Abstract  Traditional Collaborative Filtering (CF) based methods are applied to understand the personal preferences of users/customers for items/products from rating matrix. Usually, rating matrix is sparse in nature. So there are some improved variants of CF method that apply the increasing amount of side information to handle the sparsity problem. Only linear kernel or only non-linear kernel is applied in most of the available recommendation related work to understand user-item latent feature embeddings from data. Only linear kernel or only non-linear kernel is not sufficient to learn complex user-item features from users’ side information. Recently, some researchers have focused on hybrid models that learn some features with non-linear kernel and some other features with linear kernel. But it is very difficult to understand which features can be learned accurately with linear kernel or with non-linear kernel. To overcome this problem, we propose a novel deep fusion model named FusionDeepMF and the novel attempts of this model are i) learning user-item rating matrix and side information through linear and non-linear kernel simultaneously, ii) application of a tuning-parameter determining the trade-off between the dual embeddings that are generated from linear and non-linear kernels. Extensive experiments on online review datasets establish that FusionDeepMF can be remarkably futuristic compared to other baseline approaches. Empirical evidence also shows that FusionDeepMF achieves better performances compared to linear kernel of Matrix Factorization (MF) and non-linear kernel of Multi-layer Perceptron (MLP).

1 Information Profiling and Retrieval, ZBW-Leibniz Information Centre for Economics, Düsternbrooker Weg 120, 24105 Kiel, Germany.
2 Department of Computer Science & Engineering, Indian Institute of Technology Patna, Bihta, Bihar, India 801106.
E-mail: 1s.mandal@zbw.eu, 2abyaym@iitp.ac.in.
ORCID iD: 100000002-1310-7291.
* corresponding author.
**Keywords** Recommendation System, Deep Neural Network, Matrix Factorization, Reliability score, Review Network.

1 **Introduction**

In our daily life, recommender systems [40] play a vital role to suggest right products, right social media, preferable news or travel places to the target users in different sectors. Using the knowledge of demographic information, past view history and previous purchasing history, recommender systems understand the users’ preferences and suggest suitable products or preferable news or close friends to target users [40] [42]. An efficient recommendation model can not only understand users’ preferable zones but also increases both contentment for users and earnings for companies. The Content-Based model [26, 35] and Collaborative Filtering (CF) model [16, 41] are the two important categories of recommender systems. The Memory-Based approach [47] and Model-Based approach [22] are two types of CF. Memory-based models generally investigate the customer’s preferable items from rating values given by users on items. Each entry of a rating matrix is formed based on users’ rating values for their purchased items. But a model-based approach investigates the way users give ratings on items. It claims more adequate efficiency than the memory-based approach.

Initially, researchers focused only on users’ previous rating behavior activities to understand their preferences. However, only previous rating activities are insufficient for rating prediction of an unknown product. In recent studies, researchers are focusing on explicit feedback and implicit feedback as users’ side information [49]. Implicit feedback specifies users’ activities before purchasing any products. Users’ view activities, user-product interaction, analyzing product image, reading reviews are considered as implicit feedback. Implicit feedback indicates users’ preferences. Explicit feedback (ratings, reviews) indicates users’ activity after purchasing any product [50]. It indicates users’ satisfaction level. In [18, 59, 46] the authors have focused only on implicit feedback. For explicit feedback, most of the researches consider users’ ratings and reviews [3, 25, 27, 59]. In [2, 6, 18, 60], the authors have focused on analyzing review text which leads to a better rating prediction. Some researchers have focused on user-product interaction value based on rating activities [7, 9, 14, 51]. There are few works, where the authors have focused on both explicit rating values and implicit feedback [4, 21, 23, 24]. In [5, 7, 51, 56, 57], hybrid models are developed. These models consider users’ explicit and implicit feedback simultaneously.

It is shown that in review forums of e-commerce sites, there are some positive reviewers (who always give positive or good ratings), some negative reviewers (who always give negative or poor ratings), some reliable reviewers (who give ratings according to the quality of products) and some reviewers are whimsical (whose posted ratings are unpredictable). User buy products based on their preferable zones but their rating behaviour activities should
relate to their characteristics as potential reviewers. We have to understand whether a user is a positive or a negative or a reliable reviewer. In most of the papers, when the existing approaches learn latent features of a user based on similar behaviours among other users’ rating activities or preference degree, they overlook the characteristics of users.

To understand users' characteristic, in this work we consider users' reliability score as side information. This reliability score indicates the loyalty and influential power of users’ reviews for their purchased products. A user’s reliability score varies over different purchased products. So, we evaluate this score of a user for a particular product. Some research works [35, 38] have used the term “reliability”. But this score just indicates users’ trust score. The evaluation of our proposed score is demonstrated in Section 3 and the evaluation process is completely different from the existing works. In [1, 37], the researchers propose the concept of reliability based on support for users (the total ratings posted by the user) and support for products (how many ratings are posted for this product). This concept is totally different from our evaluation technique. In [43], the authors incorporate the ratings, reviews, and feedback into a probabilistic matrix factorization framework for prediction. They design users’ reliability measure that combines their consistency between ratings and sentiment of reviews and the helpfulness feedback on their reviews. We have used this method [43] as a baseline model. Our reliability concept is different from this baseline work and the concept of learning methodology is also different (discussed in Section 4).

Based on our experimental data we form a user-product bipartite network, where all existing side information are coming from the users directly. The reliability score is computed from the bipartite network between users and products. Consequently we classify reliability score as a network feedback score [31, 32]. It is neither an explicit feedback nor an implicit feedback. Traditional recommendation systems mainly focus on reviewers role, their side information as explicit and implicit feedback. Traditional recommendation systems mainly focus on reviewers role, their side information as explicit and implicit feedback to predict rating. In our work, the raters also play a vital role to notify the reliability of a reviewer’s review and help the model to learn the latent factor of the reviewer more accurately to predict rating. Most of the previous works ignore raters’ role in recommendation systems.

**Model Architecture:** Although users’ side information and selection of features have a crucial role in efficient recommendation, model architecture is another vital part in recommendation. The Matrix Factorization (MF) [17, 14, 52, 53] approach is a popular model-based approach because of its accurate recommendation and high scalability [11, 11]. This model factorizes both products and users by a low-dimensional feature vector space, where every latent vector is treated as a user’s feature vector. A lot of works [15, 29, 34, 46, 55] are continuing for accurate recommendation using probabilistic MF model. The researchers are focusing on users’ implicit [14, 39] or explicit feedback [18, 46] to increase learning quality of user-product features embeddings. For some users, linearity of a kernel can model features vectors based on users’ side in-
formation to predict ratings more accurately. But for some other users, the linear kernel-based methods may fail to learn the features of users and products [55].

To overcome the above problem, recently most of the researchers are employing deep neural networks (DNNs) [18, 54, 58], which use nonlinear kernel to model latent features based on users’ side information. In Deep belief Networks features are learned automatically [2, 14] (effective feature representation is learned from the dataset). Deep learning model achieves its popularity by its scalability for dimension reduction technique. In [8], the authors use Multi-view DNNs to match rich user features to product features. Another reason for its wide popularity is that DNNs are scalable to large datasets. In [7, 9, 14, 51], the authors have developed deep learning model to learn user-product interaction value based on rating activities. Due to its fast features learning technique and latent vector dimension reduction technique, deep learning model is also preferred in our research. However, for some users’ non-linear model learn users’ latent features very accurately, but for some users non-linear kernel fails [51].

It is very evident that both linear and nonlinear models can not learn all users’ latent features independently or separately. Recent studies are focusing on hybrid models that learn some features with non-linear kernel and some features with linear kernel. In ConvMF [19], the authors apply CNN on users’ posted reviews, products’ descriptions and learn users’ rating activities using probabilistic matrix factorization. In VMCF [39], the authors apply CNN to extract features from pictures of products and apply MF on users’ rating activities and “also-viewed” relationships. But it is very difficult to understand which features are learned accurately with linear kernel and which features are learned accurately with non-linear kernel. From our experiments, we have observed that both kernels are effective with different set of users.

To overcome this problem, we propose a fusion based model FusionDeepMF, where a MF based model applies a linear kernel to model the latent features based on users’ rating activity and reliability score and on the other hand a MLP based model uses non-linear kernel to learn users’ latent feature vectors. In concatenation layer of our model, we combine two separate embeddings, where one embedding is the output embedding of the MF model and another one is the output embedding of the MLP model. The embedding of the MF model indicates user’s rating vector which is learned through the MF model based on the users’ rating activities and reliability scores. Another embedding presents user’s rating vector which is learned through the MLP model based on rating activities and reliability scores. In this layer two separate embeddings can mutually reinforcement each other to learn better rating values.

In this paper, our major contributions are:

- We consider reliability score as a network feedback which indicates users’ characteristics as reviewers.
- Introduce FusionDeepMF model and two novel characteristics of this model are i) learning user-item rating matrix and side information through lin-
FusionDeepMF: Deep Fusion based Recommendation

ear and non-linear kernel simultaneously, ii) usage of a tuning-parameter determining the trade-off between the dual embeddings that are generated from linear and non-linear kernels of our model.

- We extensively evaluate the initialized input parameters of FusionDeepMF model through pre-training phase and demonstrate the effectiveness of our proposed model by comparing it with the state-of-the-art models.

The rest of the paper is categorized as follows: In section 2 we explain our problem and intuition. Then we introduce our side information as network feedback in section 3. How we learn users' feature vector embeddings using our model FusionDeepMF, is discussed in section 4. The result of comprehensive experiments on online review datasets is discussed in section 5. Finally, conclusion and future works are presented in section 6.

2 Problem Formulation

2.1 Notation

Suppose, $U = \{u_1, u_2, ..., u_n\}$ be the users set, $P = \{p_1, p_2, ..., p_m\}$ be the products set, $R = \{r_1, r_2, ..., r_N\}$ be the ratings set, $H = \{rel_1, rel_2, ..., rel_N\}$ be set of the reliability scores, where $n$ and $m$ represent the count of users and products, respectively. $N$ represents the count of both ratings and reliability scores. First, we build a user-product rating matrix $R \in \mathbb{R}^{n \times m}$ from dataset, where each entry $r_{ij}$ is the rating value given by user $u_i$ on product $p_j$. Similarly, we build a user-product reliability score matrix $H \in \mathbb{R}^{n \times m}$ from dataset, where each entry $rel_{ij}$ is the reliability score gained by user $u_i$ on her review on $p_j$. The range of reliability score $rel_{ij}$ is in $(0,1)$ and the raw rating $r_{ij}$ is also mapped into $(0,1]$ to learn parameters efficiently. The sigmoid function is chosen to map the inner product of latent vectors into $[0,1]$ range.

