Improving recommendation by deep latent factor-based explanation

Sixun Ouyang, Aonghus Lawlor
Insight Centre for Data Analytics
University College Dublin, Ireland
{sixun.ouyang, aonghus.lawlor}@insight-centre.org

Abstract

The latent factor methods and explanation algorithms constitute the foundation of many advanced explainable recommender systems. However, interpreting the high-dimensional latent factors has not been sufficiently addressed and continuously becomes a challenging work. Besides, only a few works have researched the use of explanation to improve recommendations. In this paper, we propose a deep learning method that generates high-quality latent factor-based explanations and efficiently ameliorating recommendations. We conduct top-K items ranking experiment on two real-world datasets and show that our method outperforms nine currently state-of-the-art recommender systems in five ranking metrics. Moreover, we conduct a qualitative and quantitative analysis of users' latent factors and reveal that we continually offer the best latent representations.

Introduction

Latent factorisation methods play a vital role in the metabolism of recommender systems. From early works (Strang 1993; Lee and Seung 2001) to recent advanced neural networks (Wang et al. 2018; Ouyang and Lawlor 2019). At the same time, the process of providing explanations for recommendations also receives substantial attention (Tintarev and Masthoff 2015). However, making interpretations of latent factors remains a major challenge, where the main challenge is that it is difficult to explain high-dimensional abstract numeric. The successful work that solved this problem is NEAR (Ouyang and Lawlor 2019), which aims to figure out the most important factor for users as the explanation. Nevertheless, finding out just one latent factor is not enough. Moreover, the interpretation generation process of NEAR is inefficient and expensive. As such, the first problem this paper aims to address is providing an effective approach to generate explanations for all latent factors of both users and items.

On the other hand, most explanation works (Lacic, Kowald, and Lex 2016; Costa et al. 2018) concentrate on investigating the value of trust and fairness of interpretation but neglect the affiliation between recommendation and explanation. The recommendation explanations are designed to serve recommender systems. It can endow persuasiveness to recommender systems, and it also can make recommender systems to be more precise. The most recent work that uses explanations to strength recommendations is (Wu et al. 2019), which critiques interpretations when making predictions. Nevertheless, their method critiques explanations in the inference step instead of learns the critiqued outcomes in the training step. Therefore, the next goal for this paper is building a feasible training fashion to contribute to the area of using explanations to improve recommendation.

In this paper, we borrow the idea of NEAR and propose a latent factor-based explanation generation algorithm through deep neural networks. Besides, we introduce a new manner of utilising interpretations to train recommender systems. Experimental results show that we surpass nine baselines and attain the state-of-the-art performance in top-K item ranking. We also qualitatively and quantitatively analyse the quality of users' latent representations and demonstrate that our method can learn superior latent embeddings than baselines.

Related Work

The definition of the term *explanation* has not been unified. According to the overview of(Guidotti et al. 2018), the notion of explanation is encompassed in two parts: global interpretability and local explainability. The first concept provides a general understanding of the inner logic of a transparent system, for example, Frosst *et al.*(Frosst and Hinton 2017) explained neural networks through a path of a soft decision tree. By contrast, the second idea aims to find the reasons for predictions from opaque intelligent systems. In this vein, Ribeiro *et al.*(Ribeiro, Singh, and Guestrin 2016) proposed a local interpretable model-agnostic explanation, which named as LIME, to faithfully interpret classifications. Wu *et al.*(Wu et al. 2019) introduced a method jointly learning the recommendations and key-phrase interpretations simultaneously. For the *explanation* in this paper, we are interested in explaining recommendations locally.

There have been many longitudinal studies involving local interpretation which have reported that the feature selection plays a critical role in the maintenance of locally explanation generation. The target of this technology is identifying the features with the highest relevance to an agnostic prediction. Geng *et al.*(Geng et al. 2007) addressed redundant selection problem when selecting features, by considering similarities between features. Lai *et al.*(Lai et al. 2013) proposed a joint convex optimization method that minimises ranking errors and conducting feature selection simultaneously. In addition to distance algorithms, other statistical algorithms and machine learning models are employed in this topic (Li et al. 2018). For example, the principal component analysis method is commonly used in this topic (Yu and Liu 2003). Besides, Recent evidence suggests that deep learning techniques can grasp personal features accurately (Costa et al. 2018). Unlike the above works, which conduct explanations on visible features, generating explanations on latent factors is more complicated, because a latent factor is just a digital number. To do so, we are not attempting to explain a numeric number, but rather to make the *factor selection* exposing the connection between latent factors and recommendations.

