Neural Feature-aware Recommendation with Signed Hypergraph Convolutional Network

XU CHEN, Renmin University of China
KUN XIONG, Tsinghua University
YONGFENG ZHANG, Rutgers University
LONG XIA, York University
DAWEI YIN, Baidu Inc., Beijing, China
JIMMY XIANGJI HUANG, York University

Understanding user preference is of key importance for an effective recommender system. For comprehensive user profiling, many efforts have been devoted to extract user feature-level preference from the review information. Despite effectiveness, existing methods mostly assume linear relationships among the users, items, and features, and the collaborative information is usually utilized in an implicit and insufficient manner, which limits the recommender capacity in modeling users' diverse preferences. For bridging this gap, in this article, we propose to formulate user feature-level preferences by a neural signed hypergraph and carefully design the information propagation paths for diffusing collaborative filtering signals in a more effective manner. By taking the advantages of the neural model's powerful expressiveness, the complex relationship patterns among users, items, and features are sufficiently discovered and well utilized. By infusing graph structure information into the embedding process, the collaborative information is harnessed in a more explicit and effective way. We conduct comprehensive experiments on real-world datasets to demonstrate the superiorities of our model.

CCS Concepts: • **Information systems** → **Collaborative filtering**; **Personalization**;

Additional Key Words and Phrases: Recommendation system, collaborative filtering, graph convolutional network, feature-based recommendation, hypergraph neural network

ACM Reference format:
Xu Chen, Kun Xiong, Yongfeng Zhang, Long Xia, Dawei Yin, and Jimmy Xiangji Huang. 2020. Neural Feature-aware Recommendation with Signed Hypergraph Convolutional Network. *ACM Trans. Inf. Syst.* 39, 1, Article 8 (November 2020), 22 pages.
https://doi.org/10.1145/3423322

This research is supported by Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098, National Natural Science Foundation of China (No. 61832017) and the Natural Sciences and Engineering Research Council (NSERC) of Canada and the York Research Chairs (YRC) program.
Authors' addresses: X. Chen (corresponding author), Beijing Key Laboratory of Big Data Management and Analysis Methods, Gaoling School of Artificial Intelligence, Renmin University of China; email: successcx@gmail.com; K. Xiong, School of Software, Tsinghua University; email: xk18@mails.tsinghua.edu.cn; Y. Zhang, Department of Computer Science, Rutgers University; email: yongfeng.zhang@rutgers.edu; L. Xia and J. X. Huang, School of Information Technology, York University; emails: {longxia, jhuang}@yorku.ca; D. Yin, Baidu Inc., Beijing, China; email: yindawei@acm.org.

1046-8188/2020/11-ART8 $15.00
https://doi.org/10.1145/3423322

1 INTRODUCTION

Recommender system, as an effective information filtering method, has been deployed in many real-world applications, ranging from e-commerce [28], video sharing websites [14], to social networks [40] and online news platforms [50]. The core of a successful recommender system lies in the accurate understanding of individual user preferences. For comprehensive user modeling, recent years have witnessed an emerging trend on enhancing recommender algorithms with user review information. Comparing with traditional rating or interaction data, user reviews are much more informative [23], pooling an extensive wealth of knowledge about user opinions and sentiments, which builds a basis for more comprehensive user perception and accurate recommendation [15, 25, 51].

In the field of review-based recommender systems, there are generally two classes of methods. On one hand, many models process the review information on the document level. Each user review is transformed into a semantic vector to improve the user/item representations based on shallow [25, 29] or deep [5, 51] models. Despite being straightforward, these algorithms inevitably involve lots of irrelevant review contents (e.g., "Bought it for my girlfriends birthday" in Figure 1) in the modeling process, which may bias the recommendation performance. On the other hand, for more clear and accurate modeling, recent methods (a.k.a. feature-aware recommendation) preprocess the review information by extracting user feature-level preference. The raw user reviews are formatted into "(user, item, feature, sentiment)" quadruples. As exampled in Figure 1, the review of "...The image quality of this iPhone is superb. The battery life is great..." will be converted to "(Elkin Gonzalez V., iPhone X, image quality, positive)" and "(Elkin Gonzalez V., iPhone X, battery life, positive)," respectively. With such structured review information, models in this line represent each user, item, or feature as a low-rank latent vector, and capture their correlations based on coupled matrix [15, 48] or tensor factorization [33]. The finally learned user-feature correlations are expected to profile users in a more comprehensive and finer-grained manner.

While feature-aware recommendation has achieved many promising results, there are still some inherent limitations: (1) **on the embedding process**, most methods base their embedding schemes solely on the ID information while ignoring the entity[1] interaction signals, which may yield insufficient collaborative modeling and suboptimal recommendation performance [35]. (2) **On the predictive function**, previous models mostly assume linear relationships between different entities, and the employed linear predictive functions can be less effective in capturing users' diverse preferences in real-world scenarios. To bridge these gaps in a unified framework, we propose to revisit the task of feature-aware recommendation with a signed hypergraph convolutional network (see Figure 2(a)). In general, we regard each user, item, or feature as a node, and the hyper-edges formulate the "user-item-feature" interactions. Unlike previous methods, where the collaborative information is implicitly inferred from the sparse supervision signals, our model explicitly encodes the graph structure information into the embedding process for sufficient collaborative modeling. More specifically, we borrow the idea of graph convolutional network (GCN) [18] for node representation. Each node embedding not only encodes its own information, but also will be enhanced by its local neighborhoods, which has been demonstrated to be a powerful representation learning strategy [18]. For taming the complex entity relationships, we predict the signs of the hyper-edges based on a neural architecture, which allows us to lower the risk of model misspecification (i.e., imposing incorrect constraints on user-item-feature interactions).

