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 a agnostic prediction. Geng et al. (Geng et al. 2007) addressed redun-
In this paper, we develop the current latent explanation generation method to explain recommendation behavior through latent factor-based explanations. As such, this research sheds new light on reforming recommender systems for advanced 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. In-
In this paper, we aim to provide exhaustive interpretation on both the user and the item side. We introduce the user explainable weights $W^U_{ij} \in \mathbb{R}^{1 \times N_i}$ and item explainable weights $W^V_{ij} \in \mathbb{R}^{1 \times N_j}$. 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 $W^U_{ij}$ or $W^V_{ij}$, the more indispensable the related latent factor is, or vice versa. We formulate the calculation procedure of user explainable weights $W^U_{ij}$ and item explainable weights $W^V_{ij}$ in the following equation.

$$W^U_{ij}, W^V_{ij} = f_{R-NEAR}(U_i, V_j) \quad (2)$$

**Generate Explanation**

We spread out the discussion of technical details of $f_{R-NEAR}$ in this section. In order to model $W^U_{ij}$ and $W^V_{ij}$, 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, $[\cdot]$ denotes the concatenation operation, $W_h \in \mathbb{R}^{2N \times h}$ is the hidden weights, $h$ is the number of hidden units, and $b_h$ is the bias.

$$H_{(i,j)} = \text{ReLU}(W_h[U_i, V_j] + b_h) \quad (3)$$

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 $W^U_{ij}$ and item explainable weights $W^V_{ij}$. Therefore, we apply the predicted rating $\hat{r}_{ij}$ as the target of the multi-layer neural networks. We stack a linear transformation layer on the hidden layer, due to the linear explanation filed, to leverage the differentiation between recommendation and ground truth. We show the loss function in Equation (4), where $o_{(i,j)}$ is the estimation of recommended rating, $W_{UV} \in \mathbb{R}^{1 \times 2N}$ is the explainable weights. We then separate explainable weights $W_{UV}$ into user explainable weights $W^U_{ij}$ and item explainable weights $W^V_{ij}$ by the embedding size $N$.

$$o_{(i,j)} = W_{UV}H_{(i,j)} + b_o$$

$$W^U_{ij}, W^V_{ij} = W_{UV} : W_{N \times N} \quad (4)$$

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.

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

**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 $W^U_{ij}$ and item explainable weights $W^V_{ij}$. 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 $W^U_{ij}$ and $W^V_{ij}$ 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 $r'_{ij}$ is the enhanced predicted rating. By doing this, our algorithm works as an extension to other recommender systems, which ensure strong scalability and robustness.

$$U'_i = U_i \odot W^U_{ij}$$

$$V'_j = V_j \odot W^V_{ij}$$

$$r'_{ij} = U'_iV'_j \quad (6)$$

In terms of learning the improvements for recommendation, 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.