The recommendation explanation is different from the explanation of other methods. The initial goal of recommendation interpretation is to persuade customers to believe the recommendation (Tintarev and Masthoff 2011). In the past decade, several researchers have sought to determine how to produce recommendation interpretation. Vig *et al.*(Vig, Sen, and Riedl 2009) used community tags to explain related recommendations. Symeonidis *et al.*(Symeonidis, Nanopoulos, and Manolopoulos 2009) introduced an approach that using users' feature similarity and rating similarity to provide interpretation. Besides heuristic and tag data, using review data is also an alternative. Lacic *et al.*(Lacic, Kowald, and Lex 2016) exploited the most contributed features of airline reviews and promoted travelers' satisfaction. Chen *et al.*(Chen and Wang 2017) proposed an explainable recommender system that supplies interpretation by feature sentiments in product reviews. Costa *et al.*(Costa et al. 2018) indicated an alternative approach, machine-generated reviews, to offer recommendation explanations. Recently, some other works believe explanation can also be used to modify recommendation behavior. NEAR (Ouyang and Lawlor 2019) generated perturbed user embeddings by their latent factor explanation, and improved recommendation performance. Wu *et al.*(Wu et al. 2019) introduced an explanation critiquing process for deep neural network-based recommender systems showing valuable enhancement for recommendations.

However, the work of generating explanations on latent factors has not been thoroughly addressed. Also, few works have integrated the explanation into the training step of recommender systems for advanced recommendation performance. As such, This research sheds new light on reforming recommendation behavior through latent factor-based explanation.

Proposed Method

In this paper, we develop the current latent explanation generator NEAR(Ouyang and Lawlor 2019) by neural networks and propose a new fashion to reform recommendation behavior. We name our method as R-NEAR. Comparing with NEAR, we argue that R-NEAR can learn more productive information and produce more meaningful factor-based explanations and lead to more precise recommendations. Intrinsically, our method works on the *latent embedding* to identify the latent connections and plays as an extension system to any latent embedding based recommender systems, as shown in Figure 1. We category our methods into two modules: explanation generation and rectify recommendation. The first module aims to generate explanations, while the second one targets on reforming recommendation performance. In the following sections, we start to discuss the basic unexplained prediction approach of latent factor-based recommendation methods, which provides the general intuition of why our method is needed. Next, we introduce how to generate interpretations by the latent factors of recommender systems. We then outline how to ameliorate recommendations by the interpretations.

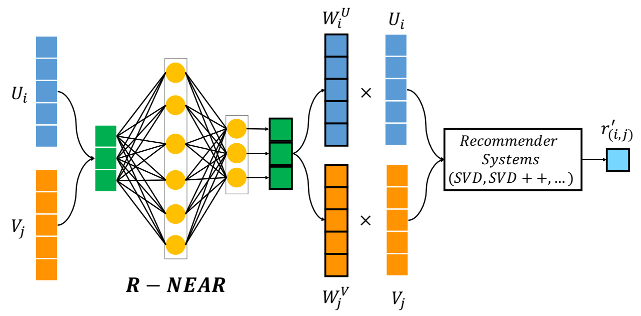

Figure 1: The proposed R-NEAR model. We produce latent factor based explanation and use these interpretations to reform recommender systems. Here, the inputs to our method are the user representation U_i and the item representation V_j, while the outputs are user explainable weights W_i^U, item explainable weights W_j^V, and the promoted recommendation rating $r'_{(i,j)}$.

Non-explainable Recommendation

In this section, we summarise the general way of factorised methods to make predictions and discuss how we can make explanations on latent factors. In latent embedding based methods, the observed user i and item j are first encoded into N dimensional vector embedding U_i and V_j. Here, $U \in \mathcal{R}^{P\times N}$ is the user embedding matrix, $V \in \mathcal{R}^{Q\times N}$ is the item embedding matrix, where P and Q represents the number of total users and items respectively. To produce a recommendation, general factorisation methods, for example, matrix factorisation, make dot product on $U-i$ and V_j estimating the rating $\hat{r}_{i,j}$ that the user i might gives to the item j. We summarise this forward calculation as:

$$\hat{r}_{(i,j)} = U_i V_j \tag{1}$$

In this computation, the high-dimensional latent representation U_i and V_j are the key aspects. Several works attempt to make explanations on these latent factors, for example, NEAR(Ouyang and Lawlor 2019) aims to find which factors are important as the interpretation. Nevertheless, NEAR uses a brute-force method that detects the factor which carries out the minimum recommendation loss, which is Inefficient. Besides, it only updates one user factor at a time, which is not enough to reform recommendations.

