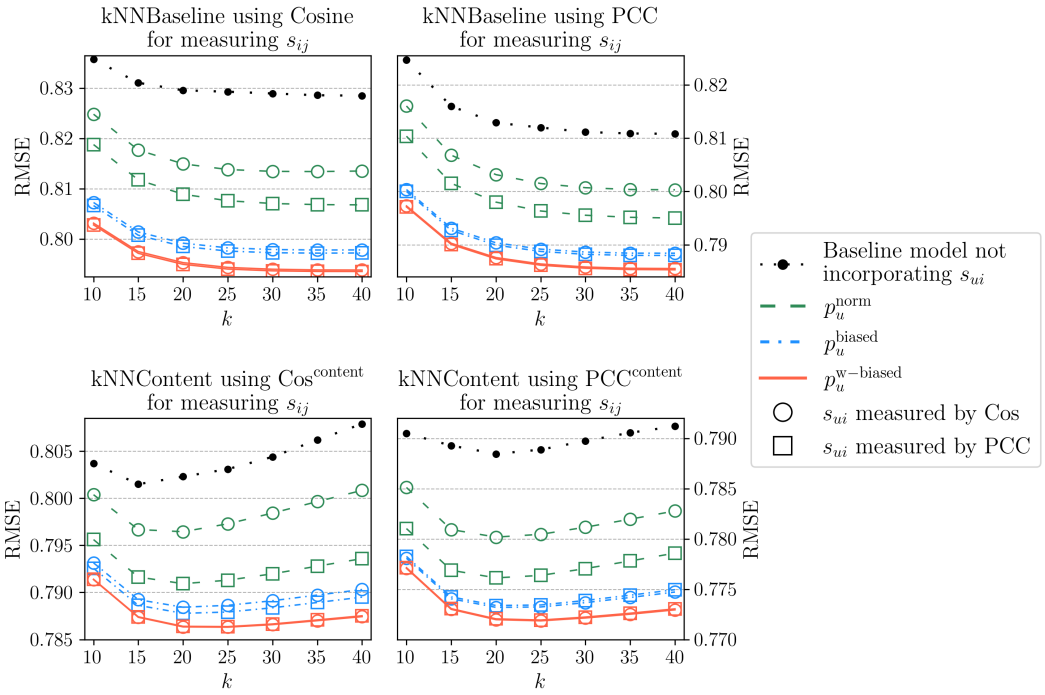

Figure 2: Error rates of kNNBaseline and kNNContent models when incorporating the user-item correlations with different sizes of the neighborhood.

As shown in Figure 2, the outperformance of the modified baseline estimates over their original makes a significant improvement in predicting accuracy: the newly proposed neighborhood-based models totally surpass their initial versions for all cases. It is noticeable that even though incorporating s_{ui} calculated by PCC is clearly better than Cos when using $\boldsymbol{p}_u^{\text{norm}}$, the difference between these two similarity measures gets much smaller in the case of $\boldsymbol{p}_u^{\text{biased}}$ and almost disappears with $\boldsymbol{p}_u^{\text{w-biased}}$. This is because the user feature vector generated by Equation (8) or Equation (9) 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. In the following experiments, $\boldsymbol{p}_u^{\text{w-biased}}$ and PCC are thus opted for calculating the user-item similarity for best accuracy.

Table 3 shows a comparison between the neighborhood-based models incorporating the user-item correlations and several common CF ones. The most accurate model, kNNContent with s_{ui}, gains:

- 4.80% lower RMSE and 4.88% lower MAE than original kNNBaseline.
- 2.11% lower RMSE and 2.03% lower MAE than original kNNContent.
- 2.56% lower RMSE and 2.91% lower MAE than SVD.
- 2.22% lower RMSE and 2.10% lower MAE than SVD++.

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
kNNBaseline incorporating s_{ui} ($k = 40$)	0.7853	0.5981	659
kNNContent incorporating s_{ui} ($k = 25$)	**0.7719**	**0.5866**	**392**

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 it in real-life applications.

6 Conclusion

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 to understand user interests in each item attribute but also make it possible to measure the user-item correlations. An innovative method was then proposed to adjust the baseline estimate of kNNBaseline model that takes the user-item similarity into account. Thereby, the resulting hybrid models achieve at least 2.11% lower RMSE and 2.03% MAE compared to their neighborhood-based counterparts. 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.

References

Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

Deepak Agarwal and Bee-Chung Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 19–28. ACM, 2009.

Tan Nghia Duong, Viet Duc Than, Tuan Anh Vuong, Trong Hiep Tran, Quang Hieu Dang, Duc Minh Nguyen, and Hung Manh Pham. A novel hybrid recommendation system integrating content-based and rating information. In *International Conference on Network-Based Information Systems*, pp. 325–337. Springer, 2019.

Nghia Duong Tan, Tuan Anh Vuong, Duc Minh Nguyen, and Quang Hieu Dang. Utilizing an autoencoder-generated item representation in hybrid recommendation system. *IEEE Access*, PP: 1–1, 04 2020. doi: 10.1109/ACCESS.2020.2989408.

Simon Funk. Netflix update: Try this at home, 2006.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):19, 2016.

Jonathan L Herlocker, Joseph A Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pp. 241–250. ACM, 2000.

Nicolas Hug. Surprise: A python library for recommender systems. *Journal of Open Source Software*, 5(52):2174, 2020. doi: 10.21105/joss.02174. URL `https://doi.org/10.21105/joss.02174`.

Arjan JP Jeckmans, Michael Beye, Zekeriya Erkin, Pieter Hartel, Reginald L Lagendijk, and Qiang Tang. Privacy in recommender systems. In *Social media retrieval*, pp. 263–281. Springer, 2013.

Yehuda 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*, pp. 426–434. ACM, 2008.

Yehuda Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(1):1, 2010.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

Ken Lang. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*, pp. 331–339. Elsevier, 1995.

Wu-Jun Li, Dit-Yan Yeung, and Zhihua Zhang. Generalized latent factor models for social network analysis. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pp. 73–105. Springer, 2011.

Raymond J Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pp. 195–204. ACM, 2000.

Fedelucio Narducci, Pierpaolo Basile, Cataldo Musto, Pasquale Lops, Annalina Caputo, Marco de Gemmis, Leo Iaquinta, and Giovanni Semeraro. Concept-based item representations for a cross-lingual content-based recommendation process. *Information Sciences*, 374:15–31, 2016.

Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27(3):313–331, 1997.

Steffen Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pp. 995–1000. IEEE, 2010.

Francesco Ricci, Lior Rokach, and Bracha Shapira. Recommender systems: introduction and challenges. In *Recommender systems handbook*, pp. 1–34. Springer, 2015.

Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. Item-based collaborative filtering recommendation algorithms. *Www*, 1:285–295, 2001.

Ajit P Singh and Geoffrey J Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 650–658. ACM, 2008.

Nava Tintarev and Judith Masthoff. A survey of explanations in recommender systems. In *2007 IEEE 23rd international conference on data engineering workshop*, pp. 801–810. IEEE, 2007.

Chong Wang and David M Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 448–456. ACM, 2011.