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

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

<table>
<thead>
<tr>
<th>Model</th>
<th>R-Precision</th>
<th>NDCG</th>
<th>MAP@5</th>
<th>MAP@10</th>
<th>MAP@20</th>
<th>Precision@5</th>
<th>Precision@10</th>
<th>Precision@20</th>
<th>Recall@5</th>
<th>Recall@10</th>
<th>Recall@20</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCF</td>
<td>0.0378</td>
<td>0.0824</td>
<td>0.0435</td>
<td>0.0394</td>
<td>0.0342</td>
<td>0.0396</td>
<td>0.0332</td>
<td>0.0266</td>
<td>0.0539</td>
<td>0.0907</td>
<td>0.1428</td>
</tr>
<tr>
<td>E-NCF</td>
<td>0.0392</td>
<td>0.0851</td>
<td>0.0449</td>
<td>0.0411</td>
<td>0.0356</td>
<td>0.0418</td>
<td>0.0349</td>
<td>0.0271</td>
<td>0.0579</td>
<td>0.0956</td>
<td>0.1456</td>
</tr>
<tr>
<td>CE-NCF</td>
<td>0.0401</td>
<td>0.0853</td>
<td>0.0458</td>
<td>0.0412</td>
<td>0.0354</td>
<td>0.0404</td>
<td>0.0344</td>
<td>0.0270</td>
<td>0.0576</td>
<td>0.0942</td>
<td>0.1447</td>
</tr>
<tr>
<td>VNCF</td>
<td>0.0372</td>
<td>0.0809</td>
<td>0.0408</td>
<td>0.0384</td>
<td>0.0341</td>
<td>0.0393</td>
<td>0.0337</td>
<td>0.0275</td>
<td>0.0526</td>
<td>0.0869</td>
<td>0.1415</td>
</tr>
<tr>
<td>E-VNCF</td>
<td>0.0376</td>
<td>0.0833</td>
<td>0.0436</td>
<td>0.0399</td>
<td>0.0351</td>
<td>0.0395</td>
<td>0.0344</td>
<td>0.0281</td>
<td>0.0511</td>
<td>0.0880</td>
<td>0.1446</td>
</tr>
<tr>
<td>CE-VNCF</td>
<td>0.0374</td>
<td>0.0827</td>
<td>0.0425</td>
<td>0.0398</td>
<td>0.0351</td>
<td>0.0398</td>
<td>0.0340</td>
<td>0.0281</td>
<td>0.0516</td>
<td>0.0904</td>
<td>0.1440</td>
</tr>
<tr>
<td>SVD</td>
<td>0.0275</td>
<td>0.0751</td>
<td>0.0564</td>
<td>0.0507</td>
<td>0.0436</td>
<td>0.0503</td>
<td>0.0418</td>
<td>0.0332</td>
<td>0.0563</td>
<td>0.0917</td>
<td>0.1421</td>
</tr>
<tr>
<td>SVD++</td>
<td>0.0354</td>
<td>0.0763</td>
<td>0.0584</td>
<td>0.0509</td>
<td>0.0458</td>
<td>0.0509</td>
<td>0.0402</td>
<td>0.0335</td>
<td>0.0582</td>
<td>0.0938</td>
<td>0.1568</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Model</th>
<th>R-Precision</th>
<th>NDCG</th>
<th>MAP@5</th>
<th>MAP@10</th>
<th>MAP@20</th>
<th>Precision@5</th>
<th>Precision@10</th>
<th>Precision@20</th>
<th>Recall@5</th>
<th>Recall@10</th>
<th>Recall@20</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCF</td>
<td>0.0603</td>
<td>0.1056</td>
<td>0.0808</td>
<td>0.0750</td>
<td>0.0679</td>
<td>0.0747</td>
<td>0.0661</td>
<td>0.0571</td>
<td>0.0530</td>
<td>0.0930</td>
<td>0.1534</td>
</tr>
<tr>
<td>E-NCF</td>
<td>0.0577</td>
<td>0.1025</td>
<td>0.0762</td>
<td>0.0715</td>
<td>0.0653</td>
<td>0.0718</td>
<td>0.0642</td>
<td>0.0561</td>
<td>0.0520</td>
<td>0.0907</td>
<td>0.1494</td>
</tr>
<tr>
<td>CE-NCF</td>
<td>0.0678</td>
<td>0.1259</td>
<td>0.0934</td>
<td>0.0881</td>
<td>0.0815</td>
<td>0.0878</td>
<td>0.0800</td>
<td>0.0709</td>
<td>0.0605</td>
<td>0.1086</td>
<td>0.1872</td>
</tr>
<tr>
<td>VNCF</td>
<td>0.0677</td>
<td>0.1248</td>
<td>0.0928</td>
<td>0.0879</td>
<td>0.0811</td>
<td>0.0874</td>
<td>0.0802</td>
<td>0.0704</td>
<td>0.0598</td>
<td>0.1080</td>
<td>0.1861</td>
</tr>
<tr>
<td>E-VNCF</td>
<td>0.0678</td>
<td>0.1259</td>
<td>0.0934</td>
<td>0.0881</td>
<td>0.0815</td>
<td>0.0878</td>
<td>0.0800</td>
<td>0.0708</td>
<td>0.0604</td>
<td>0.1086</td>
<td>0.1872</td>
</tr>
<tr>
<td>CE-VNCF</td>
<td>0.0716</td>
<td>0.1301</td>
<td>0.0920</td>
<td>0.0892</td>
<td>0.0834</td>
<td>0.0897</td>
<td>0.0839</td>
<td>0.0735</td>
<td>0.0628</td>
<td>0.1166</td>
<td>0.1974</td>
</tr>
<tr>
<td>SVD</td>
<td>0.0613</td>
<td>0.1204</td>
<td>0.1410</td>
<td>0.1299</td>
<td>0.1163</td>
<td>0.1290</td>
<td>0.1130</td>
<td>0.0958</td>
<td>0.0784</td>
<td>0.1521</td>
<td>0.2123</td>
</tr>
<tr>
<td>SVD++</td>
<td>0.0622</td>
<td>0.1236</td>
<td>0.1423</td>
<td>0.1237</td>
<td>0.1072</td>
<td>0.1320</td>
<td>0.1089</td>
<td>0.0986</td>
<td>0.0797</td>
<td>0.1563</td>
<td>0.2138</td>
</tr>
</tbody>
</table>

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\footnote{http://jmcauley.ucsd.edu/data/amazon} and BeerAdvocate\footnote{[He and McAuley 2016]}\footnote{[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.

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

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

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

- **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.

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

- **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.

  $$MAP@K = \frac{1}{Q} \sum_{k=1}^{K} precision(k) \cdot relevant(k) \quad (10)$$

- **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.

  $$DCG@K = \sum_{i=1}^{K} \frac{2^{\text{relevant}(i)} - 1}{\log_2(i + 1)}$$

  $$IDCG@k = \sum_{i=1}^{REL} \frac{2^{\text{relevant}(i)} - 1}{\log_2(i + 1)}$$

  $$NDCG@k = \frac{DCG@K}{IDCG@K} \quad (12)$$

### 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 BeerAdvocate 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

<table>
<thead>
<tr>
<th>Model</th>
<th>CDs&amp;Vinyl</th>
<th>BeerAdvocate</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD</td>
<td>1.254</td>
<td>1.114</td>
</tr>
<tr>
<td>NEAR</td>
<td>1.182</td>
<td>1.019</td>
</tr>
<tr>
<td>R-NEAR</td>
<td><strong>0.849</strong></td>
<td><strong>0.842</strong></td>
</tr>
</tbody>
</table>

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