In this paper, we aim to provide exhaustive interpretation on both the user and the item side. We introduce the user explainable weights $W_i^U \in \mathcal{R}^{1\times N}$ and item explainable weights $W_j^V \in \mathcal{R}^{1\times N}$. The size of these weights equals the size of embeddings of users and items. Similar to NEAR, the values in these explainable weights reflect the importance of corresponding latent factors. The higher value in W_i^U or W_j^V, the more indispensable the related latent factor is, or vice versa. We formulate the calculation procedure of user explainable weights W_i^U and item explainable weights W_j^V in the following equation.

$$W_i^U, W_j^V = f_{R-NEAR}(U_i, V_j) \tag{2}$$

Generate Explanation

We spread out the discussion of technical details of f_{R-NEAR} in this section. In order to model W_i^U and W_j^V, we propose the Multi-layer Neural Networks, since deep learning methods are the representation-learning algorithm that can learn excellent representations with multiple levels. What is more, they magnify the important parts and restrain irrelevances (LeCun, Bengio, and Hinton 2015), which meets the demands of our purpose. In our method, we employ fully connected neural networks. To learn more abstract and comprehensive non-linear representation, we activate the hidden layer by the $Relu$ activation. The $Relu$ function forces the output of unimportant units to be 0 to let the network become sparse, and the neurons are less dependent, and we can alleviate the over-fitting problem. We demonstrate the calculation of the hidden layer in Equation 3. Here, $[,]$ denotes the concatenation operation, $W_h \in \mathcal{R}^{2N\times h}$ is the hidden weights, h is the number of hidden units, and b_h is the bias.

$$H_{(i,j)} = Relu(W_h[U_i, V_j] + b_h) \tag{3}$$

As aforementioned, the goal of our method is to learn the latent relationship between the user-item pairs and the predicted rating, and presents these correlations on the user explainable weights W_i^U and item explainable weights W_j^V. Therefore, we apply the predicted rating $\hat{R}_{i,j}$ as the target of the multi-layer neural networks. We stack a linear transformation layer on the hidden layer, due to the linear explanations are proven to be faithful in many works(Ribeiro, Singh, and Guestrin 2016; Ouyang and Lawlor 2019). We present the computation of our explanation generation in Equation 4, where, $o_{(i,j)}$ is the estimation of recommended rating, $W^{UV} \in \mathcal{R}^{1\times 2N}$ is the explainable weights. We then separate explainable weights W^{UV} into user explainable weights W_i^U and item explainable weights W_j^V by the embedding size N.

$$\begin{gathered} o_{(i,j)} = W^{UV} H_{(i,j)} + b_o \\ W_i^U, W_j^V = W_{:N}^{UV}, W_{N:}^{UV} \end{gathered} \tag{4}$$

To train R-NEAR, we illustrate the proposed cost function in Equation 5. We measure the explanation loss by the squared error between the prediction and the recommended rating. Moreover, a recent explanation method suggests $L1$ regularisation creates a sparse weight, which is suitable for feature (factor) selection (Ribeiro, Singh, and Guestrin 2016). Thus, we add a $L1$ regularisation on the explainable weights in this function.

$$\mathcal{J}_{R-NEAR} = (\hat{r}_{i,j} - o_{(i,j)})^2 + \lambda \|W^{UV}\|_1 \tag{5}$$

Improve Recommendation

A number of explanation works have reported different ways to ameliorate recommendation behavior, for instance, NEAR attempts to manually alter user embedding through their factor-based explanation, Wu *et al.*(Wu et al. 2019) try to learn recommendation and key-phrase explanation jointly. However, these approaches are inefficient and can not be scaled. In this section, we aim to offer some critical insights into the process of improving recommendations by explanation.

In the previous section, we have calculated the user explainable weights W_i^U and item explainable weights W_j^V. On the one hand, the values in these weights indicate the importance of each latent factor. On the other hand, the vital point of personalised recommendation is amplifying relevant aspects and suppressing irrelevant variations. Therefore, we argue that R-NEAR can make a more precise personalised recommendation through multiplying the W_i^U and W_j^V with corresponded user embedding and item embedding. We reveal this computation in Equation 6, where *odot* denotes the element-wise product, U_i' and V_j' is the new embedding of user i and item j, and $r'_{(i,j)}$ is the enhanced predicted rating. By doing this, our algorithm works as an extension to other recommender systems, which ensure strong scalability and robustness.

$$\begin{gathered} U_i' = U_i \odot W_i^U \\ V_j' = V_j \odot W_j^V \\ r'_{(i,j)} = U_i' V_j' \end{gathered} \tag{6}$$

In terms of learning the improvements for recommendations, we develop a new cost function for recommender systems. In our method, we apply the Root Mean Square Error (RMSE), the most popular loss function in the recommendation filed, to leverage the differentiation between recommendation and ground truth. We show the loss function in Equation 7, where $r'_{(i,j)}$ is the improved recommendation, $r_{(i,j)}$ is the ground truth rating, and R is the possible user-item pairs in training set.

$$\mathcal{J}_{rs} = \sqrt{\frac{1}{\hat{R}} \sum_{\hat{r}_{i,j} \in \hat{R}} (r_{(i,j)} - r'_{i,j})^2} \tag{7}$$

In this end, we can interactively update R-NEAR and recommender systems. Every time we train the recommender systems, we can get more accurate predictions so that we can get better explanations. Similarly, every time we train our explanation system, we can achieve more significant factor