2.2 Problem Definition

Input: 1) The user-product rating matrix $R \in \mathbb{R}^{n \times m}$ from our dataset where each entry $r_{ij}$ is the rating value given by user $u_i$ on product $p_j$, 2) The user-product reliability score matrix $H \in \mathbb{R}^{n \times m}$ from our dataset where each entry $rel_{ij}$ is the reliability score of the rating of user $u_i$ on product $p_j$.

Output: The primary goal is to characterize the users’ latent feature vectors based on reliability and rating score in a more accurate way, and to give a better interpretation of how the similarity of users’ reliability scores affect rating prediction of the unknown products using linear kernel and non-linear kernel.
3 Network Feedback: users’ Side Information

Networks are extensively used in different fields of research as a convenient presentation of patterns of interaction between appropriate staffs. In research, social network, biological network, ecological network etc. are widely used. Based on our experimental data, we form a network named Review Network.

**Review Network:** Before purchasing any product from e-commerce sites, we check the previous users’ opinions regarding that product. A network of users is formed depending on reviewing activities of users on different products. We have named this network as Review Network and it is discussed in our previous work [28, 29, 30].

3.1 Rating Score

In e-commerce sites, a user posts ratings on purchased product. Here $r_{ij}$ is denoted as rating of user $u_i$ on $p_j$. In this paper, the rating range is 1 to 5 similar to the Amazon.com dataset. We consider rating 1 and 2 as negating ratings and 3 to 5 as positive ratings. Rating $r_{ij}$ is normalized into $(0,1]$ by using function $f(x) = x/5$ (5 is the maximum rating).

3.2 Reliability Score

We propose the reliability score of a user for a specific product. The reliability value of a user might vary for the different products the user purchased. This score of a user for each review is calculated depending on the number of helpful “yes” votes on the review, how many other users read the review as most current reviews and top ranking reviews. Here we present the overview of the reliability score in review network.

Before buying any product, users can check the previous feedback or reviews regarding that particular product. From these reviews, user can judge the quality of the product through the satisfaction levels of the previous users. In most of the e-commerce sites, after each review, sites post a query, “Was this review helpful to you? (Answer Yes/No)”. We define helpfulness as a measure for the validation of a review. We normalize the helpfulness $h_{ij}$ of user $u_i$ for product $p_j$ (i.e., $u_i$ wrote review $f_{ij}$ on $p_j$) based on how many other users are helpful from the review and the normalized helpfulness score $h_{ij}$ is represented as follows:

$$h_{ij} = \frac{l_{ij}}{n'}$$

where,

$$l_{ij} = \frac{\text{#helpful votes on review } f_{ij}}{\text{total #votes on review } f_{ij}}^{2}.$$ 

Here, the number of users who buy $p_j$, is denoted by $n'$. We assign more weight to the particular users who gain more helpful votes. Thus, Eq. [2] is
FusionDeepMF: Deep Fusion based Recommendation

As a special case, if a dataset contains only the helpful vote information without the number of total votes, then the denominator of Eq. 2 will be replaced by the maximum helpful votes gained by any review on product \( p_j \).

Usually, when a user wants to read the previous reviews for a particular product, the user has two options to choose, i.e., i) read the top-ranking reviews and ii) read the most recent reviews. Helpful votes of the reviews influence us to read. That is why any user prefers the top-ranking feedback (top-ranking feedbacks are selected based on helpful votes). Sometimes users prefer to read both the most recent reviews of a particular product to understand the current quality of the product. Some users prefer to read the top-ranking reviews and most recent reviews to come to a purchasing decision.

Based on the potential reading activities of the most current reviews we evaluate most recent score \( \text{most}_{ij} \) of user \( u_i \) for product \( p_j \) and the equation is as follows:

\[
c_{ij} = \sum_{s=1}^{n' - i} \frac{1}{s^2},
\]

and the normalized \( c_{ij} \) (i value range is 1 to \( n' \)) is denoted as \( \text{most}_{ij} \),

\[
\text{most}_{ij} = \frac{c_{ij}}{\sum_{x=1}^{n'} c_{xj}},
\]

where, \( \text{most}_{ij} \in (0, 1) \) and the denominator is a summation defined by \( x \) and \( x=1 \) to \( n' \). Here, \( n' \) denotes the count of users who posted feedback for \( p_j \) and \( n' - i \) is the count of users who buy \( p_j \) after \( u_i \). It is shown that in Fig. ??, \( u_1 \) is the first user who posts feedback for product \( p_1 \). When \( u_2 \) buys the exact same, she may check \( u_1 \)'s feedback as most current review and for \( u_2 \), \( u_1 \)'s score = \((1/1^2)\). When \( u_3 \) user buys this product, she may check \( u_1 \)'s feedback as second most current feedback and for \( u_3 \) her score = \((1/2^2)\) and for \( u_4 \), \( u_1 \)'s score = \((1/3^2)\). The score of \( u_1 \) user for product \( p_1 = \sum_{s=1}^{4} \frac{1}{s^2} = (1/1^2) + (1/2^2) + (1/3^2) + (1/4^2)\). We assign more weight to the most current reviewers regarding the same product. So the above Eq. 3 has quadratic nature.

The top ranking feedback is evaluated depending on helpfulness score of users’ reviews. Based on the reading activities of the top ranking reviews we evaluate top ranking reviewer score \( \text{top}_{ij} \) of user \( u_i \) for product \( p_j \) and the equation is as follows:

\[
q_{ij} = \left( \frac{1}{o_{ij}} \right) \times (n' - i),
\]

and the normalized \( q_{ij} \) (i value range is 1 to \( n' \)) is denotes as \( \text{top}_{ij} \),

\[
\text{top}_{ij} = \frac{q_{ij}}{\sum_{x=1}^{n'} q_{xj}},
\]

where, \( \text{top}_{ij} \in (0, 1) \) is the score of user \( u_i \) for product \( p_j \) according to the top ranking reviews. Here \( o_{ij} \) denotes the ranking for user \( u_i \) for her purchased
product \( p_j \). The top rank depends on high helpfulness score. Here, \( n' - i \) denotes the count of users who buy the product after \( i^{th} \) user and read that review. We assign more weight to the top ranking reviewers regarding the same product. So the above Eq. 5 is quadratic in nature.

Based on the reading activities of the top ranking reviews and most current reviews, we evaluate total score \( d_{ij} \) as follows:

\[
d_{ij} = \alpha \ast \text{top}_{ij} + (1 - \alpha) \ast \text{most}_{ij},
\]

where \( \alpha \in [0, 1] \) denotes the weightage. Here 0.5 is considered as the weightage, because we want to give the same priority to both \( \text{top}_{ij} \) and \( \text{most}_{ij} \). The value of \( d_{ij} \in (0, 1) \).

We define **reliability score** \( \text{rel}_{ij} \) of review \( f_{ij} \) as the average of the two scores:

\[
\text{rel}_{ij} = \frac{h_{ij} + d_{ij}}{2},
\]

where the value of \( \text{rel}_{ij} \in (0, 1) \). If reliability value \( \geq 0.5 \) (industry shall decide the threshold; here the threshold value is assigned to 0.5), that means the user is a reliable user. If reliability value \( < 0.5 \), the user is not a reliable one and rating activity indicates that the user is either a positive or a negative reviewer.

### 4 FusionDeepMF: A Methodology using Reliability Score

In this section, we explore the effectiveness of users’ reliability score to learn latent feature vectors of users and products for rating prediction. Users’ latent feature vectors based on the reliability score and rating value indicate their reliability as reviewer, validation of their reviews and rating behavior activities. We have already discussed that independently only the linear kernel of a linear model or only the non-linear kernel of a non-linear model is not sufficient to learn complex user-product features. Our experiments establish the fact. We propose a model named **FusionDeepMF**, where MF uses a linear kernel to model the latent features based on users’ reliability score and rating values, and on the other hand MLP model uses a non-linear kernel to learn users’ reliability score and rating values for their purchased products. In the concatenation layer of our model as represented in Fig. 1 we integrate two separate embeddings i.e., \( \theta^{\text{MF}} \) and \( \theta^{\text{MLP}} \), where \( \theta^{\text{MF}} \) is the final embedding of the MF model and \( \theta^{\text{MLP}} \) is the final embedding of the MLP model. The \( \theta^{\text{MF}} \) indicates user’s latent features vector that is learned through the MF model based on the user’s rating activities and reliability scores for the user’s purchased products. The \( \theta^{\text{MLP}} \) presents user’s latent features vector that is learned through the MLP model based on the user’s rating activities, reliability scores for a particular product. In this model two separate embeddings can mutually reinforce each other to learn a better rating value. The \( \theta^{\text{MF}} \) and \( \theta^{\text{MLP}} \), these two embeddings are crucial in our model. How we generate these two embeddings through MF and MLP, is discussed in the next subsections.
4.1 Formation of $\theta^{MF}$ Embedding through $MF$ Model

This model uses linear kernel to learn latent features based on users’ rating scores and reliability scores. Using this model we investigate the effectiveness of rating prediction based on reliability score. The input of the $MF$ model is the data of $u_i$ for product $p_j$ based on the user’s rating score $r_{ij}$ and reliability score $rel_{ij}$. The reliability score is evaluated from Eq. 1 to 8.

