Explainable Recommender Systems via Resolving Learning Representations

Ninghao Liu
Texas A&M University, TX, USA
nhliu43@tamu.edu

Yong Ge
University of Arizona, AZ, USA
yongge@email.arizona.edu

Li Li
Samsung Research America, CA, USA
li.li1@samsung.com

Xia Hu
Texas A&M University, TX, USA
hu@cse.tamu.edu

Rui Chen
Samsung Research America, CA, USA
rui.chen1@samsung.com

Soo-Hyun Choi
Samsung Electronics, South Korea
sh9.choi@samsung.com

ABSTRACT

Recommender systems play a fundamental role in web applications in filtering massive information and matching user interests. While many efforts have been devoted to developing more effective models in various scenarios, the exploration on the explainability of recommender systems is running behind. Explanations could help improve user experience and discover system defects. In this paper, after formally introducing the elements that are related to model explainability, we propose a novel explainable recommendation model through improving the transparency of the representation learning process. Specifically, to overcome the representation entangling problem in traditional models, we revise traditional graph convolution to discriminate information from different layers. Also, each representation vector is factorized into several segments, where each segment relates to one semantic aspect in data. Different from previous work, in our model, factor discovery and representation learning are simultaneously conducted, and we are able to handle extra attribute information and knowledge. In this way, the proposed model can learn interpretable and meaningful representations for users and items. Unlike traditional methods that need to make a trade-off between explainability and effectiveness, the performance of our proposed explainable model is not negatively affected after considering explainability. Finally, comprehensive experiments are conducted to validate the performance of our model as well as explanation faithfulness.

KEYWORDS

Explainable Artificial Intelligence; Recommender Systems

ACM Reference Format:
Ninghao Liu, Yong Ge, Li Li, Xia Hu, Rui Chen, and Soo-Hyun Choi. 2020. Explainable Recommender Systems via Resolving Learning Representations. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411919

1 INTRODUCTION

Recommender systems play a pivotal role in a wide range of web applications and services, in terms of distributing online content to targets who are likely to be interested in it. The majority of the efforts in the domain have been put into developing more effective models to achieve better performance. In contrast, the progress of analyzing the explainability aspect of recommender systems is running behind. Explainable recommender systems address *the problem of why* – besides providing recommendation results, they also give reasons to clarify why such results are derived [11, 31, 47, 55]. Explainability may benefit a recommender system in several aspects. First, explanations help system maintainers to diagnose and refine the recommendation pipeline (system-oriented). Second, by increasing transparency, explanations promote persuasiveness and customer satisfaction of recommender systems (user-oriented).

Recently, Explainable AI (XAI), or interpretable machine learning, is receiving increasing attention [5, 11, 28] (we use "explanation" and "interpretation" interchangeably in this paper). One popular direction is post-hoc methods [5, 31], where some examples include understanding the meaning of latent factors [10, 29] or reconstructing the rationale behind each prediction [47]. However, post-hoc interpretation suffers from the issue of explanation accuracy [11, 32]. Another direction is to build explainable models or add explainable components, where attention mechanism is commonly applied to highlight important features for prediction [8, 34]. Nevertheless, explanation schemes such as heatmaps in images or texts are not structured and still require subjective comprehension of users. Since effective models can learn informative latent representations from data [2, 17, 33], one of the important building blocks of model explainability is the transparency of representation learning. In traditional factorization models, input features can be directly associated with latent factors to elucidate their meanings [56], but this explanation scheme is not directly applicable to recent embedding learning frameworks.

In this work, we propose an explainable recommendation model through promoting the transparency of latent representations. To begin with, we summarize three elements that help make a model more interpretable. We name them as IOM elements since they involve certain requirements on the **I**nput data format, **O**utput attribution, and **M**iddle-level representations. The proposed explainable model is designed based on the IOM elements. Users, items and attribute entities are processed as nodes in a graph. Furthermore, the efforts on interpretable recommendation are split into three parts: (1) we disentangle the *interactions* between latent representations in different layers; (2) multiple semantic *factors* are identified automatically from data; (3) latent dimensions are divided into *segments* according to their information source (i.e., node types) and affiliated factors. The first part is achieved via graph convolutional networks (GCNs). However, different from previous work [13, 42, 44], we propose to understand GCNs from another perspective by comparing its working mechanism with those of fully-connected networks [3] and capsule networks [33]. The second and third aspects are achieved through a novel architecture design, where different dimensions of latent representations focus on different aspects of data. Different from previous work [6, 26, 39], factor discovery and representation learning are jointly conducted. In this way, we are able to depict how information flows from input features through these latent states to prediction results. Different from some existing work that sacrifices effectiveness for interpretability, the proposed model still achieves good performance in the experimental evaluation. Finally, besides visualizing explanations, we also quantitatively measure explanation accuracy.

Our contributions in this work are summarized as follows:

- We propose an explainable recommendation model that improves transparency of latent representations. Specifically, we propose to unravel the interactions between representations, and segment the latent dimensions into different factors. The consideration of interpretability does not negatively influence model effectiveness.
- We summarize and refine the elements that improve the interpretability of recommendation. These elements involve certain requirements on the input data format, output attribution, and middle-level representations, and are thus named as IOM elements. The proposed model is designed based on IOM elements.
- We conduct comprehensive experiments to evaluate the proposed model in two aspects, effectiveness and interpretability. We show that the effectiveness of the model will not be hindered by interpretability. Also, we quantitatively measure the accuracy of interpretation through adversarial attack.

2 PRELIMINARIES: ELEMENTS OF EXPLAINABLE RECOMMENDATION

In this section, we discuss the important elements in a recommendation pipeline that could promote human understanding. The proposed model, which is to be formally introduced in the next section, considers all of these three elements. It is worth noting that some directions of XAI may not focus on these elements, and they are beyond the discussion of this paper.

2.1 Conceptualization of Input

Besides complexity and parameter sizes, deep models are commonly regarded as black-boxes because they operate on low-level features, rather than high-level, discrete, human-comprehensible concepts. For examples, computer vision models operate on pixels, and natural language models take word vectors as input. The direct consequence is that there is a lack of concrete carriers to concisely depict information flows from input to output. Some recent research starts to explore how to conceptualize data into discrete interpretation basis, and derive explanations or explainable models upon it [19, 49, 52, 57].

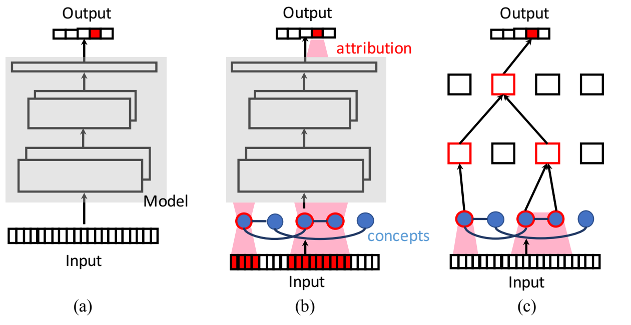

Figure 1: Models with different degrees of transparency. (a): A black-box model. (b): A model with conceptualized input and attributable output. (c): A model that extends (b) with transparent representation learning.

In this paper, we also work on discretized input data. In recommender systems, users and items can naturally be regarded as discrete concepts. For side content information, we make use of knowledge graphs (KGs) where each entity in a KG can be regarded as a concept. Developing new techniques to extract concepts from raw input, or building more informative graphs, is not a main contribution of this paper. Finally, we connect users and items with observed interactions, items and entities if they are related, entities and entities if they are already connected in a KG. We ignore the link types in a KG. Therefore, users, items and KG entities are built into a graph used as input into recommender systems.