Model	R-Precision	NDCG	MAP@5	MAP@10	MAP@20	Precision@5	Precision@10	Precision@20	Recall@5	Recall@10	Recall@20
NCF	0.0378	0.0824	0.0435	0.0394	0.0342	0.0396	0.0332	0.0266	0.0539	0.0907	0.1428
E-NCF	0.0392	0.0851	0.0449	0.0411	0.0356	0.0418	0.0349	0.0271	0.0579	0.0956	0.1456
CE-NCF	0.0401	0.0853	0.0458	0.0412	0.0354	0.0404	0.0344	0.0270	0.0576	0.0942	0.1447
VNCF	0.0372	0.0809	0.0408	0.0384	0.0341	0.0393	0.0337	0.0275	0.0526	0.0869	0.1415
E-VNCF	0.0376	0.0833	0.0436	0.0399	0.0351	0.0395	0.0344	0.0281	0.0511	0.0880	0.1446
CE-VNCF	0.0374	0.0827	0.0425	0.0398	0.0351	0.0398	0.0340	0.0281	0.0516	0.0904	0.1440
SVD	0.0275	0.0751	0.0564	0.0507	0.0436	0.0503	0.0418	0.0332	0.0563	0.0917	0.1421
SVD++	0.0354	0.0763	0.0584	0.0509	0.0458	0.0509	0.0402	0.0335	0.0582	0.0938	0.1568
NEAR	0.0393	0.0824	0.0608	0.0550	0.0476	0.0547	0.0451	0.0369	0.0621	0.0986	0.1592
R-NEAR	**0.0411**	**0.0835**	**0.0622**	**0.0555**	**0.0479**	**0.0548**	**0.0457**	**0.0370**	**0.0632**	**0.1011**	**0.1602**

Table 1: Top-K item rank of Amazon CDs&Vinyl dataset.

Model	R-Precision	NDCG	MAP@5	MAP@10	MAP@20	Precision@5	Precision@10	Precision@20	Recall@5	Recall@10	Recall@20
NCF	0.0603	0.1056	0.0808	0.0750	0.0679	0.0747	0.0661	0.0571	0.0530	0.0930	0.1534
E-NCF	0.0577	0.1025	0.0762	0.0715	0.0653	0.0718	0.0642	0.0561	0.0520	0.0907	0.1494
CE-NCF	0.0678	0.1259	0.0934	0.0881	0.0815	0.0878	0.0800	0.0709	0.0605	0.1086	0.1872
VNCF	0.0677	0.1248	0.0928	0.0879	0.0811	0.0874	0.0802	0.0704	0.0598	0.1080	0.1861
E-VNCF	0.0678	0.1259	0.0934	0.0881	0.0815	0.0878	0.0800	0.0708	0.0604	0.1086	0.1872
CE-VNCF	0.0716	0.1301	0.0920	0.0892	0.0834	0.0897	0.0839	0.0735	0.0628	0.1166	0.1974
SVD	0.0613	0.1204	0.1410	0.1299	0.1163	0.1290	0.1130	0.0958	0.0784	0.1321	0.2123
SVD++	0.0622	0.1236	0.1423	0.1237	0.1072	0.1320	0.1089	0.0986	0.0797	0.1363	0.2138
NEAR	0.0728	0.1299	0.1495	0.1379	0.1237	0.1369	0.1194	0.1016	0.0847	0.1412	0.2250
R-NEAR	**0.0734**	**0.1319**	**0.1506**	**0.1383**	**0.1238**	**0.1385**	**0.1205**	**0.1018**	**0.0856**	**0.1426**	**0.2265**

Table 2: Top-K item rank of BeerAdvocate dataset.

selection so that we can achieve more precise recommendations.

Experiments

We now present the recommendation and explanation evaluation of our method to answer the following research questions:

- Can we show exception performance on Top-K items ranking, comparing with other state-of-art recommender systems?
- Can we provide high-quality personalised latent representation?

Experimental Settings

Datasets Our experiments are conducted on 2 real-world datasets: Amazon CDs&Vinyl [1] (He and McAuley 2016) and BeerAdvocate (McAuley and Leskovec 2013), where the CD dataset contains 1097593 review records and the Beer dataset consists of 528871 rows. For each dataset, we randomly split them into training, validation, and test sets by the fraction of 80%, 10%, and 10%. Note, train set is used to train models, validate set aims to find the best model, while the following experiments are executed on the test set.

[1] `http://jmcauley.ucsd.edu/data/amazon`

Baselines In our experiments, we run our method with SVD, the most well-known recommender systems, and compare it with nine state-of-the-art recommender systems. To make a fair comparison, we keep the same negative sample size as in (Wu et al. 2019).

- SVD(Strang 1993): Singular Value Decomposition, a popular collaborative filtering methodology learning latent relationship between users and items.
- SVD++(Koren 2008): An extension scenario of SVD which uses implicit information.
- NEAR(Ouyang and Lawlor 2019): A variant SVD enhanced with NEAR. NEAR aims to find the users' most critical factor for SVD.
- NCF(He et al. 2017): A novel recommender system based on deep learning techniques.
- E-NCF(Wu et al. 2019): Explainable NCF, which learns recommendation and explanation jointly.
- CE-NCF(Wu et al. 2019): An explainable E-NCF variation applied the critiquing mechanism.
- VNCF(Wu et al. 2019): A variation NCF. state-of-the-art non-explainable recommendation method.