Challenges: While this seems to be a promising direction to improve feature-aware recommendation, it is non-trivial due to the following challenges: (1) *how to handle differently signed*

[1]We use "entity" as an umbrella term for a user, an item, or a feature.

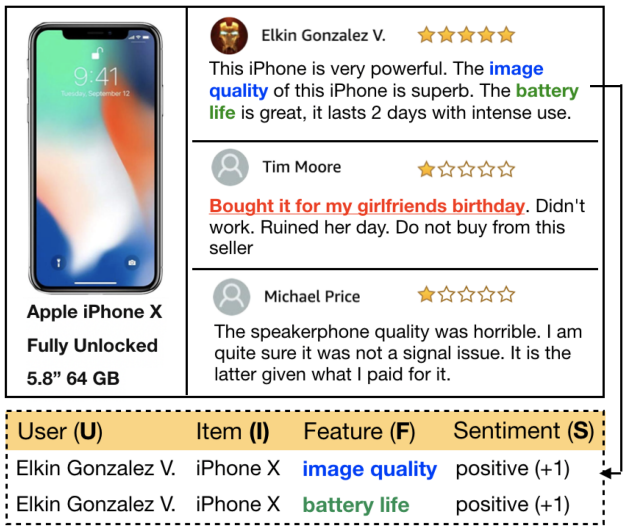

Fig. 1. Example of extracting user feature-level preference from the review information.

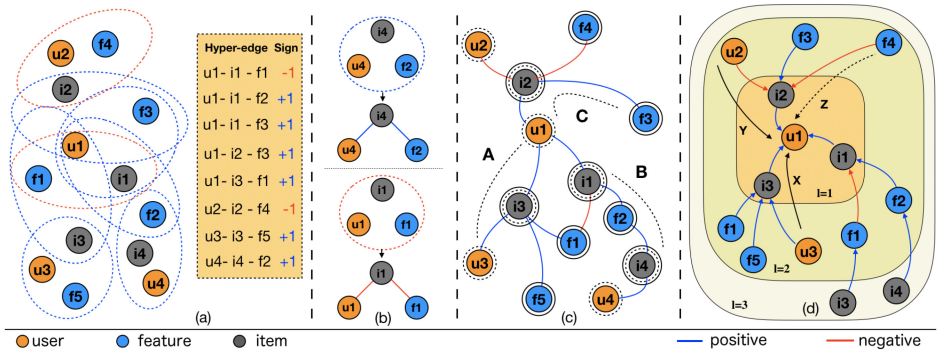

Fig. 2. (a) A toy example of formulating feature-aware recommendation with a signed hypergraph network. Each node represents a user, an item, or a feature, the hyper-edge with "+1" means the user positively comments on the item feature, while "−1" represents a negative sentiment. (b) Signed hyper-edge corruption. (c) The user-item-feature graph corresponding to (a), nodes with solid circle form the item-feature subgraph, and dotted circles make up the user-item subgraph. (d) The tree structure rooted from u1 on (c). Path X indicates consistent semantic between u3 and u1. Path Y suggests contrary characters between u2 and u1. Path Z is not allowed for information propagation.

hyper-edges in a hypergraph? It is intuitive that a user may express different sentiments towards an item feature, so in our problem, hyper-edges can be associated with both positive or negative weights. Although unsigned hypergraph has been studied a lot before [1, 13, 43], few work investigates the signed one for building recommender models, which need us to carefully design the corresponding principles. (2) *How to conduct neighbor aggregation on a signed hypergraph?* While neighbor aggregation is a standard component in traditional GCN [18], it is still unclear on how to configure such an operation in a signed hypergraph. The availability of negative hyper-edges may challenge the fundamental homophily assumption (i.e., connected entities are more similar than those without links) in unsigned graphs.

Solutions: For solving these challenges, we extend each signed hyperedge into ordinary ones according to the characters of feature-aware recommendation. And then, we utilize a dual-embedding mechanism to seamlessly handle differently signed edges and design tailored information propagation paths to capture the collaborative filtering (CF) nature in a more effective manner.

Contributions: In summary, in this article, we propose to improve feature-aware recommendation by a novel **S**igned **H**ypergraph **C**onvolutional **N**etwork (called **SHCN** for short). Our model enriches the principles of graph neural network by filling gaps in the convolutional operation for signed hyper-edges. For adapting review-based recommendation, we design a customized neighbor aggregation strategy for effective user preference delivering. Extensive experiments on real-world datasets are conducted to verify the effectiveness of our model.

2 PRELIMINARIES

In this section, we describe some basic knowledge of our work.