4.1.1 Rating Prediction: Capturing Users’ Rating Score

In this section, we discuss how we learn users’ and products’ latent features vectors based on users’ rating behaviour activity and predict ratings. First, we build a user-product rating matrix $R \in \mathbb{R}^{n \times m}$ from our dataset where each entry $r_{ij}$ is the users’ posted rating value given by $u_i$ on product $p_j$. Singular Value Decomposition ($SVD$) is a popular matrix factorization technique that represents each user and product as a real valued vector of latent features. We apply $SVD$ to the user-product rating matrix $R \in \mathbb{R}^{n \times m}$ and decompose into three matrices $W' \in \mathbb{R}^{K \times n}$, $S_o \in \mathbb{R}^{K \times K}$ and $Z' \in \mathbb{R}^{K \times m}$, where $S_o$ is a diagonal matrix whose elements (singular values) are equal to the root of the positive eigenvalues of $RR^T$ or $R^T R$. Then after observing the performance of our model, we decide to generate $W \in \mathbb{R}^{K \times n}$ and $Z \in \mathbb{R}^{K \times m}$, where $w_i \in W \in \mathbb{R}^{K \times n}$ ($w_i$ is the $i$th column of $W \in \mathbb{R}^{K \times n}$) is the user-specific latent feature vector of user $u_i$ and $z_j \in Z \in \mathbb{R}^{K \times m}$ ($z_j$ is the $j$th column of $Z \in \mathbb{R}^{K \times m}$) is the product-specific latent feature vector of product $p_j$. By applying low rank approximation method to the observed user-product rating matrix, unknown ratings by $W^T Z$ are predicted. Together, the $w_i^T z_j$ indicates how user $u_i$ rates on product $p_j$. Formally, the feature matrices $W$ and $Z$ can be learned by minimizing a loss function as follows:

$$
\varphi = \min_{W, Z} \sum_{i=1}^{n} \sum_{j=1}^{m} I_{ij}^R (r_{ij} - g(w_i^T z_j))^2 + \lambda \left( \sum_{i=1}^{n} n_{w_i} \|w_i\|_F^2 + \sum_{j=1}^{m} n_{z_j} \|z_j\|_F^2 \right),
$$

where, $\|.\|_F$ represents the Frobenius norm and $g(.)$ indicates Sigmoid function. The term $\lambda(.)$ is introduced to avoid over-fitting. Here, $I_{ij}^R = 1$ if $u_i$ rates on product $p_j$, otherwise $I_{ij}^R = 0$. $n_{w_i}$ denotes the total count of ratings, that is given by $u_i$ and $n_{z_j}$ is the total count of users who rate on product $p_j$.

4.1.2 Factorization of Reliability Score Matrix

In this section, we discuss how we learn users’ and products’ latent features vectors based on users’ reliability scores. Here, we build a user-product reliability score based matrix $H \in \mathbb{R}^{n \times m}$ from our dataset where each entry $rel_{ij}$

1 https://medium.com/@jonathan_hui/machine-learning-singular-value-decomposition-svd-principal-component-analysis-pca-1d45e885e491
is the reliability score gained by $u_i$ from the user’s review on $p_j$. We follow the decomposition technique of our rating matrix and similarly we decompose the user-product reliability matrix $H \in \mathbb{R}^{n \times m}$ into two matrices $E \in \mathbb{R}^{K \times n}$ and $F \in \mathbb{R}^{K \times m}$, where $e_i \in E^{K \times n}$ ($e_i$ is the $i^{th}$ column of $E^{K \times n}$) is the user-specific latent feature vector of $u_i$ based on reliability score and $f_j \in F^{K \times m}$ ($f_j$ is the $j^{th}$ column of $F^{K \times m}$) is the product-specific latent feature vector of $p_j$. By applying low rank approximation method to the observed reliability matrix, we can recover unknown reliability scores by $E^T F$. Together, the $e_i^T f_j$ indicates how user $u_i$’s rating gains reliability score on $p_j$. Formally, the feature matrices $E$ and $F$ can be learned by minimizing a loss function as follows:

$$\varphi = \min_{E,F} \sum_{i=1}^{n} \sum_{j=1}^{m} I_{ij}^{rel} (rel_{ij} - g(e_i^T f_j))^2 + \lambda \left( \sum_{i=1}^{n} n_{e_i} \| e_i \|_F^2 + \sum_{j=1}^{m} n_{f_j} \| f_j \|_F^2 \right),$$

(10)

where, $I_{ij}^{rel} = 1$ if user $u_i$ gains reliability score on product $p_j$, otherwise $I_{ij}^{rel} = 0$. Here $n_{e_i}$ denotes the count of products for which $u_i$ gains reliability score and $n_{f_j}$ is the total count of users who gain score on $p_j$.

4.1.3 Rating Prediction: Capturing Users’ Reliability Score

In this section, we discuss how we predict rating based on users’ reliability scores. In our model, matrix $E$ is the latent space commonly shared by $R$ and $H$, meaning the user feature matrix based on rating $W$ is approximated by user feature matrix based on reliability score $E$. Here each vector $e_i$ characterizes two aspects at the same time: how user $u_i$’s review gains reliability score for products and how the same user posts ratings on products. Together, the $e_i^T z_j$ indicates how we predict $u_i$’s rating on product $p_j$ based on $u_i$’s reliability score similarity with the other users, where $e_i$ indicates how user $u_i$ gains reliability score for all products and $z_j$ indicates the all users’ rating activity on $p_j$. It is the approximation of original rating score $r_{ij}$. Therefore, we can learn the feature matrices $E, Z$ simultaneously by minimizing the following objective function:

$$\varphi = \min_{E,Z} \sum_{i=1}^{n} \sum_{j=1}^{m} I_{ij}^R (r_{ij} - g(e_i^T z_j))^2 + \sum_{i=1}^{n} \sum_{j=1}^{m} I_{ij}^{rel} (rel_{ij} - g(e_i^T f_j))^2$$

$$+ \lambda \left( \sum_{i=1}^{n} n_{e_i} \| e_i \|_F^2 + \sum_{j=1}^{m} n_{z_j} \| z_j \|_F^2 + \sum_{j=1}^{m} n_{f_j} \| f_j \|_F^2 \right).$$

(11)

In this way, we can integrate two types of data sources and obtain latent spaces $E$ and $Z$, that can work together to give exact prediction.
4.1.4 Optimization Technique for MF model

We adopt a popular method, mini-batch Adaptive Moment Estimation (Adam) \(^{[20]}\) which tunes the learning rate for each parameters by performing smaller updates for frequent and larger updates for infrequent parameters. Mini-batch Adam optimizer is a popular method for deep learning model. Our model is the fusion of MF and MLP. So, we adopt a common optimizer as mini-batch Adam for FusionDeepMF, pre-training phase of MF and MLP. The gradient of \(\varphi\) (Eq. 9) with respect to \(w_i\) is evaluated as described in \(^{[20]}\) to optimize the objective function. Similarly, we evaluate \(\frac{\partial \varphi}{\partial z_j}\) from Eq. 10 and update the parameters. After independently training of Eq. 9 and 11 one can obtain two sets of feature matrices. Let \(w_{i}^{MF}\) and \(z_{j}^{MF}\) be the user and product-specific vector, respectively learned based on users’ rating activities. Let \(e_{i}^{vMF}\) and \(z_{j}^{vMF}\) be the user and product-specific vector, respectively learned based on users’ reliability scores. After convergent we get final updated values. Here, \(w_{i}^{MF}\), \(z_{j}^{MF}\), \(e_{i}^{vMF}\), \(z_{j}^{vMF}\) are crucial to generate \(\theta^{MF}\) embedding for user \(u_i\), that represents the user’s feature based on her rating activity and reliability score.

4.1.5 Learning of \(\theta^{MF}\) Embedding

We generate \(\theta^{MF}\) embedding to represent users’ latent features vectors based on rating behaviour activities and reliability scores. In MF layer as shown in Fig. 1 we perform addition of two output vector (\(K\) dimensional latent features vector each) of \(w_{i}^{MF}\odot z_{j}^{MF}\) and \(e_{i}^{vMF}\odot z_{j}^{vMF}\). After observing the performances of different fusion methods such as concatenation, addition, or element wise product in our model, addition method is preferred for combining two output vector. We evaluate \(\theta^{MF}\) embedding as follows:

\[
\theta^{MF} = \left[ W_{MF1}[w_{i}^{MF}\odot z_{j}^{MF}] + W_{MF2}[e_{i}^{vMF}\odot z_{j}^{vMF}] \right],
\]

where, \(W_{MF1}, W_{MF2}\) denote the weight matrices of the input of MF layer. Here, \(\odot\) indicates element wise product. The \(\theta^{MF}\) indicates user’s feature vector that is learned through MF model based on the user’s rating activities and reliability scores on her purchased products.

4.2 Formation of \(\theta^{MLP}\) embedding through MLP Model

The MF model uses linear kernel to model latent feature embedding \(\theta^{MF}\) based on users’ rating activities and reliability scores. Our observation is that for some users, the MF model uses linear kernel to model latent features very accurately and for some users MF model fails. So in our model FusionDeepMF...
as shown in Fig.1, we have generated $\theta^{MLP}$ embedding based on users’ rating activities and reliability score through a deep learning technique, that uses non-linear kernel to model latent features. Using this model we investigate the effectiveness of rating prediction based on reliability score through linear and non-linear kernel. In this section, we discuss a Multilayer Perceptron Model (MLP), that generates $\theta^{MLP}$ embedding.

4.2.1 The Input and Embedding Layer of MLP model

The input of the MLP model is the same as the initial input of the MF model, where the data of user $u_i$ for product $p_j$ is based on the user’s rating score $r_{ij}$ and reliability score $rel_{ij}$. The rating and reliability score are evaluated from Eq.1 to 8. Suppose, $U = \{ u_1, u_2, ..., u_n \}$ be the users set, $P = \{ p_1, p_2, ..., p_m \}$ be the products set, $R = \{ r_1, r_2, ..., r_N \}$ be the ratings set, $H = \{ rel_1, rel_2, ..., rel_N \}$ be the set of reliability scores, where $n$ and $m$ represent the count of users and products, respectively. $N$ denotes the count of ratings

Fig. 1: FusionDeepMF: A deep fusion model architecture with the size of embedding $K = 256$, where optimal user’s feature embeddings of linear model and non-linear model mutually reinforcement each other for a better prediction.
and reliability score in our dataset. Embedding layer is fully connected. The dimension of \( w_i, z_j, e_i, f_j \) is \((1 \times K)\), and each vector is the feature vector of user \( u_i \) and product \( p_j \) based on rating, reliability score and these vectors are passed to the next layer which is the fusion layer.

### 4.2.2 Fusion Layer

In this layer, the embedded features of users and products based on ratings and reliability score are combined for better presentation of learning. After observing the performances of different fusion methods such as concatenation, addition, or element-wise product in our model, addition fusion method is preferred for combining users’ vectors and products’ vectors. After fusion a vector represents a user’s features based on rating and reliability scores and another vector indicates an product’s features. After the addition step, we use a fully-connected neural layer and ReLU activation function is applied directly for more effective performance and it is also perceived in our experiment. The outputs of the Fusion layer are \( a_i \in \mathbb{R}^K \) and \( b_j \in \mathbb{R}^K \) which are the representation vector of user \( u_i \) and product \( p_j \) (learned from fusion part), respectively.