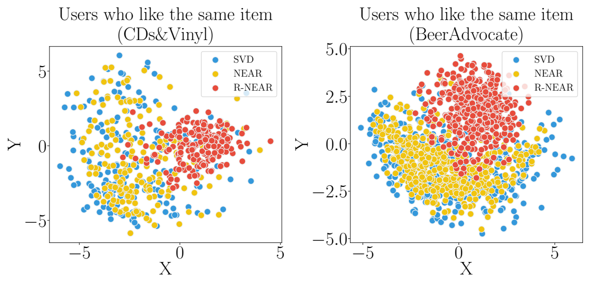

Figure 2: PCA 2D dimension reduction of user latent representation on CDs&Vinyl and BeerAdvocate datasets. We demonstrate the users who like the same item. Blue points represent the embedding of SVD algorithm. Yellow points denote the learned embedding of NEAR method. Red points stand for the rectified embedding of R-NEAR. Notably, R-NEAR shows the best compactness for the group of users who have same interests.

- E-VNCF(Wu et al. 2019): Explainable VNCF without critiquing mechanism.
- CE-VNCF(Wu et al. 2019): The state-of-the-art variational extension of explainable VNCF boosted by the critique mechanism.

Evaluation protocols We measure the overall performance by Top-K items recommendation ranking. Thus, we apply a list of ranking evaluation metrics to leverage both recommendation and interpretation conduct:

- **Precision@K** Precision@K measures the proportion of the items that the user prefers have been recommended in Top-K ranking among K.

$$Precision@K = \frac{\#\{\textit{(preferred items)} \cap \textit{(Top-K)}\}}{K} \quad (8)$$

- **Recall@K** Recall reflects the ratio of the items that the user prefers have been recommended in Top-K ranking among the whole preferred items.

$$Recall@K = \frac{\#\{\textit{(preferred items)} \cap \textit{(Top-K)}\}}{\textit{\# preferred items}} \quad (9)$$

- **MAP@K** Mean Average Precision is designed for considering measuring the order in predictions, while precision and recall are incompetent about it. It calculates the average precision (AP) among all users. For each user, we compute the AP by the precision and relevant value in top-N ranks. Here, P and Q represent the number of users items separately, and $relevant$ is a binary function that it gives out 1 if the k^{th} recommendation is the item that the user interests otherwise 0.

Model	CDs&Vinyl	BeerAdvocate
SVD	1.254	1.114
NEAR	1.182	1.019
R-NEAR	**0.849**	**0.842**

Table 3: Root-mean-square standard deviation (RMSSTD) comparison between R-NEAR and baselines on different domain datasets.

$$\begin{aligned} AP@K &= \frac{1}{Q}\sum_{k=1}^{K} precision(k) \cdot relevant(k) \\ MAP@K &= \frac{1}{P}\sum_{p=1}^{P} AP@N(p) \end{aligned} \quad (10)$$

- **R-Precision** R-Precision uses the same $relevant$ function above, and leverage the percentage of users' relevant items.

$$Precision@N = \frac{1}{M}\sum_{m=1}^{M} relevant(m) \quad (11)$$

- **NDCG@K** Normalised Discounted Cumulative Gain is a famous measurement of ranking quality. DCG measures the cumulative gain among K ranks, while IDCG leverages the cumulative gain on all relevant items. A greater NDCG@K value means recommender systems provide more precise ranks to users.

$$\begin{aligned} DCG@K &= \sum_{i=1}^{K} \frac{2^{relevant(i)} - 1}{\log_2(i+1)} \\ IDCG@k &= \sum_{i=1}^{REL} \frac{2^{relevant(i)} - 1}{\log_2(i+1)} \\ NDCG@k &= \frac{DCG@K}{IDCG@K} \end{aligned} \quad (12)$$

Recommendation Performance

To leverage the recommendation performance of R-NEAR, we compare it with nine state-of-the-art recommender systems. In this experiment, we make recommendations by top-K item rank, which ranks candidate items by the predicted ratings and selects K items. To be fair, we use the same evaluation parameters as in (Wu et al. 2019), where we set the K value to be 5, 10, and 20 for MAP, $Precision$, and $Recall$, and calculate the $NDCG$ by $NDCG@10$. Table 1 and Table 2 demonstrate the Top-k recommendation performance comparison between our method and assorted baselines on Amazon CDs&Vinyl and BeerAdovocate dataset separately. According to these results, we summarise the following key observations.