2.1 Graph Neural Network

In recent years, graph neural network (GNN) has attracted increasing attention from both academic and industry communities. The powerful capabilities in modeling irregular data structures make it shine in many areas, such as knowledge graph [38] and social network [18, 32]. In general, models in this field mostly work on a graph $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V}$ is the node set, and $\boldsymbol{E} \subseteq \boldsymbol{V} \times \boldsymbol{V}$ denotes the edge set. Mathematically, $\mathcal{G}$ is often represented by an adjacency matrix $\boldsymbol{A}(\in \mathbb{R}^{|\boldsymbol{V}|\times|\boldsymbol{V}|})$, where $A_{ij} = 1$ means there is an edge from node i to node j and 0 otherwise. A major limitation of traditional GNNs is that each edge can only connect two nodes. However, in real-world scenarios, the object relations are usually more complex than pairwise [43]. Thus, a nature extension of the traditional graph is the hypergraph [13], where each edge can link more than two nodes (called hyper-edge). A hypergraph $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$ is usually represented by an incidence matrix $\boldsymbol{H}(\in \mathbb{R}^{\boldsymbol{V}\times\boldsymbol{E}})$, where $H_{i\epsilon} = 1$ means $v_i \in \boldsymbol{V}$ is connected by hyper-edge $\epsilon \in \boldsymbol{E}$, and for each column of H, there can be more than two "1"s.

Graph convolutional network (GCN). Graph convolutional network is a successful generalization of Euclidean-based convolutional neural network (CNN). It enhances a node embedding by weighted averaging itself and its neighbors' representations [18]. More specifically, for a node $i \in \boldsymbol{V}$ on $\mathcal{G}$, the graph convolutional operation is formally defined as:

$$\boldsymbol{y}_i = \sigma\left(\boldsymbol{\Theta}^T \sum_{j\in\mathcal{N}_i\cup\{i\}} \frac{\boldsymbol{x}_j}{\sqrt{|\mathcal{N}_i||\mathcal{N}_j|}}\right), \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^{K_x}$ and $\boldsymbol{y} \in \mathbb{R}^{K_y}$ are used to distinguish the original and enhanced embeddings. $\mathcal{N}_i$ is the set of one-hop neighbors of node i on $\mathcal{G}$. j indexes the nodes involved in the convolution operation. $\frac{1}{\sqrt{|\mathcal{N}_i||\mathcal{N}_j|}}$ lowers the effect of the dominate node. $\boldsymbol{\Theta} \in \mathbb{R}^{K_x\times K_y}$ is the weighting parameters, and σ is an active function.

3 PROBLEM DEFINITION

Suppose we have a user set $\mathcal{U}$ and an item set $\mathcal{I}$. The interaction set between the users and items is defined as: $\mathcal{T} = \{(u, i)|u \in \mathcal{U}, i \in \mathcal{I}, \text{user } u \text{ interacted with item } i\}$.[2] The review information is processed according to the above section, and the set of all extracted features is defined as $\mathcal{F}$. We format user reviewing behaviors as a quad set: $\mathcal{W} = \{(u_l, i_l, f_l, s_l)\}_{l=1}^{w}$, where each element

[2]"interacted" in our problem means the user purchased the item and also reviewed on it.

(u_l, i_l, f_l, s_l) means: user $u_l(\in \mathcal{U})$ mentioned feature $f_l(\in \mathcal{F})$ in her review on item $i_l(\in \mathcal{I})$, and the sentiment is $s_l(\in \{-1, +1\})$. Formally, given the dataset of $\{\mathcal{U}, \mathcal{I}, \mathcal{F}, \mathcal{T}, \mathcal{W}\}$, our task is to learn a predictive function g, such that for each user, it can accurately rank all the items according to her preference.

Comparing with traditional recommendation task, which focuses on users' overall preference, feature-aware recommendation additionally involves user sentiments towards item-specific features. The collaborative nature is no longer purely revealed by the item similarities, but also can be reflected from user feature-level preferences. Feature-aware recommendation looks alike tag recommendation, since they both model triplet entity interactions. However, they differ fundamentally in nature: Tags are mainly used to categorize and manage different items [15], while features describe item specific attributes, and we can access users' explicit sentiments towards these attributes in feature-aware recommendation.

For solving the problem of feature-aware recommendation, we can regard each user, item, or feature as a node, and a hyper-edge exactly links three nodes corresponding to a "user-item-feature" interaction. The hypergraph formulation opens the door of leveraging graph convolutional operation to enhance the performance of feature-aware recommendation. However, the hyper-edges in our problem can be both positive and negative, which invalidates the key assumption of traditional unsigned aggregation mechanism. In essence, nodes in an unsigned graph are aggregated by assuming "connected entities are more similar than those without links." However, in a signed hypergraph, connections with different signs represent totally different semantics, which requires tailored aggregation strategies.

4 THE SHCN MODEL

In this section, we detail our framework from three aspects. On the embedding process, we devise a convolution layer for signed hypergraph to encode structure information into the node representations. On the predictive function, we learn a neural model to estimate the labels of the hyper-edges, which relaxes traditional linear assumption on the entity relations. At last, we combine these components and introduce the final optimization function.

4.1 Convolution on Signed Hypergraph

As mentioned before, existing graph convolutional networks mostly concentrate on unsigned graphs [1, 18, 43]. Few research focuses on a signed hypergraph, let alone its application on the recommender system. We explore to bridge these gaps in this section and customize our design for the problem of feature-aware recommendation. In specific, for convolving on a signed hypergraph, we first expand each signed hyper-edge into ordinary ones (i.e., **signed hyper-edge corruption**) and then design a delicate mechanism to pass collaborative information along the derived edges (i.e., **constrained convolution on signed edges**).