### 4.2.3 MLP Layer

Our model acquires two pathways to design users and products latent vectors combining rating and reliability score. In the multi-modal deep learning work [45], this design has been popularly followed. For designing a deep neural model, simply vectors concatenation are not enough to extract user-product activity from user and product latent features. To overcome this problem, we add hidden layers on the concatenated vector and this structure can furnish the model with flexibility and non-linearity to understand the user’s latent features vector based on \( a_i \) and \( b_j \). To obtain the feature vectors we have performed concatenation, where \( V = [a_i; b_j] \). It is passed into fully connected layers as follows:

\[
\theta_{\text{MLP}} = o_L(W_L(o_{L-1}(W_{L-1}...o_1(W_1V + b_1) + b_{L-1}) + b_L)),
\]

where \( L \) presents the count of hidden layers. \( W_L, b_L \) and \( o_L \) present the weight matrix, bias vector, activation function for \( L^{th} \) layer’s perceptron, respectively. \( \theta_{\text{MLP}} \) indicates user’s embedding that is learned through MLP model based on the user’s rating activities, reliability scores. We choose ReLU activation function. In our investigation the performance of ReLU is better than tanh and sigmoid.

To learn features of data accurately [12], we develop the tower structure of MLP layers where the small number of hidden units for higher layers are employed as shown in Figure [1]. Our model’s performance with different number of layers has been examined and after that, we apply four hidden layers.
4.3 Fusion between $\theta^{MF}$ and $\theta^{MLP}$ embedding

To provide more flexibility, we allow pre-training of $MF$ and $MLP$ to learn in separate way and combine the optimal output embeddings $\theta^{MF}$ and $\theta^{MLP}$ in the concatenation layer as shown in Fig. 1. The formulation of this concatenation is as follows:

$$\hat{R}_{ij} = \left( W_h \left[ \theta^{MF}; \theta^{MLP} \right] \right),$$  \hspace{1cm} (14)

where $W_h$ denotes edge weight of the concatenation layer. The derivative of the model $w.r.t$ each model parameter can be calculated with standard back-propagation.

4.4 Pre-training

Due to the non-convexity of the objective function of FusionDeepMF model, gradient based optimization methods find only locally optimal solutions. For the convergence and performance of FusionDeepMF model, initialization plays an important role. Since our model is an ensemble of $MF$ and $MLP$, we decide to initialize FusionDeepMF model using the pre-trained models of $MF$ and $MLP$.

Pre-training of $MF$ Model: We have evaluated predicted rating $\hat{r}_{ij}$, which does not follow the well-known practice as $\hat{r}_{ij} = g\left( \frac{(w_{r_{ij}}^{MF})^T z_{j}^{MF} + (e_{v_{ij}}^{MF})^T z_{j}^{MF}}{2} \right) \ast r_{max}$, because in our observation, it is found that for some users two latent factors i.e., $w_{r_{ij}}^{MF}$, $z_{j}^{MF}$ claim more weightage values out of four latent factors. For some users, $e_{v_{ij}}^{MF}$, $z_{j}^{MF}$ two latent factors claim more weightage values. So, in our work, we propose a different strategy where in $MF$ layer we perform addition of the two output vectors of $w_{r_{ij}}^{MF} \odot z_{j}^{MF}$ and $e_{v_{ij}}^{MF} \odot z_{j}^{MF}$. We evaluate the $\theta^{MF}$ embedding as mentioned in Eq. 12.

This edge-weightage matrix is learned by standard back-propagation technique with mini-batch Adam optimizer and MAE as the loss function with original rating value $r_{ij}$. $\theta^{MF}$ indicates user’s feature vector that is learned through $MF$ model based on the user’s rating activities, reliability scores. The final predicted user’s latent vector $\hat{R}_{ij}$ is evaluated as follows:

$$\hat{R}_{ij} = \left( W_{h}^{MF} \right)^T [\theta^{MF}],$$  \hspace{1cm} (15)

where, $W_{h}^{MF}$ is the edge weights of the output layer. The predicted rating ($\hat{r}_{ij}$) is obtained via a regression layer:

$$\hat{r}_{ij} = W_m \hat{R}_{ij} + b_m,$$  \hspace{1cm} (16)

where, $W_m$ and $b_m$ are the weight matrix and bias vector, respectively. Each output of our training dataset of our model is trained with the original target
output \((r_{ij})\) with the standard back-propagation technique. For our model we use the mini-batch Adam optimizer and MAE as the loss function after experimental observation.

**Pre-training of MLP Model:** The pre-training of MLP model is same as discussed in section 4.2. After evaluating \(\theta^{MLP}\) (output of the MLP layer) using the same Eq. 13, we generate final predicted user’s latent vector \(\hat{R}_{ij}\) as follows:

\[
\hat{R}_{ij} = \left( (W_{h}^{MLP})^T [\theta^{MLP}] \right),
\]

(17)

where, \(W_{h}^{MLP}\) is the edge weights of the output layer. The predicted rating \((\hat{r}_{ij})\) is obtained via a regression layer:

\[
\hat{r}_{ij} = W_{mlp}\hat{R}_{ij} + b_{mlp},
\]

(18)

where, \(W_{mlp}\) and \(b_{mlp}\) are the weight matrix and bias vector, respectively. Each output of our training dataset of our model is trained with the original target output \((r_{ij})\) with the standard back-propagation technique.

We first train MF and MLP until convergence. We then use the models parameters as the initialization for the parameters of FusionDeepMF. For Eq. 14, the edge weight matrix \(W_h\) is denoted as follows:

\[
W_h \leftarrow [\gamma W_{h}^{MF} : (1-\gamma)W_{h}^{MLP}],
\]

(19)

where, \(W_{h}^{MF}\) and \(W_{h}^{MLP}\) denote the weight vector of the pre-trained models MF and MLP. The tuning-parameter \(\gamma\) learns the trade-off between \(\theta^{MF}\) and \(\theta^{MLP}\) two embeddings.

### 4.5 Regression and Rating Prediction of FusionDeepMF Model

The predicted rating \((\hat{r}_{ij})\) of FusionDeepMF Model is obtained via a regression layer:

\[
\hat{r}_{ij} = W_{re}\hat{R}_{ij} + b_{re},
\]

(20)

where, \(\hat{R}_{ij}\) is the output of Concatenation Layer (Eq. 14) of FusionDeepMF Model. \(W_{re}\) and \(b_{re}\) are the weight matrix and bias vector, respectively. Each output of our training dataset of our model is trained with the original target output \((r_{ij})\) with the standard back-propagation technique. For our model we use the mini-batch Adam optimizer and MAE as the loss function after the experimental observation.
4.6 Complexity Analysis

The computational time for the objective function (Eq. 11) is $O(tK(|\Omega| + |\Psi|))$, where $t$ represents the iteration counts, $K$ presents the dimension of feature vectors, $\Omega$ and $\Psi$ present the count of available rating values and reliability score, respectively. The computing time of gradients against $W$, $Z$, $E$, $F$ are $O(tK|\Omega|)$, $O(tK|\Omega|)$, $O(tK|\Psi|)$, $O(tK|\Psi|)$, respectively. Therefore, total time complexity of the MF model is $O(tK(|\Omega| + |\Psi|))$.

The computing time of the MLP model is evaluated based on the computing time of the objective function of this model for rating score, reliability score and its gradients against $w_i$, $z_j$, $e_i$, $f_j$. We also consider the computing time of the hidden layers and the number of iterations. Based on the learning of all parameters, the total computing time of the MLP model is $O\left(t(K(|\Omega| + |\Psi|) + \sum_{i=1}^{L} T_{i-1}T_i)\right)$. The additional time $\sum_{i=1}^{L} T_{i-1}T_i$, is caused by the hidden layers. Here $L$ is the number of hidden layers. For the $i^{th}$ hidden layer, the computing time is $O(T_{i-1}T_i)$.

The total computational time of our model FusionDeepMF is $O\left(t(K(|\Omega| + |\Psi|) + \sum_{i=1}^{L} T_{i-1}T_i) + O(tK(|\Omega| + |\Psi|)) \approx O\left(t(K(|\Omega| + |\Psi|) + \sum_{i=1}^{L} T_{i-1}T_i)\right).\right.$

In order to maintain efficiency and accuracy, we pre-train the model to reduce our model time complexity to $O(tK(|\Omega| + |\Psi|))$. With the idea of pre-training, we put extra time into pre-training without affecting our framework and this optimization process scales linearly with size of the given data.

5 Experimental Result

For analyzing performance of FusionDeepMF model, we consider the Amazon.com online review dataset \footnote{http://jmcauley.ucsd.edu/data/amazon/; https://nijianmo.github.io/amazon/} on electronics, movies, TV, video etc. Data statistics are presented in Table 1, where the 1\textsuperscript{st} column indicates different category of data. Here, the 2\textsuperscript{nd}, 3\textsuperscript{rd} and 4\textsuperscript{th} column indicate the count of users (unique), the count of products (unique) and the count of users’ posted reviews, respectively. The Amazon.com data contains: “reviewerID”, “asin”, “reviewerName”, “helpful”, “reviewText”, “overall”, “summary”, “unixReviewTime”, “reviewTime”.

An example of the dataset is as follows:

\{
"reviewerID": "A2SUAM1J3GN3B",
"asin": "0000013714",
"reviewerName": "J. McDonald",
"helpful": [3, 5],
"reviewText": "I bought this .... Great purchase though!",
\}
“overall”: 3.0,
“summary”: “Heavenly Highway Hymns”,
“unixReviewTime”: 126472000,
“reviewTime”: “09 13, 2009”.