Firstly, NEAR, the base method our method variants on, shows excellent performance and exceed other baselines in

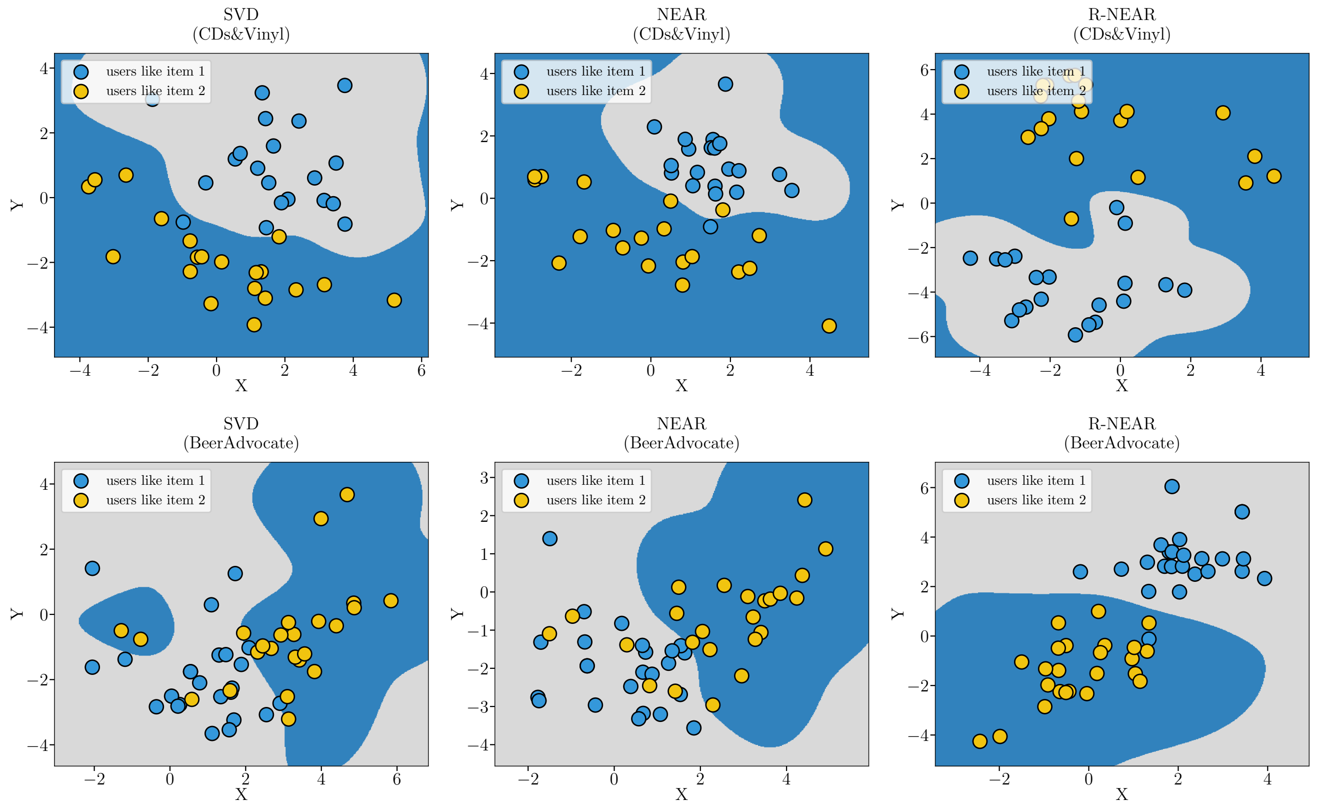

Figure 3: We demonstrate two group of users who like different items on CDs&Vinyl and BeerAdvocate datasets. Blue points means the users like item 1, while yellow points reflect the users like item 2. These figures reveal R-NEAR consistently outperform baselines, since it can clearly separate different groups of users.

majority cases, and demonstrates competitive $NDCG$ performance with CE-VNCF, the state-of-the-art recommendation algorithm. This result is not surprising because NEAR is good at filtering unhelpful factors and finding the most significant factor. Based on the critical factor, it generates perturbed embedding, which helps recommender systems understand users' tastes better than other baselines. Although the computation process of NEAR is complicated and expensive, these outcomes provide an exciting opportunity to advance our knowledge of using factor-based explanations to improve recommendations.

Secondly, our method outperforms NEAR in both two datasets in terms of all evaluation metrics. We think the reason is that NEAR only manipulates one user factor at a time, while R-NEAR considers all factors of both users and items. Additionally, we multiply the factor-based interpretations on corresponded embeddings, so that we can amplify important factors and suppressing unimportant factors. This noteworthy finding reveals the effectiveness and advance of our method.

Thirdly, comparing with other deep neural network-based algorithms, i.e., NCF, E-NCF, and CE-VNCF, our method consistently shows considerable improvements on all evaluation metrics. The reason we anticipate is that these methods use neural networks to learn other information, for example, key-phrase, while our neural network aims to explore more in-depth knowledge that can explain the attitude of users to items on factor level. Moreover, R-NEAR reforms recommendation behavior by the factor level explanation, so that it can attain advanced Top-K item ranking performance.