4.1.1 Signed Hyper-edge Corruption. In the problem of feature-aware recommendation, if a user positively comments on a feature of an item, we can intuitively infer that: (1) the user may be interested in the item and (2) the item may exhibit high quality on the feature. Thus, for a positive user-item-feature interaction (i.e., a hyper-edge with sign "+1"), we corrupt it into two positive ordinary edges, which connect the user-item and the item-feature, respectively. For example, in Figure 2(b), hyper-edge "**u4-i4-f2**" is expanded as "**u4-i4**" and "**i4-f2**" with positive signs. Similarly, for a "-1" signed hyper-edge, we corrupt it as two negative edges, which is exampled in Figure 2(b) by converting "**u1-i1-f1**" into "**u1-i1**" and "**i1-f1**." Basically, we transform a three-order hyper-edge into two semantic-clear pair-wise interactions. We did not make a direct connection between the user and the feature because: (1) Their relation is not easy to be implied from the hyper-edge

sentiment. Negative label does not necessarily mean the user dislikes the feature; it may also mean the user actually cares about this feature, but the item does not perform well on it. (2) Previous studies [15, 31] manifest that the direct modeling between the user and feature is not significant for the performance improvement in feature-aware recommendation, which is also verified by our experiments in Section 7.3.4.

Careful readers may have found that different hyperedges may produce the same ordinary edge in the corruption process. For example, corrupting "**u1-i1-f1**," "**u1-i1-f2,**" and "**u1-i1-f3**" in Figure 2(a) will all result in a connection between **u1** and **i1**, and different hyperedges support different signs on "**u1-i1**." In specific, "**u1-i1-f1**" and "**u1-i1-f3**" imply positive signs on "**u1-i1**," while "**u1-i1-f2**" suggests a negative correlation. In such a scenario, we determine the final edge sign by a "voting" mechanism; that is, if there are more positive supports, the edge is assigned as "+1," and more negative supports leadto a "−1" labeling. For the special case that there are exactly the same positive and negative supports, it is hard to determine the relationship between the corresponding nodes, and we cut down this edge to avoid any biased judgment that may limit the effectiveness of the following convolutional operation. It should be noted that there can be more advanced methods for resolving collisions, and we left them for future explorations.

4.1.2 Constrained Convolution on Signed Edges. Convolutional operation has been demonstrated to be useful in promoting the recommendation performance, since it can leverage the crucial collaborative information in a more explicit and effective manner [45]. However, traditional graph convolutional network is not applicable for feature-aware recommendation because: (1) **Heterogeneity.** There are three types of nodes with distinct semantics in our problem, and not all paths are reasonable for propagating the crucial collaborative signals. As exampled in Figure 2(c), information can be safely aggregated along path A with the collaborative intuition that "two users who positively comment on the same item are similar." While in path C, although we know "the user likes the item" and "the item performs well on the feature," it is less intuitive to push the feature to the user, since the relations on the path are semantically incompatible, which makes it hard to reveal the collaborative nature. (2) **Signed connection.** Instead of only unsigned edges, the connections in our problem can be both positive and negative. Since the properties vastly differ between differently signed edges [11], a delicate information propagation mechanism is needed for more effective and accurate neighbor aggregation. For filling these gaps, we devise a constrained graph convolution layer for signed edges. We begin by formally defining some key concepts used in our model and then illustrate the principles.

Definition 1 (User-Item-Feature Graph). For a graph $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$, if $\boldsymbol{V} = \mathcal{U} \cup \mathcal{I} \cup \mathcal{F}$, and $\boldsymbol{E} \subseteq \boldsymbol{V} \times \boldsymbol{V}$ defines the set of corrupted edges from some user-item-feature interactions (see Section 4.1.1), then we call $\mathcal{G}$ as a user-item-feature graph. **Example:** Figure 2(c) is an example of user-item-feature graph. We can see there is no edge between the users and features, while an item can be connected with both of them.

Definition 2 (User-Item Subgraph, Item-Feature Subgraph). Let $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$ be a user-item-feature graph. The user-item subgraph $\mathcal{G}^{UI}$ is composed of all the user nodes, item nodes, and their connections. The item-feature subgraph $\mathcal{G}^{IF}$ includes all the item nodes, feature nodes, and the edges between them. **Example:** In Figure 2(c), nodes with solid circles form the item-feature subgraph, and dotted circles label the user-item subgraph.

Definition 3 (User-Item Path, Item-Feature Path). Let $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$ be a user-item-feature graph, for a path $v_1 \rightarrow e_1 \rightarrow v_2 \rightarrow e_2 \rightarrow v_3 \rightarrow \cdots \rightarrow e_{n-1} \rightarrow v_n$ on $\mathcal{G}$, if $v_i \in \mathcal{U} \cup \mathcal{I}, \forall\, i \in [1, n]$, we call this path a user-item path, if $v_i \in \mathcal{I} \cup \mathcal{F}, \forall\, i \in [1, n]$, we call this path an item-feature path. **Example:** In Figure 2(c), A is a user-item path, and B is an item-feature path. Apparently, each path in the

user-item subgraph is a user-item path, and all the user-item paths are contained in the user-item subgraph. The same relations also apply between the item-feature paths and the item-feature subgraph.