In this dataset, “asin” presents product id and “overall” presents rating. Helpfulness score is evaluated from “helpful” entry and “helpful”: [3, 5] indicates three users are helpful from the review and other two users feel it is not helpful. We identify the brand name from “reviewText” field of the review. Users’ view-products and pictures of products are collected from [13, 33]. It has “asin”, “title”, “price”, “imUrl”, “related”, “salesRank”, “brand”, “category” entries. Here “related” entry has information about user id(s) who only view products but not buy and the user id(s) who purchase products after viewing products. The pictures of products are collected from the “imUrl” entry.

| Dataset          | # users | # products | # reviews/ ratings |
|------------------|---------|------------|--------------------|
| Electronics      | 811,034 | 82,067     | 1,241,778          |
| Books            | 2,588,991 | 929,264   | 12,886,488         |
| Music            | 1,134,684 | 556,814   | 6,396,350          |
| Movies & TV      | 1,224,267 | 212,836   | 7,850,972          |
| Home & Kitchen   | 644,509  | 79,006     | 991,794            |
| Amazon Instant Video | 312,930 | 22,204     | 717,651            |

5.1 Baseline methods

In our model, we use reliability score as users’ side information. To investigate the effectiveness of our model with considered side information, we choose some popular baselines, which are developed based on different side information and model architecture as shown in Table 2. To assess the performance of our methodology, the baselines are applied on Amazon.com dataset. For implementation purpose we follow the same experimental settings as mentioned in the baseline models [3, 7, 9, 19, 21, 39, 43, 51]. The baseline models are as follows:

i) **SVD++ [21]**: Both implicit feedback and explicit feedback are integrated in this model. Implicit feedback is decided from available rating values. If a user buys a product and rates it, that means he likes the product and positive implicit feedback is generated. Similarly if the user does not buy a product, that means he does not like the product and negative implicit feedback is derived. The users’ rating activities are treated as explicit feedback.

ii) **ConvMF [19]**: Matrix factorization integrates convolutional neural network (CNN) into probabilistic matrix factorization (PMF) that captures
contextual information of documents to learn users’ preference areas. This model focuses on explicit feedback. A document latent vector from proposed CNN is used as the mean of Gaussian noise of a product. The authors propose a model that plays an important role as a bridge between PMF and CNN that helps to analyze both description documents and ratings.

iii) VMCF [39]: This approach integrates the product pictures and “also-viewed” product information for rating prediction. The authors use CNN to extract visual features. Then the embedded visual features and embedded “also-viewed” products are used in proposed PMF model for better rating prediction.

iv) DMF [51]: In Deep Matrix Factorization Model, the authors construct a matrix with explicit ratings and non-preference implicit response. Meanwhile, the authors mark a zero if the rating is unknown, which is named as non-preference implicit feedback in this paper. If there is any interaction between user $u_i$ and product $p_j$, then it is denoted as explicit rating value where in [14] it is denoted by 1. In this paper, the predicted interaction value of a user for a new product indicates her rating value on this product. The model [51] performs better than the popular model [14] when user-product interaction feature is considered based on rating activities. So, we choose [51] as a baseline model.

v) HCFDS [7]: The authors utilize advances in learning effective representations in deep learning, and propose a hybrid model which jointly performs deep users and products’ latent factors learning from side information and collaborative filtering from the rating matrix. They use users’ rating activities and additional side information of users to learn user-product latent features embeddings.

vi) CM+RIM [9]: In Novel Deep Learning-Based Collaborative Filtering Model, the authors first capture the overall co-occurrence and local co-occurrence given by certain rating using Constraint Model (CM) and Rating Independent Model (RIM), respectively. They propose a multiviews neural networks relying on user-product interaction. Moreover, both CM or RIM are fed into the multiviews neural networks to further enhance the performance of rating prediction.

vii) SBMF+R [43]: The authors develop a sentiment analysis approach using a new star-based dictionary construction technique to obtain the sentiment score. They design a user reliability measure that combines user consistency and the feedback on reviews. They also incorporate the ratings, reviews, and feedback into a probabilistic matrix factorization framework for prediction.

viii) DBNSA [3]: The authors propose a deep learning model to process user comments and to generate a possible user rating for user recommendations. First, the system uses sentiment analysis to create a feature vector as the input nodes. Next, the system implements noise reduction in the data set to improve the classification of user ratings. Finally, a deep belief network and sentiment analysis (DBNSA) achieves data learning for the recommendations
Table 2: Properties of the considered methods are compared here.

| Models          | Explicit Ratings | Side information                                      | Model                      |
|-----------------|-------------------|-------------------------------------------------------|----------------------------|
| SVD++           | ✓                 | users’ preferences generated from rating               | SVD                        |
| VMCF            | ✓                 | users’ review                                         | Hybrid                     |
| SBMF+R          | ✓                 | product-images, also-viewed activity                   | SVD+Deep neural network     |
| DMF             | ✓                 | ratings, reviews, and reliability of feedback          | Probabilistic matrix factorization |
| HCFDS           | ✓                 | degree of preference from users’ ratings               | Hybrid                     |
| DBNSA           | ✓                 | user-product interaction                               | Deep belief network         |
| CM+RIM          | ✓                 | users’ review and product description                  | Fusion of MF and MLP        |

5.2 Evaluation Metrics

The error evaluation metrics *Root Mean Square Error (RMSE)* and *Mean Absolute Error (MAE)* are applied to evaluate the performances of our proposed model. We apply the ranking metrics i.e., *Precision*, *Recall*, *F1-score* and *Normalized Discounted Cumulative Gain (NDCG)* to evaluate product ranking of our models and baselines.

*RMSE* is formulated as

$$RMSE = \sqrt{\frac{\sum_{(i,j)\in\tau} (\hat{r}_{ij} - r_{ij})^2}{|\tau|}},$$ (21)

and *MAE* is formulated as

$$MAE = \frac{\sum_{(i,j)\in\tau} |\hat{r}_{ij} - r_{ij}|}{|\tau|},$$ (22)

where, $|\tau|$ presents our predicted ratings set.

*Precision*, *Recall*, *F1-score*, Mean Average Precision (MAP) and *NDCG* are formulated as follows:

$$Precision = \frac{1}{n} \sum_{i=1}^{n} \frac{|Pre(i) \cap Orig(i)|}{|Pre(i)|},$$ (23)

$$Recall = \frac{1}{n} \sum_{i=1}^{n} \frac{|Pre(i) \cap Orig(i)|}{|Orig(i)|},$$ (24)

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall},$$ (25)

$$MAP = \frac{1}{n} \sum_{i=1}^{n} \frac{\sum_{t=1}^{y} Prec@t(i).I_{i,t}}{|Orig(i)|},$$ (26)

$$NDCG = \frac{1}{n} \sum_{i=1}^{n} NDCG(i) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{Z_i} DCG(i),$$ (27)

*Precision*, *Recall*, *F1-score*, Mean Average Precision (MAP) and *NDCG* are formulated as follows:

$$Precision = \frac{1}{n} \sum_{i=1}^{n} \frac{|Pre(i) \cap Orig(i)|}{|Pre(i)|},$$ (23)

$$Recall = \frac{1}{n} \sum_{i=1}^{n} \frac{|Pre(i) \cap Orig(i)|}{|Orig(i)|},$$ (24)

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall},$$ (25)

$$MAP = \frac{1}{n} \sum_{i=1}^{n} \frac{\sum_{t=1}^{y} Prec@t(i).I_{i,t}}{|Orig(i)|},$$ (26)

$$NDCG = \frac{1}{n} \sum_{i=1}^{n} NDCG(i) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{Z_i} DCG(i),$$ (27)
where,

$$DCG(i) = \sum_{j \in \omega(i)} \frac{2^{r_{ij}} - 1}{\log_2(1 + \text{rank}(j))}. \quad (28)$$

Here, $\omega(i)$ indicates the number of products in the test set those are rated by $u_i$. $\text{Orig}(i) = \{j \in \omega(i) \mid r_{ij} \geq 3\}$ indicates the count of preferable products of $u_i$, that is treated as ground truth $r_{ij}$ in test dataset and $\text{Pre}(i) = \{j \in \omega(i) \mid \hat{r}_{ij} \geq 3\}$ indicates the count of suggested products to $u_i$ based on rating prediction. $\text{rank}(j)$ denotes the ranking place of $p_j$ in the sorted list of $\omega(i)$, based on prediction of users’ posted rating values. Based on the actual rating values in the test set, the normalized factor $Z_i$ is the DCG value of the original ranking place of $\omega(i)$. The range of NDCG is $(0,1)$. The top-$t$ based ranking metric is denoted by $\text{Precision}@t$. $\text{Precision}@t$ is denoted by $\text{Prec}@t$.

The length of ranking list of $u_i$ for her buying products is denoted as $s$. $I_{i,t} = 1$ if the product at the $t^{th}$ place is defined by $u_i$ and 0 otherwise.

Table 3: Comparative RMSE results among MF, MLP and FusionDeepMF on the Amazon.com online reviews dataset.

| Dataset          | Training size | MF    | MLP   | FusionDeepMF |
|------------------|---------------|-------|-------|--------------|
| Electronics      | 40%           | 1.663 | 1.621 | 1.593        |
|                  | 50%           | 1.411 | 1.363 | 1.201        |
|                  | 60%           | 1.311 | 1.263 | 1.022        |
|                  | 70%           | 1.271 | 1.153 | **0.907**    |
| Books            | 40%           | 1.373 | 1.401 | 1.321        |
|                  | 50%           | 1.193 | 1.223 | 1.161        |
|                  | 60%           | 1.101 | 1.141 | 0.968        |
|                  | 70%           | 1.051 | 1.120 | **0.827**    |
| Music            | 40%           | 1.393 | 1.411 | 1.355        |
|                  | 50%           | 1.196 | 1.263 | 1.101        |
|                  | 60%           | 0.991 | 1.057 | 0.852        |
|                  | 70%           | 0.874 | 0.927 | **0.724**    |
| Movies & TV      | 40%           | 1.401 | 1.312 | 1.237        |
|                  | 50%           | 1.291 | 1.146 | 1.093        |
|                  | 60%           | 1.011 | 0.962 | 0.830        |
|                  | 70%           | 0.901 | 0.864 | **0.711**    |
| Home & Kitchen   | 40%           | 1.577 | 1.543 | 1.511        |
|                  | 50%           | 1.391 | 1.311 | 1.253        |
|                  | 60%           | 1.292 | 1.123 | 0.911        |
|                  | 70%           | 1.101 | 0.959 | **0.867**    |
| Amazon Instant Video | 40%         | 1.687 | 1.735 | 1.637        |
|                  | 50%           | 1.493 | 1.512 | 1.461        |
|                  | 60%           | 1.071 | 1.271 | 0.926        |
|                  | 70%           | 0.996 | 1.074 | **0.896**    |
Table 4: Comparison of error evaluation results on all users among the baselines (present in 3rd to 10th column) and our approach (present in 11th column). Best performances are marked in bold.