Latent Representation Analysis

We outlined previously that high-quality latent embedding is the critical aspect of a good recommendation. Therefore, we thoroughly appraise the quality of latent embedding in R-NEAR in this section. Correctly, we qualitatively evaluate the user embeddings by visualisation and observation and quantitatively assess the user latent representations by statistically evaluation metrics in the unsupervised learning field.

In this experiment, we visualise users as scatters in clusters. Two main concepts determine the quality of clusters: the compactness and the separation (Hassani and Seidl 2017). The compactness means how close the users who have the same interests in a cluster, while the separation reflects how to differentiate a user cluster are from other clusters. We first evaluate the compactness by the users who like the same items. Then we measure both coherency and sepa-

Model	CH	D	S	DB	XB	SD
SVD	0.06	0.12	0.32	1.20	23.92	1.36
NEAR	0.07	0.06	0.35	1.12	19.29	1.45
R-NEAR	**0.24**	**0.53**	**0.56**	**0.63**	**14.57**	**0.81**

Table 4: Internal cluster measurements on Amazon CDs&Vinyl dataset.

Model	CH	D	S	DB	XB	SD
SVD	0.02	0.04	0.18	1.57	45.47	1.61
NEAR	0.05	0.06	0.27	1.15	26.88	1.48
R-NEAR	**0.30**	**0.23**	**0.63**	**0.47**	**8.67**	**0.82**

Table 5: Internal cluster measurements on BeerAdvocate dataset.

ration by the users who like different items.

As Hassani *et al.*(Hassani and Seidl 2017) suggests, we employ the sum of Root-mean-square standard deviation (RMSSTD) to leverage the compactness in the first task. Then we report six internal clustering measures in the second task, i.e., Calinski Harabasz index (CH), Dunn's indices (D), Silhouette index (S), Davies-Bouldin index (DB), Xie-Beni index (XB) and SD validity index (SD).

Users with same interests In the first experiment, we randomly choose an item from the test set. Then we find out the users who are interested in this item, which means the users votes at least 3 ratings for the item. After that, we extract the latent embeddings of these users and employ PCA (Yu and Liu 2003) to reduce the dimensions.

We qualitatively analyse the embedding quality by observing the dimensionally-reduced embedding visualisation, where the area of regions in the outcome denotes the level of density. We compare our method with SVD and NEAR on both Amazon CDs&Vinyl and BeerAdvocate datasets, as shown in Figure 2. There are several important observations in this result. First, the SVD shows the most sparse cluster, while NEAR improves a little over SVD. As we explained before, it is because NEAR ameliorates one latent factor at each training time, leading a small development on the representation quality. In contrast, R-NEAR shows the densest cluster by intuitive observation, which reveals our method can learn users' preferences better than baselines.

We then quantitatively calibrate the compactness by the Root-mean-square standard deviation (RMSSTD). RMSSTD calculates the square root variance of attributes in each cluster, which are commonly employed to measure only the density of clusters (Hassani and Seidl 2017). The smaller value of RMSSTD, the better quality of latent embeddings. Table 3 demonstrates the evaluation results, which significantly reveal R-NEAR surpass baselines on both two datasets. In terms of compactness, this result provides strong statistical evidence to demonstrate that our method can learn personal embeddings with good quality.

Users with different interests In this section, we run the experiment of users who have different interests to evaluate both coherency and separation. Similar to the prior experiment, we first randomly choose two items from the test set. Then we extract the users who like the two items respectively. We also compare our method with SVD and NEAR in this experiment on the two datasets.

We demonstrate the qualitative analysing results in Figure 3. SVD shows the worst because users with different interests are not wholly separated, and users with the same interests are not aggregated. NEAR revise several outliers and performs slightly better than SVD. R-NEAR continuously outperforms baselines since it can isolate the users with distinct interests and group the users who are interests in the same item. This instinctive observation reveals that R-NERA can achieve both good compactness and separation for user clusters.

We then apply six clustering measurements to quantitatively appraise both compactness and separation on latent embeddings, as shown in Table 4 and Table 5. All these six evaluation methods consider how much the cluster centers are expanded and how close the scatters around their center simultaneously. Here, the larger value of CH, D, and S, the better quality of learned embeddings. In contrast, the smaller value of DB, XB, and SD, the more optimal performance. In these results, we can observe that R-NEAR persistently exceed other recommender systems, which substantiates the ability of our method to learn high-quality embedding and to achieve state-of-the-art recommendation performance.