Customized information propagation paths. For reasonable CF modeling, once we got a user-item-feature graph $\mathcal{G}$ from Section 4.1.1, we conduct convolution operation on subgraphs $\mathcal{G}^{UI}$ and $\mathcal{G}^{IF}$ separately. This constrains the information propagation to the user-item paths and the item-feature paths, which can more intuitively reveal the collaborative nature, such as "users with similar interacted items may be similar" and "items with similar features may be alike." Formally, the user and feature embeddings on the lth convolutional layer can be derived by aggregating embeddings on the $(l-1)$th layer as:

$$\boldsymbol{s}_i^{(l)} = \sigma\left(\boldsymbol{W}_{x_i}^{(l)}\left[\sum_{j\in\mathcal{N}_i^Y}\frac{\boldsymbol{s}_j^{(l-1)}}{\sqrt{|\mathcal{N}_i^Y||\mathcal{N}_j^Y|}} + \boldsymbol{s}_i^{(l-1)}\right]\right), \tag{2}$$

where i is a node ID, with $x_i \in \{\mathcal{U}, \mathcal{F}\}$ indicating i's node type (a user node or a feature node). $\boldsymbol{W}_{x_i}^{(l)}$ is the weighting parameter. $Y = \begin{cases}\mathcal{G}^{UI} & \text{if } x_i = \mathcal{U}\\ \mathcal{G}^{IF} & \text{if } x_i = \mathcal{F}\end{cases}$, and $\mathcal{N}_i^Y$ is the 1-hop neighbor set of node i on subgraph Y. By recursively applying this formula on $\mathcal{G}^{UI}$ and $\mathcal{G}^{IF}$, the user and feature embeddings are enhanced by assembling their neighbors from multi-hop connections.

For an item, since it can be related with both users and features, we update its embedding from both user-item and item-feature subgraphs, that is:

$$\begin{aligned}\boldsymbol{s}_i^{(l)} = \sigma\Bigg(&\underbrace{\boldsymbol{W}_{\mathcal{U}}^{(l)}\left[\sum_{j\in\mathcal{N}_i^{\mathcal{G}^{UI}}}\frac{\boldsymbol{s}_j^{(l-1)}}{\sqrt{|\mathcal{N}_i^{\mathcal{G}^{UI}}||\mathcal{N}_j^{\mathcal{G}^{UI}}|}} + \boldsymbol{s}_i^{(l-1)}\right]\boldsymbol{A}^{(l)}}_{\text{user-item subgraph}}\\ &+\underbrace{\boldsymbol{W}_{\mathcal{F}}^{(l)}\left[\sum_{j\in\mathcal{N}_i^{\mathcal{G}^{IF}}}\frac{\boldsymbol{s}_j^{(l-1)}}{\sqrt{|\mathcal{N}_i^{\mathcal{G}^{IF}}||\mathcal{N}_j^{\mathcal{G}^{IF}}|}} + \boldsymbol{s}_i^{(l-1)}\right]\boldsymbol{B}^{(l)}}_{\text{item-feature subgraph}}\Bigg),\end{aligned} \tag{3}$$

where the first under-braced part represents the information propagated from the user-item subgraph, while the second one comes from the item-feature subgraph. $\boldsymbol{A}^{(l)}$ and $\boldsymbol{B}^{(l)}$ are adapting parameters projecting these two parts into the same space.

Dual-embedding mechanism. Above, we customize the convolutional paths with constraints inspired from the basic assumption of collaborative filtering. Another challenge in our problem stems from the signed edges. In traditional unsigned graph, connected nodes are presumed to be more similar than those without links, and the signals delivering across different paths are consistent and homologous. While when edges can be both positive and negative, the propagated information becomes more complex, which may reflect extremely different semantics. As exampled in Figure 2(d), path X implies a higher similarity between user **u3** and **u1**, since they both positively interact with the same item. While according to path Y, neighbor **u2** may stand on the opposite side of user **u1**, reflecting her reverse preference, which is evidenced in their different attitudes towards item **i2**. As a result, nodes **u3** and **u2** reveal user **u1**'s properties from different perspectives, which should be tackled separately.

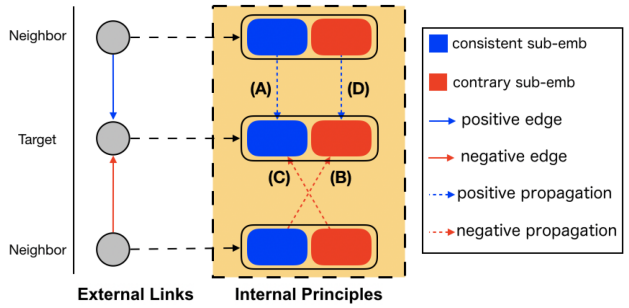

Fig. 3. Principle of our dual-embedding mechanism for propagating collaborative information on signed edges.

To handle such signed graph, we can straightforwardly convert it into an unsigned one by either dropping the negative edges or treating them the same as positive links. However, each of these methods has significant weaknesses. On one hand, negative connections contain parts of the graph structure information, and ignoring them may lead to insufficient message passing and limit the neighbor aggregation effectiveness. On the other hand, the positive and negative links have totally different meanings and functions; it is unreasonable to trivially equalize them for propagating the crucial collaborative information.