2.2 Attributable Output

A basic requirement of model interpretability is the ability to obtain the attribution of features to a prediction output (i.e., output is attributable to certain features). Depending on whether the attribution is obtained after or along with the prediction, interpretation techniques can be categorized into post-hoc methods [31, 35, 36, 47] and building intrinsically explainable models [38]. Post-hoc methods usually approximate the original model with simpler but explainable ones, or resort to backpropagation to estimate the model's sensitivity to different features. Post-hoc methods have the advantage of being model-agnostic [31], but they have also been criticised as not being faithful enough in reproducing the working mechanism of the original model [32]. In this work, our goal is to design an *explainable model*, instead of developing new post-hoc methods, to provide human comprehensible rationales for recommendation.

2.3 Resolved Middle-Level Representations

The resultant model, after considering the two elements above, can be roughly depicted in Figure 1(b). It is better than the fully black-box model in Figure 1(a), but still does not shed light on the model's internal mechanism.

The main difficulty for better interpreting complex models lies in disentangling and understanding the interactions between latent representations. An example is shown in Figure 2 where we build a recommendation model based on a Multi-Layer Perceptron (MLP) with three hidden layers with L_1 regularization on the LastFM dataset [42]. The feature importance explanation of each prediction is computed using the denoised gradient-based method [36]. We

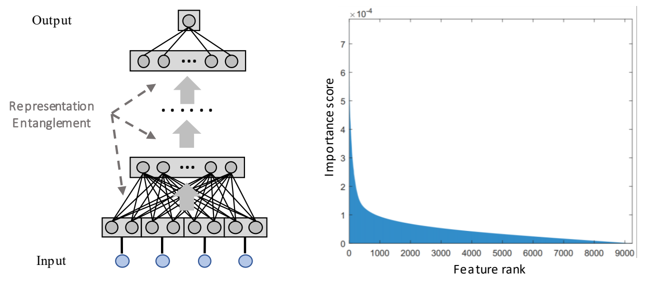

Figure 2: Feature interaction entangling in MLP.

then normalize the importance scores to the unit sum, and sort the features according to their scores. We randomly select 500 samples from test data and compute the average of the sorted scores, as shown in the right part of Figure 2. It is observed that, although a small number of features are important for each prediction, the effect (area under the curve) of the long tail of less important features together is still significant. The reason for this phenomenon is that a fully-connected network tends to entangle features without explicitly separating the interactions of latent dimensions [37]. Without tackling this challenge, the efforts of conceptualizing input and attributing output would be largely thwarted, since contributions of different features will be indistinguishable.

2.3.1 Disentanglement of Representation Interactions. From the analysis above, we are facing the dilemma where we hope to both: (1) explore *flexible* feature interactions, and (2) constrain feature interactions to be *concise* to trace. These requirements remind us of the recent Capsule Network (CapsNet) [33]. Each "capsule" represents a certain concept, and a dynamic routing process is run to decide the contribution between capsules of adjacent layers. Let $\mathbf{z}_j$ and $\mathbf{z}_i$ denote the embeddings of a higher-level capsule v_j and a lower-layer capsule v_i. The capsules interact in a way that

$$\mathbf{z}_j = \text{squash}(\sum_{i=1}^{I} a_{i,j}\mathbf{z}_{j|i}), \quad \mathbf{z}_{j|i} = \mathbf{W}_{i,j}\mathbf{z}_i, \tag{1}$$

where $\mathbf{W}_{i,j}$ is the bilinear mapping matrix, "squash" is a non-linear function, and $a_{i,j} \geq 0$ is the contribution score tracking the relation between capsules. Some recent work [24, 33, 48, 53] claims to use CapsNet to improve model interpretability. Nevertheless, several challenges impede us from directly applying CapsNet to certain recommendation scenarios. First, if we treat users or items as capsules, then the number of parameters $\{\mathbf{W}_{i,j}\}$ would be too large. Second, dynamic routing significantly increases time complexity and memory. Third, how to instill structured knowledge from human experts into the model remains unsolved.

Next we will show that graph convolution processes information in a similar manner to CapsNet, and the challenges faced by CapsNet can be easily tackled by GCN. Let $\mathbf{z}_j$ be the embedding of node v_j. Through graph convolution, its embedding is updated as:

$$\mathbf{z}_j \leftarrow \sigma(\sum_{v_i \in \mathcal{N}_j \cup \{v_j\}} a_{i,j}\mathbf{W}\mathbf{z}_i), \tag{2}$$

where $\mathcal{N}_j$ denotes the neighbors of v_j, $\sigma(\cdot)$ is the activation function, $a_{i,j}$ is the attention score. By comparing Eq(1) and Eq(2), it can be observed that both models use vectors to represent objects, and apply the weighted vector sum to propagate information. In this sense, GCN may be seen as a special case of CapsNet, where iterative routing is omitted and bilinear mapping reduces to a single matrix $\mathbf{W}$ that reduces the number of parameters. Moreover, GCN can naturally handle relational data and human knowledge organized in the form of graphs. In this paper, we use graph convolution as the basic operation to build explainable recommender systems.

2.3.2 Segmentation of Latent Dimensions. Another aspect of representation learning transparency is to explicitly separate the information captured by different dimensions of representations. There are two advantages for latent dimension separation. First, each set of dimensions corresponds to one semantic aspect, so we are able to learn meaningful representations. Second, disentanglement of latent dimensions may improve model effectiveness via encoding more nuanced relations in data [6]. For example, if we use an embedding to depict the interest of a user, then ideally we would want different dimensions to separately capture user interests in different item genres, such as clothes, electronics, and daily necessities.

Disentangling latent dimensions is a nontrivial task despite some preliminary work proposed recently. [6, 26] extend network embedding by learning several vectors for each node. Wang *et al.* [39] utilize item categories as part of the supervised information to associate latent dimensions with different aspects. However, how to deal with extra knowledge or automatically discover factors without category information are beyond their discussion.

3 PROBLEM FORMULATION

Notations. In this paper, column vectors are represented by boldface lowercase letters, such as $\mathbf{a}$ and $\mathbf{b}$; matrices are represented by boldface capital letters, such as $\mathbf{W}$ and $\mathbf{Y}$; sets are represented by calligraphic capital letters, such as $\mathcal{U}$ and $\mathcal{N}$. The i-th entry of a vector $\mathbf{b}$ is denoted by b_i, and entry (i, j) of a matrix $\mathbf{W}$ is denoted by $W_{i,j}$. As discussed above, the input into recommender systems is constructed as a graph $\mathcal{G}$ with nodes denoted by $\mathcal{V}$. We include three types of nodes in the graph, i.e., users, items and knowledge entities, where $\mathcal{U}$, $\mathcal{T}$ and $\mathcal{E}$ denote the user set, item set and entity set, respectively. Here note that $v \in \mathcal{V} = \mathcal{U} \cup \mathcal{T} \cup \mathcal{E}$ can be used to denote any type of nodes when the node type is not specified. In this paper, we consider three types of links in our scenario, user-item links $\{(u, t)\}$, item-entity links $\{(t, e)\}$ and entity-entity links $\{(e, e')\}$, where $u \in \mathcal{U}$, $t \in \mathcal{T}$ and $e, e' \in \mathcal{E}$. A user-item link (u, t) indicates an observed interaction (e.g., purchase, impression) between user u and item t. An item-entity link (t, e) unveils the item's quality or attribute described by entity e. An entity-entity link (e, e') simply stores the relation between the two entities. Without loss of generality, it is also possible to add user-entity links $\{(u, e)\}$ to reflect a user's attention or attitude towards an entity, but such information is not directly available in our experimental data. There is also a ground-truth interaction matrix $\mathbf{Y} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{T}|}$, where $Y_{u,t} = 1$ means that there is an observed interaction between user u and item t, and $Y_{u,t} = 0$ otherwise.