| Dataset            | Metrics | SVD++ | ConvMF | VMCF   | SBFM+R | DMF   | HCFDS | CM+RIM | DBNSA | FusionDeepMF |
|--------------------|---------|-------|--------|--------|--------|-------|-------|--------|-------|--------------|
| Electronics        | RMSE    | 1.371 | 1.294  | 1.121  | 1.107  | 1.102 | 1.101 | 1.101  | 1.067 | 0.934        |
|                    | MAE     | 0.997 | 0.912  | 0.871  | 0.815  | 0.812 | 0.809 | 0.809  | 0.791 | 0.763        |
| Books              | RMSE    | 1.167 | 1.101  | 0.927  | 0.921  | 0.911 | 0.901 | 0.901  | 0.801 | 0.789        |
|                    | MAE     | 0.915 | 0.811  | 0.793  | 0.793  | 0.783 | 0.783 | 0.783  | 0.753 | 0.753        |
| Music              | RMSE    | 1.153 | 0.988  | 0.910  | 0.821  | 0.811 | 0.809 | 0.809  | 0.796 | 0.769        |
|                    | MAE     | 0.901 | 0.856  | 0.805  | 0.803  | 0.801 | 0.781 | 0.781  | 0.741 | 0.724        |
| Movies & TV        | RMSE    | 1.147 | 0.987  | 0.901  | 0.841  | 0.802 | 0.800 | 0.800  | 0.758 | 0.778        |
|                    | MAE     | 0.946 | 0.911  | 0.961  | 0.773  | 0.770 | 0.769 | 0.769  | 0.741 | 0.701        |
| Home & Kitchen     | RMSE    | 1.281 | 1.211  | 1.174  | 1.165  | 1.101 | 0.983 | 0.986  | 0.941 | 0.867        |
|                    | MAE     | 1.175 | 1.031  | 0.985  | 0.911  | 0.907 | 0.894 | 0.894  | 0.861 | 0.798        |
| Amazon Instant Video| RMSE   | 1.312 | 1.276  | 1.145  | 1.112  | 1.101 | 1.087 | 0.981  | 0.967 | 0.890        |
|                    | MAE     | 1.216 | 1.122  | 1.011  | 0.991  | 0.988 | 0.977 | 0.951  | 0.943 | 0.838        |

Table 5: Performance of ranking metrics results between the baselines (present in 3rd to 10th column) and our model (present in 11th column).

| Dataset            | Metrics | SVD++ | ConvMF | VMCF   | SBFM+R | DMF   | HCFDS | CM+RIM | DBNSA | FusionDeepMF |
|--------------------|---------|-------|--------|--------|--------|-------|-------|--------|-------|--------------|
| Electronics        | F1-score| 0.572 | 0.608  | 0.650  | 0.723  | 0.731 | 0.742 | 0.722  | 0.781 | 0.820        |
|                    | NDCG    | 0.671 | 0.696  | 0.707  | 0.744  | 0.751 | 0.771 | 0.782  | 0.793 | 0.846        |
| Books              | F1-score| 0.600 | 0.624  | 0.664  | 0.714  | 0.780 | 0.782 | 0.794  | 0.801 | 0.884        |
|                    | NDCG    | 0.644 | 0.656  | 0.741  | 0.761  | 0.783 | 0.787 | 0.804  | 0.811 | 0.884        |
| Music              | F1-score| 0.628 | 0.667  | 0.704  | 0.746  | 0.733 | 0.741 | 0.789  | 0.804 | 0.837        |
|                    | NDCG    | 0.658 | 0.704  | 0.728  | 0.751  | 0.773 | 0.780 | 0.803  | 0.823 | 0.883        |
| Movies & TV        | F1-score| 0.612 | 0.668  | 0.729  | 0.744  | 0.763 | 0.783 | 0.791  | 0.805 | 0.884        |
|                    | NDCG    | 0.638 | 0.717  | 0.744  | 0.763  | 0.780 | 0.791 | 0.811  | 0.845 | 0.890        |
| Home & Kitchen     | F1-score| 0.619 | 0.684  | 0.698  | 0.742  | 0.741 | 0.744 | 0.762  | 0.786 | 0.815        |
|                    | NDCG    | 0.623 | 0.673  | 0.703  | 0.732  | 0.749 | 0.759 | 0.775  | 0.784 | 0.853        |
| Amazon Instant Video| F1-score| 0.651 | 0.904  | 0.743  | 0.755  | 0.759 | 0.783 | 0.795  | 0.811 | 0.862        |
|                    | NDCG    | 0.679 | 0.704  | 0.730  | 0.766  | 0.779 | 0.787 | 0.801  | 0.823 | 0.887        |

5.3 Parameter Settings

To scrutinize the capability of the MF model in handling data sparsity problem, we set the latent dimension $K = 256$ based on an extensive experimental observations. We set $\lambda = 0.1$ for Electronics, Home & Kitchen, Amazon Instant Video datasets. For the other three datasets we set $\lambda = 0.01$.

We implement FusionDeepMF on the Keras. We apply the mini-batch Adaptive Moment Estimation (Adam) as an optimizer and Mean Absolute Error as the loss function in this model. For neural network, we randomly initialize model parameters with a Gaussian distribution (with a mean of 0 and standard deviation of 0.01). We test the layers of (2, 4, 6, 8), the batch size of (128, 256, 512, 1024) and the epochs of (1, 5, 10, 15, 20, 25, 30, 40, 50, 60). Our model’s capability is determined based on the embedding size (predictive factors) of the last hidden layer. We perform experiment on the factors of (8, 16, 32, 64, 128). We also perform on larger factors but large factors may cause overfitting and degrade the performance. After observation, we choose 4 hidden layers, 512 batch size and 12 epochs in our deep learning model. For example, if the size of predictive factors is 64, then architecture...
of the neural CF layers is $512 - > 256 - > 128 - > 64$, and embedding size is 256. Details of observation are discussed later. For the FusionDeepMF with pre-training, $\gamma$ was set to 0.5, allowing the pre-trained MF and MLP to contribute equally to FusionDeepMF initialization. In our experiment, the respective optimal parameters are chosen based on our investigations. Based on the validation dataset, we tune the hyper-parameter.

5.4 Performance Analysis

To evaluate the performances of the FusionDeepMF, MF (only linear) and MLP (only non-linear) on our experimental dataset with different training size, five-fold cross validation is performed. For training purpose $x\%$ of data, for validation purpose $\left(\frac{100-x}{2}\right)\%$ of data are used and the remaining data is used for testing phase. Table 3 follows the above settings. For all the other experiments throughout the paper, we follow the following setting: To evaluate hyper-parameter of our model, five-fold cross validation is performed. In each fold, for training 70\% of data, for validation 15\% of data and for testing purpose the remaining 15\% of data are set in both our approach and baselines. We perform random sampling five times and calculate the mean as the final output for each.

Table 3 shows RMSE results from the MF, MLP and fusion of MF and MLP FusionDeepMF based on the data. For all the datasets, the FusionDeepMF performs better compared to the MF model and MLP model. The two separate embeddings, generated by the MF and MLP model simultaneously, can mutually reinforcement each other to learn better rating value. This is the advantage of the fusion of the MF and MLP. If we compare the performance between the MF and MLP, Table 3 shows that the MLP performs better on Electronics, Movies & TV and Home & Kitchen datasets. The linear kernel of MF model performs better on Books, Music and Amazon Instant Video datasets. For these three datasets, the linear kernel learns users’ features very accurately compared to non-linear kernel and improves overall performances. For this reason, the idea of the fusion of MF and MLP is very effective.

The comparison of RMSE, MAE results of predicted rating and original rating between the baseline models and our model based on the Amazon.com online review dataset is shown in Table 4. In this table, 3rd to 10th column present the performance of baseline models. The last column shows our model’s performances. For each category data, our model performs better significantly. The significant performance of our model indicates the importance of fusion between linear and non-linear kernel and the importance of reliability score. Our approach achieves better performances than DBNSA which indicates the effectiveness to learn users’ and products’ dual features embeddings based on users’ reliability score through our fusion model. In SBMF + R model the authors have considered users’ review text and reliability score based on helpful votes. But this reliability concept is different from our reliability concept and
we learn this features using linear and non-linear kernel based methods where \textit{SBMF} + \textit{R} model is based on probabilistic matrix factorization method. For the fusion between linearity and non-linearity our model learns users’ features very accurately.

\textit{DMF}, \textit{HCFDS} and \textit{CM} + \textit{RIM} are developed based on user-product interaction ratings and these three baselines use deep learning model to learn users’ features, where our model learns users’ features based on reliability score using linear and non-linear kernel. For this learning technique our model outperforms them. Our approach also achieves better performances over the \textit{ConvMF} and \textit{VMCF} for its dual embedding features learning capability. In Table 5 the evaluation of IR metrics between the baselines and our approach are presented. In this table, 3\textsuperscript{rd} to 10\textsuperscript{th} column presents the performance of the baselines. The last column shows the performance of our model. For each category data, our model \textit{FusionDeepMF} performs better significantly.

In section 4.6, we have discussed the time complexity of our model. In Table 6 we present the training time of the baseline models and our model. We conduct this experiment on a Intel Core 3.07 GHz CPU and 128 GB RAM with Ubuntu 16.04. Our approach is more effective compared to \textit{DMF}, \textit{VMCF}, \textit{CM} + \textit{RIM} and \textit{ConvMF} based on the total empirical runtime in the training phase. The training sample of \textit{VMCF} contains one observed rating and two side information as product images and also-viewed activity. \textit{VMCF} applies pre-trained \textit{CNN} to extract features from the pictures of products. Then \textit{PMF} based approach is applied to understand users’ rating activities from extracted features of product pictures and also-viewed information of users. Therefore, latent factor matrices of the users and products have to be updated twice for one training phase. The empirical training time of \textit{VMCF} for each iteration is lower than our methodology. But \textit{VMCF} converges approximately after 15 iterations, where our model converges after 12 iterations. For this reason the empirical total runtime of our methodology is slightly better than \textit{VMCF}. The time complexity of \textit{ConvMF} is \(O(k^3nR + k^3N + k^3M + n_{c.p.l.M}) \) \cite{18}. In this hybrid methodology, linear and non-linear kernel are integrated. \textit{CNN} is applied to analyze both review text and Gaussian noise. Then the objective function is used to optimize variables using users’ posted rating values and contextual data. \textit{ConvMF} converges approximately after 16 iterations. Due to analyzing review text, its empirical runtime is more than our approach.

If we analyze training times of \textit{DMF} and \textit{CM} + \textit{RIM}, we can find that empirical runtime of each iteration of these models is lower than our model, but total train time is higher than our model because our model converges approximately after 12 iterations, where these two baselines converge approximately after 21 iterations. This indicates the effectiveness of the learning technique with reliability score. \textit{SVD} ++ training sample contains only users’ posted rating values. \textit{SVD} ++ needs to update the latent factor matrices of the users and products once for each training sample, where our methodology learns users’ features using linear and non-linear way. For this reason the empirical runtime of our methodology is slightly higher than \textit{SVD} ++. Our model converges approximately after 12 iterations, where these two baseline \textit{HCFDS}
and DBNSA converge approximately after 16 or 17 iterations. If we analyze the train times of HCFDS and DBNSA, we can find that empirical runtime of each iteration of these models is lower than our model, but the total training time is higher than our model. The empirical runtime of our model is almost similar to SBMF + R, but according to the recommendation accuracy our model is the best.