To overcome these shortcomings, we adopt a dual-embedding mechanism for coherently handling signed edges. As mentioned before, different signed paths may result in various relationships between the end nodes. Traditional unified embedding scheme is less effective in such a scenario, since it is hard to distinguish semantically different information in a single latent vector, and there is no tailored aggregation strategy for different path semantics. In our model, rather than keeping a single representation for each node, we maintain two sub-embeddings for taming different path semantics in a more explicit and finer-grained manner. The principle of our dual-embedding scheme is illustrated in Figure 3. Each node is represented by a pair of consistent and contrary sub-embeddings. The consistent sub-embedding aims to encode information that is similar to the node, while the contrary one is used to capture the reverse properties. A target node can be connected with its neighbor by a positive or negative edge. With the collaborative filtering assumption, the positive edge keeps the semantic of the information passing through it, while the negative edge reverses the relation between the end nodes. As a result, the target node's consistent sub-embedding is achieved by passing the neighbor's consistent sub-embedding through a positive edge (**A**) or contrary sub-embedding from a negative edge (**C**). Similarly, the neighbor's consistent sub-embedding with a negative edge (**B**) or contrary sub-embedding with a positive edge (**D**) contributes the target node's contrary sub-embedding.

Formally, the sub-embeddings for a user or a feature at the first convolutional layer (i.e., $l = 1$) are computed by revising Equation (2) as:

$$\begin{aligned} \boldsymbol{s}_i^{P(1)} &= \sigma\left(\boldsymbol{W}_{x_i}^{P(1)}\left[\sum_{j\in\mathcal{N}_i^{Y+}}\frac{\boldsymbol{s}_j^0}{\sqrt{|\mathcal{N}_i^{Y+}||\mathcal{N}_j^{Y+}|}}+\boldsymbol{s}_i^0\right]\right) \\ \boldsymbol{s}_i^{N(1)} &= \sigma\left(\boldsymbol{W}_{x_i}^{N(1)}\left[\sum_{j\in\mathcal{N}_i^{Y-}}\frac{\boldsymbol{s}_j^0}{\sqrt{|\mathcal{N}_i^{Y-}||\mathcal{N}_j^{Y-}|}}+\boldsymbol{s}_i^0\right]\right), \end{aligned} \tag{4}$$

where $\boldsymbol{s}_j^0 \in \mathbb{R}^{d_0}$ is the initial node embedding. i and x_i indicate the node ID and type as defined above. $\boldsymbol{W}_{x_i}^{P(1)} \in \mathbb{R}^{d_1 \times d_0}$ and $\boldsymbol{W}_{x_i}^{N(1)} \in \mathbb{R}^{d_1 \times d_0}$ are weighting parameters. Y discriminates different subgraphs as mentioned in Equation (2). $\mathcal{N}_i^{Y+}$ and $\mathcal{N}_i^{Y-}$ are the sets of positive and negative neighbors of node i in subgraph Y, respectively. In this equation, $\boldsymbol{s}_i^{P(1)}$ aggregates information that is similar to i, while $\boldsymbol{s}_i^{N(1)}$ encodes i's opposite properties. We can compute the first layer item embeddings by modifying Equation (3) in a similar manner.

For multiple convolutional layers, the high-level representations are determined by the lower ones in a dynamic programming manner. Specifically, the sub-embeddings of a user or feature node at the lth convolutional layer ($l > 1$) are computed by:

$$\begin{aligned}
\boldsymbol{s}_i^{P(l)} &= \sigma\left(\boldsymbol{W}_{x_i}^{P(l)}\left[\underbrace{\sum_{j\in\mathcal{N}_i^{Y+}} \frac{\boldsymbol{s}_j^{P(l-1)}}{\sqrt{|\mathcal{N}_i^{Y+}||\mathcal{N}_j^{Y+}|}}}_{A} + \underbrace{\sum_{j\in\mathcal{N}_i^{Y-}} \frac{\boldsymbol{s}_j^{N(l-1)}}{\sqrt{|\mathcal{N}_i^{Y-}||\mathcal{N}_j^{Y-}|}}}_{B} + \boldsymbol{s}_i^{P(l-1)}\right]\right) \\
\boldsymbol{s}_i^{N(l)} &= \sigma\left(\boldsymbol{W}_{x_i}^{N(l)}\left[\underbrace{\sum_{j\in\mathcal{N}_i^{Y+}} \frac{\boldsymbol{s}_j^{N(l-1)}}{\sqrt{|\mathcal{N}_i^{Y+}||\mathcal{N}_j^{Y+}|}}}_{C} + \underbrace{\sum_{j\in\mathcal{N}_i^{Y-}} \frac{\boldsymbol{s}_j^{P(l-1)}}{\sqrt{|\mathcal{N}_i^{Y-}||\mathcal{N}_j^{Y-}|}}}_{D} + \boldsymbol{s}_i^{N(l-1)}\right]\right),
\end{aligned} \tag{5}$$