Problem Definition. Given the input graph $\mathcal{G}$, our goal is to predict whether user u is interested in an item t. That is, we aim to learn a prediction function $\hat{Y}_{u,t} = f(u, t|\mathcal{G}, \Theta)$, where $\hat{Y}_{u,t}$ is the estimated probability that user u is interested in item t, and Θ denotes

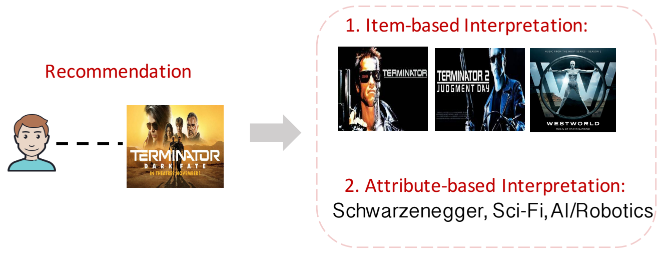
Recommendation
1. Item-based Interpretation:
2. Attribute-based Interpretation:
Schwarzenegger, Sci-Fi, AI/Robotics

Figure 3: Interpretation schema. A user's interest is explained through the historical items he has interacted with, as well as knowledge entities.

the parameters to learn. More importantly, additional constraints for explainability are added to the architecture of f.

4 INTERPRETABLE RECOMMENDATION WITH GRAPH DATA

In this section, we introduce the proposed explainable recommender system, which is based on the IOM elements introduced in previous sections. First, we extend the traditional explainable factor decomposition model into a more general representation learning framework. Second, we introduce the recommendation interpretation schema. Third, we introduce a novel graph convolution architecture to resolve latent dimensions, as well as the variational autoencoder based learning framework.

4.1 Node-Factor Factorization

In each recommendation application, we assume that there exist a number of latent factors which compactly describe different aspects of the nodes in the graph. Suppose there are C factors. In traditional factorization models [23], we consider the parameterization where a nonnegative score $F_{v,c}$ is assigned for each pair of node $v \in \mathcal{V}$ and factor $c \in \{1, ..., C\}$ to measure their affiliation strength. More specifically, for a user node $u \in \mathcal{U}$, $F_{u,c}$ represents the user's attention or interest on factor c. For an item node $t \in \mathcal{T}$, $F_{t,c}$ quantifies the degree that factor c can be used to describe item t. Therefore, the tendency that factor c independently triggers the interaction between u and t could be computed as $F_{u,c} \cdot F_{t,c}$. Then, an observed link $Y_{u,t}$ can be decomposed as $Y_{u,t} \approx \sum_{c=1}^{C} F_{u,c} \cdot F_{t,c}$. The explanation behind is that user u is more likely to interact with item t if the item has high quality or reputation on more factors which are also highly valued by the user.

Traditionally, $F_{u,c}$ and $F_{t,c}$ are directly solved as nonnegative parameters optimized through matrix factorization [22, 23, 56]. However, the design of explicit factors is not fully compatible with recent representation learning frameworks. To combine the explainability of explicit factorization and the flexibility of recent learning frameworks, we embed different factors within nodes separately into the latent space. That is, we use a vector $\mathbf{z}_v^c \in \mathbb{R}^D$ to represent node v described by factor c, where D is the embedding dimension of a single factor. The score between user u and item t under factor c is thus computed as $\langle \mathbf{z}_u^c, \mathbf{z}_t^c \rangle$, where $\langle \cdot, \cdot \rangle$ denotes inner product. And the link factorization is thus formulated as $Y_{u,t} \approx \sum_{c=1}^{C} \langle \mathbf{z}_u^c, \mathbf{z}_t^c \rangle$. The link prediction can also be written as $Y_{u,t} \approx \langle \mathbf{z}_u, \mathbf{z}_t \rangle$, where $\mathbf{z}_v = [\mathbf{z}_v^1; \mathbf{z}_v^2; ...; \mathbf{z}_v^C] \in \mathbb{R}^{C \times D}$. For each observed user-item pair, it is expected that only a part of the factors will be activated in preserving similarity. If factor c is the potential cause of making user u interact with item t, in order for $\langle \mathbf{z}_u^c, \mathbf{z}_t^c \rangle$ to have a large score, not only should the angle be small between $\mathbf{z}_u^c$ and $\mathbf{z}_t^c$, we also expect them to have large lengths.

In certain models, more flexible interactions between factorized representations are designed. For example, in Factorization Machines [30], cross-factor interactions are allowed (i.e., $\langle \mathbf{z}_u^{c_1}, \mathbf{z}_t^{c_2} \rangle$ also contributes to $Y_{u,t}$ and $c_1 \neq c_2$). In addition, if we assume that users have multiple interests but items are constrained with a single property, then $Y_{u,t} \approx \sum_{c=1}^{C} \langle \mathbf{z}_u^c, \mathbf{z}_t \rangle$.

4.2 Interpretation Hierarchy

In this work, we treat user nodes, item nodes and entity nodes as *concepts*. Different concepts are assigned to different levels. A concept on level ℓ is represented by concepts on level ℓ' where $\ell' < \ell$. Specifically, let v and v' denote concepts on level ℓ and ℓ', then we could interpret the embedding of v as $\mathbf{z}_v \approx \sum_{v'} p_{v,v'} \mathbf{z}_{v'}$, where $p_{v,v'}$ denotes the contribution from v' to v. Low-level concepts tend to be simple, while high-level concepts are complex and composite. Lower-level concepts can be regarded as the *basis* to represent higher-level ones. Note that besides $\ell' = \ell - 1$, any $\ell' < \ell$ can be chosen as the basis level to represent concepts on level ℓ.

The levels of concepts are defined in an iterative manner. The user nodes are on the top level ℓ^{max}. The item nodes are on level $\ell^{max} - 1$ as they connect to users. The entities which connect to items are on level $\ell^{max} - 2$. In general, if a node v connects to another node on level ℓ and v does not belong to a level higher than ℓ, then it belongs to level $\ell - 1$. In the end, the bottom entity nodes are on level 0. In this work, to simplify illustration, we assume that there is only one level of entity nodes, so that $\ell^{max} = 2$ for user nodes. An example of recommendation explanation for a user is given in Figure 3. The proposed model can be easily extended to deal with multi-level knowledge entities.

4.3 Resolving Information Flow

We now introduce how to segment $\mathbf{z}_v$ to different factors. Under traditional GCNs shown in Figure 4(a) and Eq(2), the embeddings of user u and item t on the final GCN layer are computed as:

$$\begin{aligned} \mathbf{z}_u &\leftarrow \mathtt{COMBINE}(\mathbf{z}_u, \mathtt{AGGREGATE}(\{\mathbf{z}_t : t \in \mathcal{N}_u \subseteq \mathcal{T}\})), \\ \mathbf{z}_t &\leftarrow \mathtt{COMBINE}(\mathbf{z}_t, \mathtt{AGGREGATE}(\{\mathbf{z}_e : e \in \mathcal{N}_t \subseteq \mathcal{E}\})). \end{aligned} \tag{3}$$

It is worth noting that we restrict the information flow direction. An item embedding $\mathbf{z}_t$ receives information from knowledge entities, as well as its own embedding on the lower GCN layer. Similarly, a user embedding $\mathbf{z}_u$ receives information from itself on the lower layer, the items that have been interacted, and knowledge entities as second order connections. Finally, top-layer $\mathbf{z}_u$ and $\mathbf{z}_t$ engage in computing recommendation prediction $\hat{Y}_{u,t}$. Common COMBINE() operations include summation or concatenation followed by transformation, while common AGGREGATE() includes summation and mean operations [12, 44, 50]. In the proposed model, we modify traditional COMBINE() and AGGREGATE() operations to resolve information propagation between representations.