Table 6: Comparison on total time needed (in second) to learn the parameters between baselines (present in 2nd to 9th column) and our approach (present in 10th column).

| Dataset             | SVD++ | VMCF | ConvMF | DMF | HCFDS | CM+RIM | SBMF+R | DBNSA | FusionDeepMF |
|---------------------|-------|------|--------|-----|-------|--------|--------|-------|-------------|
| Electronics         | 825   | 1215 | 1987   | 1688| 1137  | 1632   | 1033   | 1175  | 1030        |
| Books               | 912   | 1257 | 2216   | 1697| 1325  | 1641   | 1136   | 1320  | 1143        |
| Music               | 883   | 1085 | 2198   | 1560| 1025  | 1549   | 981    | 1033  | 995         |
| Movies & TV         | 871   | 1011 | 2011   | 1331| 980   | 1357   | 903    | 995   | 911         |
| Home & Kitchen      | 851   | 1307 | 2083   | 1463| 1201  | 1492   | 1113   | 1229  | 1107        |
| Amazon Instant Video| 837   | 1273 | 2077   | 1366| 1157  | 1384   | 1012   | 1163  | 1069        |

5.5 Utility of Pre-training

To understand the effectiveness of pre-training of FusionDeepMF, we observe the performance of two versions of our model with and without pre-training. As observed in Table 7, the FusionDeepMF with pre-training performs better. This result proves the effectiveness of the pre-training strategy in our model.

Table 7: Performance of MAE of FusionDeepMF for rating prediction with and without pre-training.

| Dataset            | with pre-training | without pre-training |
|--------------------|-------------------|----------------------|
| Electronics        | 0.884             | 0.911                |
| Books              | 0.753             | 0.778                |
| Music              | 0.710             | 0.730                |
| Movies & TV        | 0.701             | 0.729                |
| Home & Kitchen     | 0.798             | 0.835                |
| Amazon Instant Video| 0.838             | 0.855                |

5.6 Different settings of predictive factors

The sensitivity of the factors in each layer is high in our approach. The performances with different number of factors on the top final latent space are compared. The investigation on our method with four layer are arranged and we set the number of the total factors on the top layer from 8 to 128. In Table 8...
Table 8: Results for our method with different predictive factors of the final latent space.

| Dataset            | Metric | 8       | 16      | 32       | 64       | 128      |
|--------------------|--------|---------|---------|----------|----------|----------|
| Electronics        | RMSE   | 1.203   | 1.137   | 1.014    | 0.907    | **0.889** |
|                    | MAE    | 1.171   | 1.012   | 0.975    | 0.884    | **0.879** |
| Books              | RMSE   | 1.193   | 1.044   | 0.943    | **0.827** | 0.833    |
|                    | MAE    | 1.047   | 0.955   | 0.861    | **0.753** | 0.766    |
| Music              | RMSE   | 1.084   | 0.971   | 0.844    | **0.724** | 0.738    |
|                    | MAE    | 1.007   | 0.947   | 0.817    | **0.710** | 0.719    |
| Movies & TV        | RMSE   | 0.995   | 0.875   | 0.796    | **0.711** | 0.727    |
|                    | MAE    | 0.894   | 0.801   | 0.758    | **0.701** | 0.719    |
| Home & Kitchen     | RMSE   | 1.177   | 0.991   | 0.904    | 0.867    | **0.851** |
|                    | MAE    | 0.945   | 0.881   | 0.837    | 0.798    | **0.779** |
| Amazon Instant Video| RMSE | 1.109   | 0.989   | 0.907    | 0.896    | **0.879** |
|                    | MAE    | 1.032   | 0.965   | 0.918    | 0.848    | **0.820** |

Table 9: Performance of MAE with different number of layers of FusionDeepMF for rating prediction.

| Dataset            | MLP-2 | MLP-4 | MLP-6 | MLP-8 |
|--------------------|-------|-------|-------|-------|
| Electronics        | 0.911 | **0.884** | 0.889 | 0.894 |
| Books              | 0.781 | **0.753** | 0.761 | 0.767 |
| Music              | 0.733 | 0.710 | **0.701** | 0.716 |
| Movies & TV        | 0.729 | **0.701** | 0.711 | 0.719 |
| Home & Kitchen     | RMSE  | 0.821 | 0.798 | **0.789** | 0.801 |
| Amazon Instant Video| MAE  | 0.857 | **0.838** | 0.843 | 0.852 |

it is shown that the final layer with 64 factors achieves the best performance on Books, Music and Movies & TV datasets. For the other three datasets, the final layer with 128 factors gets the best performance. From this observation we can say that the final representations with more predictive factors might be more effective when data is small and sparse in nature.

5.7 Different Number of Layers

We investigate whether deep neural network is suitable for our recommendation system or not. We experiment with different number of hidden layers and Table 9 shows the results. In this table, 2nd to 5th column show the performance of the FusionDeepMF model with different number of hidden layers. MLP-2 means two hidden layers are used in the FusionDeepMF model. In most of the cases, we are getting good performance when four hidden layers are used. As shown in Table 9 on Electronics, Books, Movies & TV, Amazon Instant Video datasets our model with layer 4 illustrates the best performance. For the other two datasets Music and Home & Kitchen, our model with layer 6 illustrates the best performance.

We also observed the similar performance of our approach with different number of layers when RMSE is applied. From this observation, we can say...
that using a deep network structure with stacking more hidden layers is not always beneficial for a recommendation task.

5.8 Ranking of Top-t products recommendation

Evaluation of ranking metrics@Top-t between the baselines and our approach are represented in Table 10. The 1st column of this table presents different categories of data, 3rd to 10th column show the performance of the baselines. The last column shows the evaluation of our approach. For all category data, FusionDeepMF performs the best.

Table 10: Comparison with baselines (present in 3rd to 10th column) and our methodology (present in last column) based on Top-t products recommendation.

| Dataset          | Metrics | SVDF++ | ConvMF | VMCF | SMCF+R | DMF | HCFDS | CM+RIM | DINSA | FusionDeepMF |
|------------------|---------|--------|--------|------|--------|-----|-------|--------|-------|--------------|
| Electronics      | NDCG    | 0.671  | 0.696  | 0.707 | 0.744  | 0.751| 0.771 | 0.782  | 0.783 | 0.846        |
|                  | MAP     | 0.602  | 0.621  | 0.688 | 0.732  | 0.740| 0.745 | 0.765  | 0.771 | 0.828        |
|                  | F1@5    | 0.497  | 0.557  | 0.597 | 0.601  | 0.609| 0.617 | 0.628  | 0.643 | 0.733        |
|                  | F1@10   | 0.511  | 0.587  | 0.604 | 0.611  | 0.620| 0.634 | 0.641  | 0.655 | 0.747        |
| Books            | NDCG    | 0.644  | 0.656  | 0.741 | 0.764  | 0.783| 0.797 | 0.804  | 0.811 | 0.881        |
|                  | MAP     | 0.599  | 0.611  | 0.688 | 0.727  | 0.740| 0.751 | 0.771  | 0.782 | 0.837        |
|                  | F1@5    | 0.498  | 0.587  | 0.647 | 0.649  | 0.653| 0.658 | 0.660  | 0.662 | 0.771        |
|                  | F1@10   | 0.511  | 0.603  | 0.651 | 0.657  | 0.661| 0.667 | 0.670  | 0.675 | 0.784        |
| Music            | NDCG    | 0.688  | 0.704  | 0.728 | 0.754  | 0.780| 0.789 | 0.803  | 0.823 | 0.883        |
|                  | MAP     | 0.602  | 0.609  | 0.704 | 0.733  | 0.750| 0.767 | 0.778  | 0.791 | 0.831        |
|                  | F1@5    | 0.576  | 0.604  | 0.669 | 0.714  | 0.721| 0.727 | 0.733  | 0.742 | 0.778        |
|                  | F1@10   | 0.598  | 0.622  | 0.689 | 0.722  | 0.728| 0.731 | 0.639  | 0.750 | 0.782        |
| Movies & TV      | NDCG    | 0.638  | 0.711  | 0.728 | 0.764  | 0.780| 0.791 | 0.811  | 0.845 | 0.890        |
|                  | MAP     | 0.605  | 0.694  | 0.709 | 0.729  | 0.739| 0.753 | 0.785  | 0.803 | 0.844        |
|                  | F1@5    | 0.567  | 0.600  | 0.697 | 0.711  | 0.720| 0.727 | 0.732  | 0.740 | 0.779        |
|                  | F1@10   | 0.587  | 0.613  | 0.702 | 0.717  | 0.724| 0.731 | 0.729  | 0.747 | 0.784        |
| Home & Kitchen   | NDCG    | 0.584  | 0.604  | 0.614 | 0.624  | 0.644| 0.661| 0.671  | 0.685 | 0.853        |
|                  | MAP     | 0.598  | 0.621  | 0.694 | 0.714  | 0.730| 0.743 | 0.761  | 0.775 | 0.822        |
|                  | F1@5    | 0.510  | 0.566  | 0.608 | 0.609  | 0.622| 0.634 | 0.642  | 0.647 | 0.701        |
|                  | F1@10   | 0.568  | 0.602  | 0.633 | 0.644  | 0.661| 0.671 | 0.676  | 0.680 | 0.717        |
| Amazon Instant Video | NDCG | 0.679  | 0.704  | 0.730 | 0.766  | 0.799| 0.797 | 0.803  | 0.823 | 0.887        |
|                  | MAP     | 0.665  | 0.699  | 0.713 | 0.751  | 0.763| 0.774 | 0.791  | 0.804 | 0.853        |
|                  | F1@5    | 0.557  | 0.681  | 0.720 | 0.722  | 0.731| 0.737 | 0.741  | 0.747 | 0.779        |
|                  | F1@10   | 0.576  | 0.694  | 0.739 | 0.741  | 0.743| 0.749 | 0.753  | 0.759 | 0.785        |

6 Conclusion and Future Work

In this work, we propose the FusionDeepMF model, where MF integrates a linear kernel to model the latent feature vectors based on users’ rating activity and reliability score and on the other hand MLP uses a non-linear kernel to learn users’ latent feature vectors. This model uses a tuning-parameter determining the trade-off between the dual embeddings that are generated from the linear and non-linear kernels. We observe that the new way of modeling users’ reliability score not only characterizes the users’ latent feature vectors accurately but also presents a better interpretation of how the similarity of
users’ reliability score affects rating prediction. The raters also play a vital role to notify the reliability of a reviewer’s review and help the model to learn the latent factor of the reviewer more accurately to predict rating.