where $\boldsymbol{W}_{x_i}^{P(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ and $\boldsymbol{W}_{x_i}^{N(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ are weighting parameters. The consistent sub-embedding $\boldsymbol{s}_i^{P(l)}$ aggregates information based on **A**: the consistent sub-embeddings of i's 1-hop positive neighbors (i.e., $j \in \mathcal{N}_i^{Y+}$), and **B**: the contrary sub-embeddings of i's 1-hop negative neighbors (i.e., $j \in \mathcal{N}_i^{Y-}$). While the contrary sub-embedding $\boldsymbol{s}_i^{N(l)}$ is composed of **C**: the contrary sub-embeddings of i's 1-hop positive neighbors, and **D**: the consistent sub-embeddings of i's 1-hop negative neighbors. The final embedding of node i at the lth layer concatenates $\boldsymbol{s}_i^{P(l)}$ and $\boldsymbol{s}_i^{N(l)}$ (i.e., $[\boldsymbol{s}_i^{P(l)}, \boldsymbol{s}_i^{N(l)}]$) together. Essentially, we use two aggregators to model different path semantics brought by the signed edges, and the collaborative information is propagated by recursively updating the consistent and contrary sub-embeddings in a finer-grained manner. The computational rules for the lth layer item sub-embeddings can be derived by extending Equation (3) in a similar way.

4.2 Prediction and Optimization

4.2.1 Neural Hyper-link Prediction. User preference usually exhibits complex and diverse properties in real-world scenarios. Unlike previous methods [15, 33, 48], which assume linear relationship between the user, item, and feature, we do not impose any constraint on the entity interactions. In our method, a deep architecture is leveraged to capture more flexible correlations in the dataset, and the neural modeling is expected to lower the risk of model misspecification. Formally, suppose the convolutional enhanced embeddings for user u, item i, and feature f are $\boldsymbol{s}_u$, $\boldsymbol{s}_i$, and $\boldsymbol{s}_f$, respectively, we predict the label of a hyper-edge by: $\hat{y}_{uif} = \phi_K(\phi_{K-1}(\ldots\phi_1([\boldsymbol{s}_u; \boldsymbol{s}_i; \boldsymbol{s}_f])))$, where $[\cdot;\cdot;\cdot]$ concatenates the input vectors and $\{\phi_1, \phi_2, \ldots, \phi_K\}$ are non-linear layers with sigmoid as the active function.

4.2.2 Optimization. We use binary cross-entropy to optimize model parameters, which has been intensively adopted in the recommender system [9, 16]. Basically, it aims to maximize the prediction of positive hyper-edges and simultaneously minimize the negative ones. The objective

function is:

$$\begin{aligned}\mathcal{L}_1 &= \sum_{(u,i,f)\in O} \Big(y_{uif} \ln \sigma(\hat{y}_{uif}) + (1 - y_{uif}) \ln \sigma(1 - \hat{y}_{uif})\Big) \\ &= \sum_{(u,i)\in\mathcal{T}} \left(\sum_{f\in\mathcal{T}_{ui}^{+}} \ln \sigma(\hat{y}_{uif}) + \sum_{f\in\mathcal{T}_{ui}^{-}} \ln \sigma(1 - \hat{y}_{uif}) \right),\end{aligned} \tag{6}$$

where O is the user-item-feature interaction set. y_{uif} is the ground truth, which is 1 for positive interactions (i.e., user u mentioned feature f with positive sentiment in her review on item i), and 0 for others.[3] $\mathcal{T}$ is the observed user-item interaction set. $\mathcal{T}_{ui}^{+}$ and $\mathcal{T}_{ui}^{-}$ are the feature sets mentioned in user u's review on item i with positive and negative sentiments, respectively.

Recommender system essentially aims to compute the matching degree between a user and an item, thus, we explicitly incorporate the direct user-item interaction loss into the final objective. Specifically, the overall user/item representation is composed of two parts: **Review-related embedding**: We leverage the above $\boldsymbol{s}_u$ and $\boldsymbol{s}_i$ to represent the user/item properties reflected in the review information. **Review-irrelevant embedding**: User preferences and item properties can be diverse and complex, and it is intuitive that many characters are not involved in the review information. Thus, we incorporate an auxiliary embedding[4] to capture such information, which can make the model more comprehensive and robust. The overall user/item representation concatenates the above two parts, i.e., $\boldsymbol{s}_x^* = [\boldsymbol{s}_x; \boldsymbol{s}_x^{aux}] \in \mathbb{R}^D$ $(x = u, i)$. At last, the matching score between user u and item i is computed as: $\hat{t}_{ui} = \boldsymbol{s}_u^* \cdot \boldsymbol{s}_i^*$, and the implicit interactions are modeled by maximizing:

$$\mathcal{L}_2 = \sum_{(u,i,j)\in\mathcal{P}} \ln \sigma(\hat{t}_{ui} - \hat{t}_{uj}), \tag{7}$$

where $\mathcal{P} = \{(u, i, j) | (u, i) \in \mathcal{T}, (u, j) \notin \mathcal{T}\}$ is the set of pair-wise training data, and j is randomly sampled from u's un-interacted items.