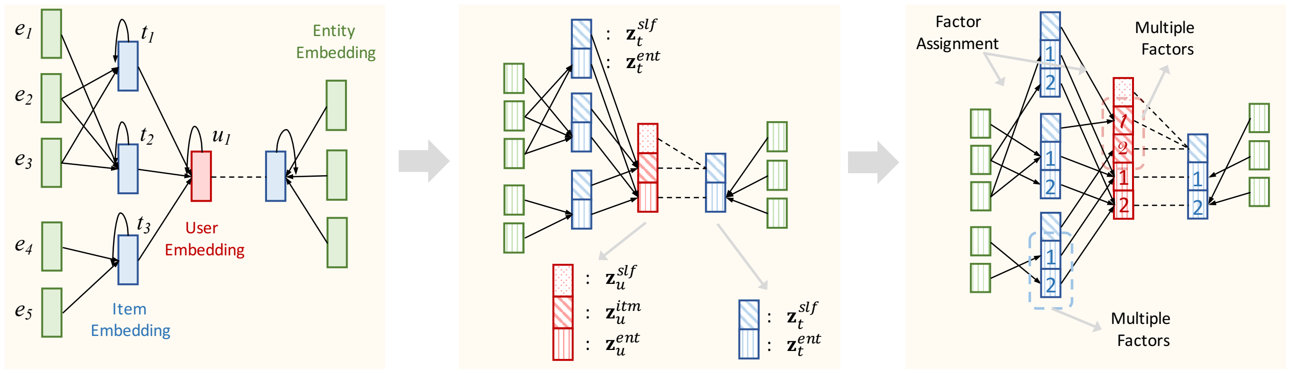

Figure 4: Overview of the proposed model design procedure, best viewed with color. (a): Traditional GCN information propagation. (b): GCN after discriminating information sources. (c): GCN after resolving information sources and factors.

4.3.1 Modification to COMBINE(). Instead of merging information from lower-level representations, we keep $\mathbf{z}_v$ and $\texttt{AGGREGATE}(\{\mathbf{z}_{v'} : v' \in \mathcal{N}_v\})$ separate if v and v' are different types of nodes. Specifically, as shown in Figure 4(b), we define item embedding as:

$$\mathbf{z}_t = \mathbf{z}_t^{slf} \parallel \mathbf{z}_t^{ent}, \quad \text{where} \quad \mathbf{z}_t^{ent} = \texttt{AGGREGATE}(\{\mathbf{z}_e^{slf} : e \in \mathcal{N}_t \subseteq \mathcal{E}\}). \tag{4}$$

Here $\mathbf{z}_t^{slf}$ denotes the item itself's bottom-level embedding, and $\parallel$ means concatenation. Similarly, for each user embedding $\mathbf{z}_u$, it receives information from both items and knowledge entities that describe those items, so we define it as:

$$\begin{aligned} \mathbf{z}_u &= \mathbf{z}_u^{slf} \parallel \mathbf{z}_u^{itm} \parallel \mathbf{z}_u^{ent}, \\ \text{where} \quad \mathbf{z}_u^{itm} &= \texttt{AGGREGATE}(\{\mathbf{z}_t^{slf} : t \in \mathcal{N}_u \subseteq \mathcal{T}\}), \\ \mathbf{z}_u^{ent} &= \texttt{AGGREGATE}(\{\mathbf{z}_t^{ent} : t \in \mathcal{N}_u \subseteq \mathcal{T}\}). \end{aligned} \tag{5}$$

Here $\mathbf{z}_u^{itm}$ describes the user through the historical items he has interacted with. $\mathbf{z}_u^{ent}$ describes the user with knowledge entities.

4.3.2 Modification to AGGREGATE(). Furthermore, instead of directly aggregating information from items or entities without distinguishing their natures or semantics, we first assign low-level embeddings into different factors, and then send information accordingly. Specifically, as shown in Figure 4(c), we define item-side aggregation as:

$$\mathbf{z}_t^{ent} = \big\Vert_{c=1}^{C_1} \mathbf{z}_t^{ent,c}, \quad \text{where} \quad \mathbf{z}_t^{ent,c} = g\Big(\sum_{e \in \mathcal{N}_t} \frac{p(e,c)}{|\mathcal{N}_t|} \mathbf{z}_e^{slf}\Big). \tag{6}$$

Here $p(e,c) \in [0,1]$ denotes the affiliation degree between entity e and factor c. Also, we define user-side aggregation as:

$$\mathbf{z}_u^{itm} = \big\Vert_{c=1}^{C_2} \mathbf{z}_u^{itm,c}, \quad \text{where} \quad \mathbf{z}_u^{itm,c} = g\Big(\sum_{t \in \mathcal{N}_u} \frac{p(t,c)}{|\mathcal{N}_u|} \mathbf{z}_t^{slf}\Big), \tag{7}$$

$$\mathbf{z}_u^{ent} = \big\Vert_{c=1}^{C_1} \mathbf{z}_u^{ent,c}, \quad \text{where} \quad \mathbf{z}_u^{ent,c} = \sum_{t \in \mathcal{N}_u} \frac{1}{|\mathcal{N}_u|} \mathbf{z}_t^{ent,c}. \tag{8}$$

Here $p(t,c) \in [0,1]$ denotes the affiliation degree between item t and factor c. g is a nonlinear mapping module. The design above is based on the principle that: (1) $\mathbf{z}_v^{slf}$ will first be assigned to some factors c according to $p(v,c)$ scores, and then contribute to the corresponding higher-level embeddings after the nonlinear mapping; (2) For those embeddings that are already factorized (i.e., $\mathbf{z}_t^{ent,c}$), they will directly contribute to higher-level counterparts (i.e., $\mathbf{z}_u^{ent,c}$) within the same factor c. It is worth noting that we include two index sets $\{c|1 \le c \le C_1\}$ and $\{c|1 \le c \le C_2\}$ because they index different factors. Also, if we have more layers of entity nodes, their embeddings could be learned just via normal GCNs without applying our method, because entity nodes in different layers are still of the same type.

Till now, the learnable parameters Θ in our model include: $\mathbf{z}_e^{slf}$, $\mathbf{z}_t^{slf}$, $\mathbf{z}_u^{slf}$, $p(e,c)$ and $p(t,c')$ for $e \in \mathcal{E}$, $t \in \mathcal{T}$, $u \in \mathcal{U}$, $1 \le c \le C_1$ and $1 \le c' \le C_2$.

4.4 The Proposed Model Architecture

We now introduce the model details for learning node embeddings, estimating factor affiliation scores, and making predictions. In general, the recommendation model is based on a variational autoencoder (using other encoder-decoder schemes may also work, but is beyond our discussion). The model details are as below.

4.4.1 Encoder. The encoder returns a user embedding $\mathbf{z}_u$ and an item embedding $\mathbf{z}_t$ for each interaction. Encoder modules are based on graph convolution designed in the above subsection, with the new `COMBINE()` and `AGGREGATE()` operations. To estimate factor affiliation degrees $p(e,c)$ and $p(t,c)$, we maintain two sets of embeddings $\{\mathbf{z}_c^{ent}|1 \le c \le C_1\}$ and $\{\mathbf{z}_c^{itm}|1 \le c \le C_2\}$ as learnable parameters for each factor c. Then we define

$$p(e,c) = softmax(\cos(\mathbf{z}_e^{slf}, \mathbf{z}_c^{ent})/\gamma), \tag{9}$$

$$p(t,c) = softmax(\cos(\mathbf{z}_t^{slf}, \mathbf{z}_c^{itm})/\gamma), \tag{10}$$

where $softmax()$ normalizes over different factors, γ is a hyperparameter to skew distributions as a form of self-training [27, 46]. For example, making $\gamma < 1$ amplifies the gap between strong and weak assignments. Through explicitly parameterizing factor embeddings, we reduce the number of parameters than treating $p(e,c)$ or $p(t,c)$ as independent coefficients over different argument pairs.