Conclusion

In this paper, we proposed R-NEAR, a universal explanation method for any latent factorised based recommender systems and addressed both recommendation and interpretation problems. Besides, we introduce a new training fashion that applying explanation to reform recommendation quality. We compared our method with nine state-of-the-art recommendation methods on two real-world datasets from distinct domains. Experimental results revealed that R-NEAR continuously beat baselines and achieve state-of-the-art performance in the Top-K item ranking task. We thoroughly evaluated the embedding quality through both qualitative and quantitative analysis. The qualitative assessment demonstrated that our method attains good compactness and excellent separations for user clusters. The quantitative evaluation used six internal clustering measurements and proved that our method had learned personal embeddings with exceptional quality. These outcomes as a whole are convincing arguments for the extensive use of latent factorised explanation to improve recommendations. Overall, explaining is not enough, while joining explanations into recommender systems to reform recommendation performance is the matter. We hope this work provides a rich foundation for the extensions of using general explanation improving recommender systems, for example, natural language interpretation.

Acknowledgments

This work is supported by Science Foundation Ireland through the Insight Centre for Data Analytics under grant number SFI/12/RC/2289_P2

References

[Chen and Wang 2017] Chen, L., and Wang, F. 2017. Explaining recommendations based on feature sentiments in product reviews. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, 17–28. ACM.

[Costa et al. 2018] Costa, F.; Ouyang, S.; Dolog, P.; and Lawlor, A. 2018. Automatic generation of natural language explanations. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion*, 57. ACM.

[Frosst and Hinton 2017] Frosst, N., and Hinton, G. 2017. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*.

[Geng et al. 2007] Geng, X.; Liu, T.-Y.; Qin, T.; and Li, H. 2007. Feature selection for ranking. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 407–414. ACM.

[Guidotti et al. 2018] Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; and Pedreschi, D. 2018. A survey of methods for explaining black box models. *ACM Computing Surveys (CSUR)* 51(5):93.

[Hassani and Seidl 2017] Hassani, M., and Seidl, T. 2017. Using internal evaluation measures to validate the quality of diverse stream clustering algorithms. *Vietnam Journal of Computer Science* 4(3):171–183.

[He and McAuley 2016] He, R., and McAuley, J. 2016. 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*, 507–517. International World Wide Web Conferences Steering Committee.

[He et al. 2017] He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, 173–182. International World Wide Web Conferences Steering Committee.

[Koren 2008] Koren, Y. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 426–434. ACM.

[Lacic, Kowald, and Lex 2016] Lacic, E.; Kowald, D.; and Lex, E. 2016. High enough?: Explaining and predicting traveler satisfaction using airline reviews. In *Proceedings of the 27th ACM Conference on Hypertext and Social Media*, 249–254. ACM.

[Lai et al. 2013] Lai, H.-J.; Pan, Y.; Tang, Y.; and Yu, R. 2013. Fsmrank: Feature selection algorithm for learning to rank. *IEEE transactions on neural networks and learning systems* 24(6):940–952.

[LeCun, Bengio, and Hinton 2015] LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *nature* 521(7553):436–444.

[Lee and Seung 2001] Lee, D. D., and Seung, H. S. 2001. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, 556–562.

[Li et al. 2018] Li, J.; Cheng, K.; Wang, S.; Morstatter, F.; Trevino, R. P.; Tang, J.; and Liu, H. 2018. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)* 50(6):94.

[McAuley and Leskovec 2013] McAuley, J. J., and Leskovec, J. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*, 897–908. ACM.

[Ouyang and Lawlor 2019] Ouyang, S., and Lawlor, A. 2019. Near: A partner to explain any factorised recommender system. In *Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization*, 247–249. ACM.

[Ribeiro, Singh, and Guestrin 2016] Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–1144. ACM.

[Strang 1993] Strang, G. 1993. The fundamental theorem of linear algebra. *The American Mathematical Monthly* 100(9):848–855.

[Symeonidis, Nanopoulos, and Manolopoulos 2009] Symeonidis, P.; Nanopoulos, A.; and Manolopoulos, Y. 2009. Moviexplain: a recommender system with explanations. In *Proceedings of the third ACM conference on Recommender systems*, 317–320. ACM.

[Tintarev and Masthoff 2011] Tintarev, N., and Masthoff, J. 2011. Designing and evaluating explanations for recommender systems. In *Recommender systems handbook*. Springer. 479–510.

[Tintarev and Masthoff 2015] Tintarev, N., and Masthoff, J. 2015. Explaining recommendations: Design and evaluation. In *Recommender systems handbook*. Springer. 353–382.

[Vig, Sen, and Riedl 2009] Vig, J.; Sen, S.; and Riedl, J. 2009. Tagsplanations: explaining recommendations using tags. In *Proceedings of the 14th international conference on Intelligent user interfaces*, 47–56. ACM.

[Wang et al. 2018] Wang, L.; Zhang, W.; He, X.; and Zha, H. 2018. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2447–2456. ACM.

[Wu et al. 2019] Wu, G.; Luo, K.; Sanner, S.; and Soh, H. 2019. Deep language-based critiquing for recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*, 137–145. ACM.

[Yu and Liu 2003] Yu, L., and Liu, H. 2003. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, 856–863.