We would like to further improve the performances of our approach and would like to experiment on other datasets. There are many existing e-commerce companies, which have links with social media where users share their opinion on their social pages. We would like to investigate if social network feedback can be merged with the reliability score (review network feedback) to learn users’ latent feature vectors more accurately. Many online e-commerce companies share users’ posted feedback on their social sites. We would prefer to analyze how social networks can be used to learn users’ preferable zones.

We would prefer to experiment on an attention layer based deep learning model to extract users’ preferable products’ features. Some micro behaviors activities such as clicking activities, searching activity, add to cart, order, dwell time, etc. will also be considered in our model. In [39], the authors are considering “also-viewed” categories for quality vector embedding. We will consider on other relations among products, like “also-bought”, “frequently-bought-together” and “bought-after-buying.” It has been decided to check whether the above relationships are effective in projecting good quality product vector embedding and how to exactly suit in our methodology. This concept gives us other fascinating way to extend our approach including more characteristics of users.

References

1. Bobadilla, J., Gutiérrez, A., Ortega, F., Zhu, B.: Reliability quality measures for recommender systems. Information Sciences 442, 145–157 (2018)
2. Chen, C., Zhang, M., Liu, Y., Ma, S.: Neural attentional rating regression with review-level explanations. In: Proceedings of the 2018 World Wide Web Conference on World Wide Web, pp. 1583–1592. International World Wide Web Conferences Steering Committee (2018)
3. Chen, R.C., et al.: User rating classification via deep belief network learning and sentiment analysis. IEEE Transactions on Computational Social Systems 6(3), 535–546 (2019)
4. Chen, S., Peng, Y.: Matrix factorization for recommendation with explicit and implicit feedback. Knowledge-Based Systems 158, 109–117 (2018)
5. Cheng, W., Shen, Y., Zhu, Y., Huang, L.: Delf: A dual-embedding based deep latent factor model for recommendation. In: IJCAI, pp. 3329–3335 (2018)
6. Chin, J.Y., Zhao, K., Joty, S., Cong, G.: Aur: Aspect-based neural recommender. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 147–156. ACM (2018)
7. Dong, X., Yu, L., Wu, Z., Sun, Y., Yuan, L., Zhang, F.: A hybrid collaborative filtering model with deep structure for recommender systems. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
8. Elkahky, A.M., Song, Y., He, X.: A multi-view deep learning approach for cross domain user modeling in recommendation systems. In: Proceedings of the 24th International Conference on World Wide Web, pp. 278–288. International World Wide Web Conferences Steering Committee (2015)

9. Fu, M., Qu, H., Yi, Z., Lu, L., Liu, Y.: A novel deep learning-based collaborative filtering model for recommendation system. IEEE transactions on cybernetics 49(3), 1084–1096 (2018)

10. Gao, H., Tang, J., Hu, X., Liu, H.: Exploring temporal effects for location recommendation on location-based social networks. In: Proceedings of the 7th ACM conference on Recommender systems, pp. 93–100. ACM (2013)

11. Gu, Q., Zhou, J., Ding, C.: Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs. In: Proceedings of the 2010 SIAM International Conference on Data Mining, pp. 199–210. SIAM (2010)

12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778 (2016)

13. He, R., McAuley, J.: Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In: proceedings of the 25th international conference on world wide web, pp. 507–517. International World Wide Web Conferences Steering Committee (2016)

14. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web, pp. 173–182. International World Wide Web Conferences Steering Committee (2017)

15. Ji, K., Sun, R., Li, X., Shu, W.: Improving matrix approximation for recommendation via a clustering-based reconstructive method. Neurocomputing 173, 912–920 (2016)

16. Jacobi, J.A., Benson, E.A.: System and methods for collaborative recommendations (2000). US Patent 6,064,980

17. Kim, D., Park, C., Oh, J., Lee, S., Yu, H.: Convolutional matrix factorization for document context-aware recommendation. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 233–240. ACM (2016)

18. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

21. Koren, Y.: 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)
22. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42(8), 30–37 (2009)
23. Li, G., Chen, Q.: Exploiting explicit and implicit feedback for personalized ranking. Mathematical Problems in Engineering 2016 (2016)
24. Li, H.L., Cao, J., Jiang, H., Alsaedi, A.: Finite-time synchronization of fractional-order complex networks via hybrid feedback control. Neurocomputing 320, 69–75 (2018)
25. Liang, D., Alotaet, J., Charlin, L., Blei, D.M.: Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In: Proceedings of the 10th ACM conference on recommender systems, pp. 59–66. ACM (2016)
26. Linden, G., Smith, B., York, J.: Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet computing 7(1), 76–80 (2003)
27. Ling, G., Lyu, M.R., King, I.: Ratings meet reviews, a combined approach to recommend. In: Proceedings of the 8th ACM Conference on Recommender systems, pp. 105–112. ACM (2014)
28. Maiti, A., Mandal, S.: System and method for determining company performance (2019). Publication Date (U/S 11A): 21/06/2019, Application No: 201731045513, Indian Patent (Published)
29. Mandal, S., Maiti, A.: Explicit feedbacks meet with implicit feedbacks: A combined approach for recommendation system. In: International Conference on Complex Networks and their Applications, pp. 169–181. Springer (2018)
30. Mandal, S., Maiti, A.: Explicit feedback meet with implicit feedback in gpmf: a generalized probabilistic matrix factorization model for recommendation. Applied Intelligence 50(6), 1955–1978 (2020)
31. Mandal, S., Maiti, A.: Deep collaborative filtering with social promoter score-based user-item interaction: a new perspective in recommendation. Applied Intelligence 51(11), 7855–7880 (2021)
32. Mandal, S., Maiti, A.: Rating prediction with review network feedback: A new direction in recommendation. IEEE Transactions on Computational Social Systems (2021)
33. McAuley, J., Targett, C., Shi, Q., Van Den Hengel, A.: Image-based recommendations on styles and substitutes. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 43–52. ACM (2015)
34. Mnih, A., Salakhutdinov, R.R.: Probabilistic matrix factorization. In: Advances in neural information processing systems, pp. 1257–1264 (2008)
35. Mooney, R.J., Roy, L.: Content-based book recommending using learning for text categorization. In: Proceedings of the fifth ACM conference on Digital libraries, pp. 195–204. ACM (2000)
36. Moradi, P., Ahmadian, S.: A reliability-based recommendation method to improve trust-aware recommender systems. Expert Systems with Applications 42(21), 7386–7398 (2015)
37. Núñez-Valdez, E.R., Quintana, D., Crespo, R.G., Isasi, P., Herrera-Viedma, E.: A recommender system based on implicit feedback for se-
lective dissemination of ebooks. Information Sciences 467, 87–98 (2018)

38. O’Donovan, J., Smyth, B.: Trust in recommender systems. In: Proceedings of the 10th international conference on Intelligent user interfaces, pp. 167–174. ACM (2005)

39. Park, C., Kim, D., Oh, J., Yu, H.: Do also-viewed products help user rating prediction? In: Proceedings of the 26th International Conference on World Wide Web, pp. 1113–1122. International World Wide Web Conferences Steering Committee (2017)

40. Resnick, P., Varian, H.R.: Recommender systems. Communications of the ACM 40(3), 56–58 (1997)

41. 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. ACM (2001)

42. Schafer, J.B., Konstan, J.A., Riedl, J.: E-commerce recommendation applications. Data mining and knowledge discovery 5(1-2), 115–153 (2001)

43. Shen, R.P., Zhang, H.R., Yu, H., Min, F.: Sentiment based matrix factorization with reliability for recommendation. Expert Systems with Applications 135, 249–258 (2019)

44. Shi, C., Hu, B., Zhao, W.X., Philip, S.Y.: Heterogeneous information network embedding for recommendation. IEEE Transactions on Knowledge and Data Engineering 31(2), 357–370 (2018)

45. Srivastava, N., Salakhutdinov, R.R.: Multimodal learning with deep boltzmann machines. In: Advances in neural information processing systems, pp. 2222–2230 (2012)

46. Wang, H., Wang, N., Yeung, D.Y.: Collaborative deep learning for recommender systems. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1235–1244. ACM (2015)

47. Wang, J., De Vries, A.P., Reinders, M.J.: Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 501–508. ACM (2006)

48. Wang, R., Cheng, H.K., Jiang, Y., Lou, J.: Tdcr: A two-stage deep learning based recommendation model. Expert Systems with Applications 145, 113116 (2020)

49. Weng, S.S., Liu, M.J.: Feature-based recommendations for one-to-one marketing. Expert Systems with Applications 26(4), 493–508 (2004)

50. Xiong, R., Wang, J., Zhang, N., Ma, Y.: Deep hybrid collaborative filtering for web service recommendation. Expert Systems with Applications 110, 191–205 (2018)

51. Xue, H.J., Dai, X., Zhang, J., Huang, S., Chen, J.: Deep matrix factorization models for recommender systems. In: IJCAI, pp. 3203–3209 (2017)

52. Yu, Y., Gao, Y., Wang, H., Wang, R.: Joint user knowledge and matrix factorization for recommender systems. World Wide Web 21(4), 1141–1163 (2018)
53. Yu, Y., Wang, C., Wang, H., Gao, Y.: Attributes coupling based matrix factorization for item recommendation. Applied Intelligence 46(3), 521–533 (2017)
54. Zhang, F., Yuan, N.J., Lian, D., Xie, X., Ma, W.Y.: Collaborative knowledge base embedding for recommender systems. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp. 353–362 (2016)
55. Zhang, S., Wang, W., Ford, J., Makedon, F.: Learning from incomplete ratings using non-negative matrix factorization. In: Proceedings of the 2006 SIAM International Conference on Data Mining, pp. 549–553. SIAM (2006)
56. Zhang, S., Yao, L., Xu, X.: Autosvd++: An efficient hybrid collaborative filtering model via contractive auto-encoders. In: Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval, pp. 957–960. ACM (2017)
57. Zhang, W., Zhang, X., Wang, H., Chen, D.: A deep variational matrix factorization method for recommendation on large scale sparse dataset. Neurocomputing 334, 206–218 (2019)
58. Zheng, L., Noroozi, V., Yu, P.S.: Joint deep modeling of users and items using reviews for recommendation. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, pp. 425–434 (2017)
59. Zheng, X., He, W., Li, L.: Distributed representations based collaborative filtering with reviews. Applied Intelligence 49(7), 2623–2640 (2019)
60. Zheng, Y., Mobasher, B., Burke, R.: Similarity-based context-aware recommendation. In: International Conference on Web Information Systems Engineering, pp. 431–447. Springer (2015)