During the training process, we adopt the variational autoencoder paradigm [20]. The generation of $\mathbf{z}_u$ and $\mathbf{z}_t$ are not deterministic, but are under certain probabilistic distribution. Also, we let different segments be mutually independent, so distributions are defined as below:

$$q(\mathbf{z}_t|\mathcal{G}) = q(\mathbf{z}_t^{slf}|\mathcal{G}) \cdot \prod_{c=1}^{C_1} q(\mathbf{z}_t^{ent,c}|\mathcal{G}), \tag{11}$$

$$q(\mathbf{z}_u|\mathcal{G}) = q(\mathbf{z}_u^{slf}|\mathcal{G}) \cdot \prod_{c=1}^{C_2} q(\mathbf{z}_u^{itm,c}|\mathcal{G}) \cdot \prod_{c=1}^{C_1} q(\mathbf{z}_u^{ent,c}|\mathcal{G}). \tag{12}$$

The probability of each segment follows a multivariate normal distribution. According to the designed information flow, $q(\mathbf{z}_t^{ent,c}|\mathcal{G}) = N(\boldsymbol{\mu}_t^{ent,c}, \mathrm{diag}((\boldsymbol{\sigma}_t^{ent,c})^2))$, where by following Eq(6), we define $\boldsymbol{\mu}_t^{ent,c} = g^{\mu}(\sum_{e\in\mathcal{N}_t} \frac{p(e,c)}{|\mathcal{N}_t|}\mathbf{z}_e^{slf})$, and $\boldsymbol{\sigma}_t^{ent,c} = g^{\sigma}(\sum_{e\in\mathcal{N}_t} \frac{p(e,c)}{|\mathcal{N}_t|}\mathbf{z}_e^{slf})$. The generation of other segments are similar, so we skip them here.

4.4.2 Decoder. The decoder computes the similarity between $\mathbf{z}_u$ and $\mathbf{z}_t$ jointly over various factors, as illustrated in Figure 4(c):

$$p(Y_{u,t}|\mathbf{z}_u, \mathbf{z}_t) \propto \exp(\langle \mathbf{z}_u^{slf}, \mathbf{z}_t^{slf}\rangle) + \sum_{c=1}^{C_1} p(t,c)\exp(\langle \mathbf{z}_u^{itm,c}, \mathbf{z}_t^{slf}\rangle) + \sum_{c=1}^{C_2} \exp(\langle \mathbf{z}_u^{ent,c}, \mathbf{z}_t^{ent,c}\rangle), \tag{13}$$

where $\langle\cdot,\cdot\rangle$ denotes inner product. The first term can be seen as the traditional collaborative filtering component. The second term contributes to prediction based on user historical interactions, where the embedding space contains C_1 factors. The third term contributes to prediction based on knowledge entities, where the embedding space contains C_2 factors. In this work, however, we discard the first term because it hurts the interpretability of the model. Also, it is worth noting that $p(t,c)$ in the second term is necessary, since it scales $\mathbf{z}_t^{slf}$ corresponding to different factors.

4.4.3 Learning. The overall likelihood is decomposed into the sum of likelihoods of individual interactions as $\log p(\mathbf{Y}) = \sum_{u,t} p(Y_{u,t})$. In practice, we optimize its variational lower bound L_{ELBO} [20]:

$$\begin{aligned} L_{\mathrm{ELBO}}(Y_{u,t}) =& \mathbb{E}_{q(\mathbf{z}_u,\mathbf{z}_t|\mathcal{G})} \log p(Y_{u,t}|\mathbf{z}_u,\mathbf{z}_t) \\ &- \mathrm{KL}[q(\mathbf{z}_u,\mathbf{z}_t|\mathcal{G})||p(\mathbf{z}_u,\mathbf{z}_t)], \end{aligned} \tag{14}$$

where $\mathrm{KL}[\cdot||\cdot]$ is the Kullback-Leibler divergence. The definition of $p(Y_{u,t}|\mathbf{z}_u,\mathbf{z}_t)$ could be found in Eq(13) with normalization over all item candidates. We also assume that $\mathbf{z}_u$ and $\mathbf{z}_t$ are generated independently, so $p(\mathbf{z}_u,\mathbf{z}_t) = p(\mathbf{z}_u)\cdot p(\mathbf{z}_t)$ and $q(\mathbf{z}_u,\mathbf{z}_t|\mathcal{G}) = q(\mathbf{z}_u|\mathcal{G})\cdot q(\mathbf{z}_t|\mathcal{G})$. We further take Gaussian prior, where $p(\mathbf{z}_u) = N(\mathbf{z}_u|\mathbf{0},\mathbf{I})$ and $p(\mathbf{z}_t) = N(\mathbf{z}_t|\mathbf{0},\mathbf{I})$. $q(\mathbf{z}_u|\mathcal{G})$ and $q(\mathbf{z}_t|\mathcal{G})$ are generated by encoders as introduced before.

5 EXPERIMENT

In this section, we evaluate the proposed model on several real-world datasets. First, we evaluate the effectiveness of the proposed model. Second, we quantitatively measure the accuracy of interpretation, and analyze the insight from interpretation.

Dataset	MovieLens	LastFM	Yelp
#users	138,159	1,872	45,919
#items	16,954	3,846	45,538
#entities	85,615	5,520	1,341
#edges	13,501,622	42,346	1,185,068
D	30	16	25
C	4	4	6
γ	0.1	0.1	0.1
lr	4×10^{-4}	2×10^{-4}	2×10^{-4}
l_2 weight	10^{-8}	10^{-8}	5×10^{-9}
batch_size	512	128	256
epoch	10	100	15

Table 1: Statistics of the datasets.

5.1 Datasets and Metrics

- **MovieLens**[1]: A benchmark dataset from movie recommendation. Original ratings are converted into binary scores, indicating whether a user has interacted with a movie [15].
- **LastFM**[2]: A dataset obtained from the Last.fm online music website. Only interactions between users and music artists are used. The social network information is not included.
- **Yelp**[3]: A dataset adopted from Yelp challenge. Restaurants and bars are used as items. Attributes in the dataset are used as knowledge entities. Each user and item have at least 10 interactions.

Each dataset is associated with a knowledge graph as side information released in [41, 44]. The datasets' statistics are available in Table 1. For each dataset, we randomly hold out 200 users for validation and 200 users for testing. For both groups of users, 80% of interactions are randomly sampled and put into training data, while the remaining are put into validation and testing data. The model to be tested has the best performance on the validation dataset.

The hyperparameter settings of the proposed model on each dataset are listed in Table 1. Specifically, D denotes the dimension of each factor's embedding. C denotes the total number of factors, so that the total embedding dimension is $2C\times D$ for each user node, $(C+1)\times D$ for each item node, and D for each entity node. lr is the learning rate. l_2 weight is the regularization weight for model parameters. Batch size corresponds to the number of users put into models for training, so that the number of batches in each epoch is #users/batch_size. For the decoder part in Eq(13), instead of using $\mathbf{z}_t^{slf}$ and $\mathbf{z}_t^{ent,c}$ from GCN, a more effective solution is to maintain another embedding dictionary as learnable parameters to replace them [27]. Such a replacement is beneficial for model performance. The interpretation for user interest is not affected. The reported experimental results are based on this practical modification. The effectiveness of models is evaluated under two metrics, *Recall@k* ($k = 2, 10, 50, 100$) and *NDCG*@100.

5.2 Baseline Methods

We compare the proposed model with the most representative state-of-the-art methods described below.

[1]https://grouplens.org/datasets/movielens/
[2]https://grouplens.org/datasets/hetrec-2011/
[3]https://www.yelp.com/dataset/challenge/

	MovieLens				LastFM				Yelp			
	R@2	R@10	R@50	R@100	R@2	R@10	R@50	R@100	R@2	R@10	R@50	R@100
NMF	0.064	0.219	0.435	0.556	0.118	0.247	0.428	0.520	0.021	0.051	0.113	0.152
FM	0.074	0.224	0.432	0.551	0.120	0.250	0.436	0.532	0.022	0.053	0.169	0.228
CKE	0.070	0.221	0.445	0.571	0.121	0.243	0.443	0.548	0.020	0.060	0.177	0.260
RippleNet	0.185	0.242	0.479	0.599	0.080	0.248	0.458	0.569	0.033	0.064	0.195	0.275
KGCN	0.190	0.248	0.470	0.603	0.110	0.242	0.462	0.588	0.032	0.071	0.192	0.271
VGAE	0.190	0.252	0.478	0.597	0.112	0.262	0.473	0.585	0.033	0.081	0.202	0.280
Splitter	0.185	0.242	0.472	0.589	0.093	0.238	0.462	0.550	0.033	0.061	0.189	0.271
Proposed	0.210	0.248	0.500	0.620	0.128	0.281	0.514	0.615	0.050	0.089	0.200	0.275

Table 2: Recommendation performance comparison in *Recall@k*.

- **NMF** [23] is a benchmark collaborative filtering model for recommendation, based on non-negative matrix factorization. The number of factors is chosen as 16, 16, 12 for MovieLens, LastFM and Yelp. L_1 and L_2 regularizations are weighted equally, with regularization coefficient of 0.005.
- **FM** [30] is a benchmark factorization model which combines second-order feature interactions with linear modeling. We include user IDs, item IDs and items' associated knowledge entities as input. The total dimension of embedding is the same as the proposed model.
- **CKE** [51] extends collaborative filtering with side knowledge information as regularization for recommendations. We use both user-item interactions and knowledge entities as input. The weight for knowledge entity part is set as 0.2 for all datasets.
- **RippleNet** [40] is a memory-network-like approach which propagates from items to other entities in the knowledge graph. We set $dim = 16$, $H = 2$, $\lambda_1 = 10^{-6}$, $\lambda_2 = 0.01$, $\eta = 0.01$ for MovieLens; $dim = 16$, $H = 3$, $\lambda_1 = 10^{-5}$, $\lambda_2 = 0.02$, $\eta = 0.005$ for LastFM; $dim = 16$, $H = 2$, $\lambda_1 = 10^{-7}$, $\lambda_2 = 0.02$, $\eta = 0.01$ for Yelp.
- **KGCN** [42] is a GCN-based model that captures inter-item relations by mining their attributes on a knowledge graph (KG). We set $dim = 120$, $H = 2$, $\lambda = 10^{-7}$, $\eta = 2 \times 10^{-2}$ for MovieLens; $dim = 64$, $H = 1$, $\lambda = 2\times10^{-5}$, $\eta = 2\times10^{-4}$ for LastFM; $dim = 64$, $H = 2$, $\lambda = 5 \times 10^{-7}$, $\eta = 5 \times 10^{-4}$ for Yelp.
- **VGAE** [21] is built upon a graph variational autoencoder. It can be regarded as the fundamental version of the proposed model where the latent dimensions are not disentangled. The hyperparameter setting is similar to the proposed method, except $dim = 120$ for MovieLens, $dim = 64$ for LastFM, and $dim = 64$ for Yelp. The learning rate is 5×10^{-4} for MovieLens, 2×10^{-4} for LastFM, and 3×10^{-4} for Yelp.
- **Splitter** [6] learns multiple embedding vectors for each node. A clustering step is performed independently as factor identification before embedding learning. Another similar work is [26]. We use overlapping community detection for user-item subgraph and item-entity subgraph. The community detection results are used for factor assignment. Then, we apply C GCN models to learn several embeddings for each node. The community assignments remain fixed. The resultant embeddings are then concatenated for joint prediction. The embedding dimensions and the number of communities are set the same as the proposed model.

	MovieLens	LastFM	Yelp
NMF	0.2920	0.1983	0.0681
FM	0.3020	0.2383	0.0820
CKE	0.3064	0.2392	0.0959
RippleNet	0.3153	0.2321	0.1078
KGCN	0.3230	0.2493	0.1168
VGAE	0.3234	0.2535	0.1180
Splitter	0.3242	0.2331	0.1039
Proposed	0.3427	0.2656	0.1202

Table 3: Recommendation evaluation in *NDCG*@100.

5.3 Recommendation Performance Evaluation

We first compare the performance of the proposed method with the baseline methods. The performances are presented in Table 2 and Table 3. Some observations are drawn below:

- The proposed model is at least comparable to the best performance in most cases. VGAE can be seen as the variant of the proposed model without resolving representation learning. It thus shows that considering interpretability will not negatively affect recommendation performances.
- Although also emphasizing the idea of disentangled representation learning, the proposed model achieves better performance than Splitter. It demonstrates the advantage of simultaneously learning embedding and identifying factors over treating them as independent steps. The embedding learning in Splitter could be negatively affected by the gap of factor activation between training and inference.
- VGAE achieves slightly better performance than KGCN. As we ignore link types in graphs, which is not our focus in this work, the improvement could be from the variational autoencoder.
- NMF does not utilize attribute entities, which has a negative impact on its performance, but it serves as the baseline to reflect the degree that attribute information helps in recommendation.

5.3.1 Hyperparameter Sensitivity Analysis. The most important hyperparameter in this work is the number of factors C_1 and C_2. Therefore, we further analyze model performances by changing factor numbers. Here we let $C_1 = C_2 = C$. Results are reported in Table 4. For each pair of values, the left value means that we fix

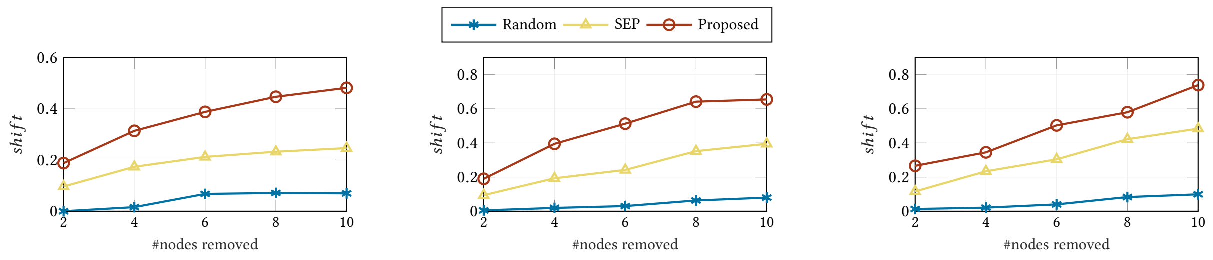

Figure 5: Quantitative evaluation of explanation faithfulness on MovieLens (left), LastFM (middle) and Yelp (right).

C	MovieLens	LastFM	Yelp
$c_0 - 2$	0.3398/0.3437	0.2541/0.2722	0.1170/0.1267
$c_0 - 1$	0.3422/0.3439	0.2613/0.2754	0.1195/0.1234
c_0	0.3427/0.3427	0.2656/0.2656	0.1202/0.1202
$c_0 + 1$	0.3429/0.3308	0.2721/0.2547	0.1204/0.1098

Table 4: Performance of the proposed model in $NDCG@100$ by varying the number of factors C, where $c_0 = 4, 4, 6$ for MovieLens, LastFM and Yelp, respectively.

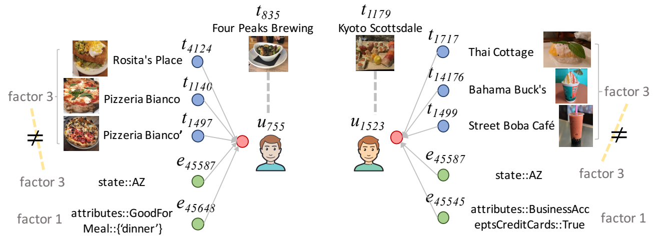

Figure 6: Visualization of the explanations.

the dimension D of each embedding segment, while the right value means that we fix the total embedding dimension $C \times D$.

We could observe that, by increasing C, model performance in general increases if we fix D, since more factors are considered and the parameter size also increases. The time complexity and memory cost will also increase in this case. However, the model performance decreases if we set C too large while keeping $C \times D$ fixed, since the dimension for each factor decreases and is not adequate to contain all the information needed.

5.4 Evaluation of Explanation Accuracy

In this part, we will first evaluate the faithfulness of interpretation obtained from our model. Then, we qualitatively analyze the insights from interpretation. Explanation information comes from two parts: (1) affiliation scores tracing the information flows; (2) factors extracted on different embedding segments.

5.4.1 Quantitative Evaluation of Explanation. Evaluating the correctness of model explanation quantitatively is a challenging task, because usually there is no ground-truth explanation to compare with. To tackle this, we utilize the duality between interpretation and adversarial perturbation [7]. The high-level intuition is that, after removing the features that are important to the current prediction, the model prediction should dramatically change.

Specifically, we use the 200 users in testing data for evaluation. After performing recommendation, interpretation is obtained as the historical items or important attribute entities. For each user-item pair (u, t), we compute the importance score of an item/entity j as $s(j; u, t)$. The exact definition of $s(j; u, t)$ is introduced in Appendix. In brief, $s(j; u, t)$ is computed based on the affiliation scores $p(e, c)$ and $p(t, c)$ as they trace contribution from entities to items and from items to users. Then, the items/entities with large importance scores are removed from the original input. Finally, we perform recommendations again with the new input. Let $recall$ and $recall'$ denote the $Recall@10$ performance before and after the perturbation, respectively. We define $shift = \frac{recall - recall'}{recall}$ to measure the explanation accuracy. A higher value indicates more accurate explanation.

We also introduce two baseline methods to compare with the proposed interpretation scheme. Popular methods such as LIME [31] is not included since there is no explicit concept of classes in recommendation settings.

- **Random**: We randomly remove the same number of historical items as the proposed method.
- **SEP** [47]: A post-hoc path-based interpretation method for recommender systems. Items in the resultant explanation paths are retrieved as interpretation.

The experimental results are shown in Figure 5. We make 5 runs for each case and take the average. We vary the number of items to remove in x-axis, and the $shift$ is y-axis. Observations are summarized as follows:

- The interpretation obtained directly from model inference is more accurate than using post-hoc explanations. It thus validates the benefit of designing explainable models over using external explanation methods.
- The amount of information removed is small compared with the overall input. Also, the $shift$ value increases as the amount of removed information increases. It thus further demonstrates the faithfulness of the obtained explanations.

5.4.2 Case Studies. After the quantitative evaluation, we provide some visualization of explanation results in Figure 6. We can see that restaurant t_{835} is recommended to user u_{755} because the user has visited similar restaurants (near Phoenix) and his activities are mainly in Arizona emphasized by attribute information. Similarly, restaurant t_{1179} is recommended to user u_{1523} because he has visited many Asian restaurants. The factor that each node is affiliated with is also shown.

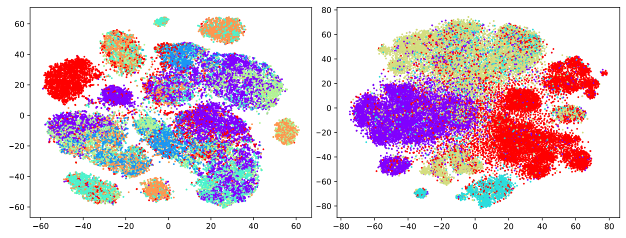

Figure 7: Visualization of embeddings for Yelp dataset.

We also observe an interesting phenomenon in the second recommendation. The recommendation is partially explained by the attribute e_{45545}, which indicates that the restaurant accepts credit cards for payment. However, intuitively it may not be a good reason shown to users to explain recommendation results in practice. The reason that this attribute is selected could be that many restaurants accept credit cards, but the strong correlation is not suitable as the reason to persuade users. This could be a limitation for graph convolution models. It also reminds us that interpretability is not equal to credibility, and we leave it for future exploration.

5.4.3 Visualization of Latent Dimensions. In this part, we visualize the item embeddings in Yelp dataset in Figure 7. The left plot follows the previous settings where $C_1 = 6$, while the right plot is drawn as we let $C_1 = 4$. The color of each point indicates its closest affiliated factor. Some colors are mixed since an embedding could be assigned to multiple factors, but only one factor could be shown for each embedding. The clustering phenomenon is consistent with factor affiliations. It demonstrates that our model is able to discover different factors from data and assign embeddings accordingly.

6 RELATED WORK

Machine Learning for Recommender Systems. In addition to traditional content-based and neighborhood-based methods [9, 25], advanced machine learning methods have been developed for recommendation, which could be categorized into two groups: linear and nonlinear. Examples of linear methods include matrix factorization (MF) and factorization machines (FMs). Specifically, MF [1] represents an individual user or item by a single latent vector and models the interaction of a user-item pair by the inner product of latent vectors. FMs [30] embed features into a latent space and model user-item interactions by summing up the inner products of embedding vectors between all pairs of features. To this end, a lot of recent efforts have been put on modeling user-item interactions in a nonlinear way such as using DNNs. Example studies include the Wide & Deep model for app recommendation in Google Play [3], the non-linear extensions of MF and FM [14, 15], the recommender modeled by recurrent neural networks (RNNs) [4], etc. It has been demonstrated that these nonlinear models usually yield better effectiveness, where a comprehensive review is available in [54].

Interpretable Machine Learning. Model interpretability receives increasing attention recently [5, 28]. The efforts can be divided into two categories, post-hoc interpretation methods and intrinsically interpretable models. Specifically, post-hoc methods can be further divided into global and local methods, where the former transforms black-box models into understandable structures [52], and the latter explains individual predictions [31, 35]. There are gradient-based methods that measure feature contributions on output or intermediate neurons through backpropagation [35], perturbation-based methods [7], approximation-based methods [31], and entropy-based methods [10]. Intrinsically interpretability focuses on developing transparent model components. A typical strategy is to pose regularization on parameters to conform with certain prior [17]. [43] proposes to incorporate expert knowledge to penalize model weights. [37] develops disentangled MLPs to explicitly regulate feature interactions between layers.

Explainable Recommender Systems. The explainability of recommender systems recently starts to be seen as an important topic [55]. Before the popularity of deep models, explicit factor models have been proposed to relate explicitly constructed features with latent factors as explanations for the factors [56]. To balance effectiveness and explainability, a common strategy is to feed interpretable models with engineered features [16] or construct domain-specific knowledge graphs [8]. Graphs can be used as the basis to pass information for embedding learning [40, 44, 45]. [18] proposes an attention based model, but it only works on text data. [27] proposes a new recommendation model to learn disentangled representations, but how to apply it into general scenarios containing extra attribute information and knowledge is beyond the discussion. To discriminate different semantics in embedding, [6, 26] extend random-walk-based models and use multiple embeddings to represent each node, but embedding learning and factor extraction are done separately.

7 CONCLUSION AND FUTURE WORK

In this work, we tackle the explainability problem in recommender systems. Specifically, we first analyze the IOM elements that benefit model interpretability, and then propose a novel model with a more transparent representation learning module. The IOM elements include reshaping input into discrete concepts, making output attributable, and disentangling the middle-level representations. The proposed model considers all the three elements above. The data is formatted as a graph. The model resolves information passing in graph convolution according to different source types and factors. Different from previous work, in our model, the representation learning process and factor identification are achieved simultaneously. Experiments on real-world datasets validate the effectiveness and interpretation accuracy of our model.

The future work includes: (1) developing more effective graph construction methods for automatic input conceptualization; (2) designing human-computer interaction interfaces to better render interpretation results to users; (3) extending the proposed model to handle a larger number of facets.

REFERENCES

[1] Linas Baltrunas, Bernd Ludwig, and Francesco Ricci. 2011. Matrix factorization techniques for context aware recommendation. In *RecSys*.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: a review and new perspectives. *IEEE TPAMI* (2013).

[3] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *DLRS*.

[4] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. 2017. Sequential user-based recurrent neural network recommendations. In *RecSys*.

[5] Mengnan Du, Ninghao Liu, and Xia Hu. 2018. Techniques for interpretable machine learning. *arXiv preprint arXiv:1808.00033* (2018).

[6] Alessandro Epasto and Bryan Perozzi. 2019. Is a single embedding enough? Learning node representations that capture multiple social contexts. In *WWW*.

[7] Ruth C Fong and Andrea Vedaldi. 2017. Interpretable explanations of black boxes by meaningful perturbation. In *ICCV*.

[8] Jingyue Gao, Xiting Wang, Yasha Wang, and Xing Xie. 2019. Explainable recommendation through attentive multi-view learning. In *AAAI*.

[9] Yong Ge, Hui Xiong, Alexander Tuzhilin, and Qi Liu. 2014. Cost-aware collaborative filtering for travel tour recommendations. *ACM TOIS* (2014).

[10] Chaoyu Guan, Xiting Wang, Quanshi Zhang, Runjin Chen, Di He, and Xing Xie. 2019. Towards a deep and unified understanding of deep neural models in NLP. In *ICML*.

[11] David Gunning and David W Aha. 2019. DARPA's explainable artificial intelligence program. *AI Mag.* (2019).

[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*.

[13] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: methods and applications. *arXiv preprint arXiv:1709.05584* (2017).

[14] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *SIGIR*.

[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*.

[16] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *KDADD*.

[17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-VAE: learning basic visual concepts with a constrained variational framework. In *ICLR*.

[18] Liang Hu, Songlei Jian, Longbing Cao, and Qingkui Chen. 2018. Interpretable Recommendation via Attraction Modeling: Learning Multilevel Attractiveness over Multimodal Movie Contents.. In *IJCAI*.

[19] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. 2018. Interpretability beyond feature attribution: quantitative testing with concept activation vectors (TCAV). In *ICML*.

[20] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[21] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[22] Deguang Kong, Chris Ding, and Heng Huang. 2011. Robust nonnegative matrix factorization using l21-norm. In *CIKM*.

[23] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *IEEE Comput.* (2009).

[24] Chenliang Li, Cong Quan, Li Peng, Yunwei Qi, Yuming Deng, and Libing Wu. 2019. A capsule network for recommendation and explaining what you like and dislike. In *SIGIR*.

[25] Huayu Li, Richang Hong, Defu Lian, Zhiang Wu, Meng Wang, and Yong Ge. 2016. A relaxed ranking-based factor model for recommender system from implicit feedback. In *IJCAI*.

[26] Ninghao Liu, Qiaoyu Tan, Yuening Li, Hongxia Yang, Jingren Zhou, and Xia Hu. 2019. Is a single vector enough? Exploring node polysemy for network embedding. In *KDD*.

[27] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019. Learning disentangled representations for recommendation. In *NeurIPS*.

[28] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2018. Methods for interpreting and understanding deep neural networks. *DSP* (2018).

[29] Georgina Peake and Jun Wang. 2018. Explanation mining: post hoc interpretability of latent factor models for recommendation systems. In *KDD*.

[30] Steffen Rendle. 2010. Factorization machines. In *ICDM*.

[31] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why should I trust you?" Explaining the predictions of any classifier. In *KDD*.

[32] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *NMI* (2019).

[33] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *NIPS*.

[34] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *RecSys*.

[35] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).

[36] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825* (2017).

[37] Michael Tsang, Hanpeng Liu, Sanjay Purushotham, Pavankumar Murali, and Yan Liu. 2018. Neural interaction transparency (NIT): disentangling learned interactions for improved interpretability. In *NIPS*.

[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[39] Hao Wang, Tong Xu, Qi Liu, Defu Lian, Enhong Chen, Dongfang Du, Han Wu, and Wen Su. 2019. MCNE: An end-to-end framework for learning multiple conditional network representations of social network. In *KDD*.

[40] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: propagating user preferences on the knowledge graph for recommender systems. In *CIKM*.

[41] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *KDD*.

[42] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge graph convolutional networks for recommender systems. In *WWW*.

[43] Jiaxuan Wang, Jeeheh Oh, Haozhu Wang, and Jenna Wiens. 2018. Learning credible models. In *KDD*.

[44] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: knowledge graph attention network for recommendation. In *KDD*.

[45] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *AAAI*.

[46] Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *ICML*.

[47] Fan Yang, Ninghao Liu, Suhang Wang, and Xia Hu. 2018. Towards interpretation of recommender systems with sorted explanation paths. In *ICDM*.

[48] Min Yang, Wei Zhao, Jianbo Ye, Zeyang Lei, Zhou Zhao, and Soufei Zhang. 2018. Investigating capsule networks with dynamic routing for text classification. In *EMNLP*.

[49] Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. 2018. Exploring visual relationship for image captioning. In *ECCV*.

[50] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *KDD*.

[51] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *KDD*.

[52] Quanshi Zhang, Ruiming Cao, Feng Shi, Ying Nian Wu, and Song-Chun Zhu. 2018. Interpreting CNN knowledge via an explanatory graph. In *AAAI*.

[53] Quanshi Zhang and Song-Chun Zhu. 2018. Visual interpretability for deep learning: a survey. *FITEE* (2018).

[54] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *CSUR* (2019).

[55] Yongfeng Zhang and Xu Chen. 2018. Explainable recommendation: a survey and new perspectives. *arXiv preprint arXiv:1804.11192* (2018).

[56] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*.

[57] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. 2018. Interpretable basis decomposition for visual explanation. In *ECCV*.

A APPENDIX

A.1 Node Importance Estimation

This part is to briefly introduce the computation of $s(j;u,t)$ in Section 5.4. The importance computation closely follows the information propagation path in graph convolution. Specifically, if j is an item node, then its importance within factor c is:

$$s(j,c;u,t) = p(t,c)\exp(\langle \mathbf{z}_u^{itm,c}, \mathbf{z}_t^{slf}\rangle) \cdot \mathit{1}^{(u,j)} \frac{p(j,c)}{|\mathcal{N}_u|},$$

where the first part is from the decoder, and the second half is factor affiliation degree. $\mathit{1}^{(m,j)} = 1$ if j connects to u. Here we let $s(j;u,t) = \sum_c s(j,c;u,t)$.

If j is an entity node, then its importance within factor c is:

$$s(j,c;u,t) = \exp(\langle \mathbf{z}_u^{ent,c}, \mathbf{z}_t^{ent,c}\rangle) \cdot \sum_{m \in |\mathcal{N}_u|} \frac{1}{|\mathcal{N}_u|} \mathit{1}^{(m,j)} \frac{p(j,c)}{|\mathcal{N}_m|},$$

where the first part is from the decoder, and the second part is traced from information propagation path in GCN. Similarly, we let $s(j;u,t) = \sum_c s(j,c;u,t)$.