By combining the above components, the final objective function is:

$$\mathcal{L} = (1 - \alpha)\mathcal{L}_1 + \alpha\mathcal{L}_2 - \lambda\mathcal{L}_{\text{reg}}, \tag{8}$$

where α balances the weights between different optimization parts. $\mathcal{L}_{\text{reg}} = \sum_{\theta\in\Theta} \|\theta\|_2^2$ is the regularization term with Θ as the parameter set. In this optimization target, $\mathcal{L}_1$ models the user-item-feature triplet as whole, and the nodes' compatibility is predicted without separating the hyper-edge pair-wisely. $\mathcal{L}_2$ considers the direct user-item interactions. $\mathcal{L}_1$ and $\mathcal{L}_2$ share the same parameters of s_u and s_i, which makes our model basically a multi-task learning framework.

5 FURTHER ANALYSIS

For better understanding our designed model, we present some analysis in the following:

Comparison with previous methods. Unlike existing feature-aware recommender models [15, 33, 48], where the user, item, and feature embeddings are produced directly from the ID information, we leverage graph convolutional operation to enhance the embedding process. The specifically designed aggregation paths can help to more sufficiently capture the collaborative filtering signals. In addition, instead of imposing unpractical linear assumption on the entity relations, we use a neural network to more flexibly learn the correlations from the data, which greatly improves the user profiling accuracy as well as the recommendation results.

[3] Note that the ground truth of a hyper-link with sign "−1" is regarded as 0.

[4] In the optimization process, the auxiliary embedding is randomly initialized and learned based on stochastic gradient descent.

Time complexity analysis. Here, we present a complexity analysis on our model's computational cost in the testing phase. As mentioned above, our model is composed of the embedding and prediction components. In the embedding process, suppose the largest node degree in our graph is M, then the complexity of aggregating neighbors on the lth layer mainly depends on the matrix multiplications in Equations (4) and (5), which costs $O(N * M * d_l * d_{l-1})$, where N is the number of the nodes. By convolving L layers, the total complexity of the embedding process takes $O(\sum_{l=1}^{L} N * M * d_l * d_{l-1})$. For the prediction part, let the kth user-item-feature prediction layer ϕ_k project $\mathbb{R}^{t_{k-1}}$ to $\mathbb{R}^{t_k}$, then the total complexity for the triplet predictions is $O(|\mathcal{W}| * \sum_{l=1}^{K} t_k * t_{k-1})$, where $|\mathcal{W}|$ is the number of user-item-feature interactions. Since the user-item prediction layer (i.e., Equation (7)) only depends on linear operations, the complexity is simply $O(|\mathcal{T}| * D)$ with $|\mathcal{T}|$ as the user-item interaction number. As a result, the total complexity of our model is $O(\sum_{l=1}^{L} N * M * d_l * d_{l-1} + |\mathcal{W}| * \sum_{l=1}^{K} t_k * t_{k-1} + |\mathcal{T}| * D)$.

6 RELATED WORK

Our study is a combination between review-based recommender system and GCN. We review the relevant research areas in the following.

6.1 Review-based Recommendation

Leveraging side information to enhance the recommendation performance [7, 20, 34, 46, 47] and explainability [21, 42, 44] has attracted increasing attention in the recent years. The user reviews, as an important side information, have been widely studied. Early models [25, 29] mostly based the review modeling on topic models [3], and the studies mainly concentrated on how to connect the topic distributions with the user/item representations. With the ever prospering of the representation learning technology, people have made several explorations [27, 30, 51] on processing user reviews based on deep architectures. Comparing with topic models, deep learning methods can more effectively represent the semantics of the user reviews [51]. For example, Reference [39] leveraged attention mechanism to extract important features from the review information and enhance the user, item representation by these features. Reference [22] designed two networks to transfer the knowledge between user/item embeddings and the review information. Reference [19] formulated review-based recommendation with capsule network and the recommendation results can be explainable. Reference [6] profiled the users or the items by aggregating all the review information, where the model can not only predict the results but also can generate the most useful reviews for the target users. Reference [10] extended Reference [6] by modeling users' dynamic preferences.

These methods mostly process the review information on the document-level. However, user reviews are usually quite noisy, so indiscriminately integrating all the contents may be inappropriate. In another research line, people leverage the review information by extracting user feature-level preference, which utilizes user reviews in a more clear manner. In specific, Reference [15] designed a TriRank method for modeling user-item-feature ternary relationships. Reference [48] leveraged a co-matrix factorization model to capture pair-wise relations among the users, items, and features. Reference [8] further extended Reference [48] by optimizing a feature-aware ranking loss. Reference [33] utilized tensor factorization method to incorporate features into the user-item interaction modeling. Different from these algorithms, our model formulates review-based recommendation with a hypergraph network, which enables us to enhance the embedding process by the convolutional operation. In addition, our model no longer depends the relation modeling on the linear assumption, and we leverage neural network to capture more complex interaction patterns.

6.2 Recommendation Based on Graph Neural Network

GNN has been successfully applied in many applications, such as social network [11, 12] and neural language processing [4]. To explore the effectiveness of GNN for recommender system, people have built many promising models [2, 12, 35, 40, 41, 52]. References [2, 35] equipped user-item bipartite graph with convolutional operation. To deploy graph neural network upon industry environment, Reference [45] designed an efficient graph network to propagate information along the item-item relations. Different from other methods, where the edge sign information is ignored in the modeling process, Reference [32] proposed a model to estimate signed links in heterogeneous network. In this article, we apply GNN to the problem of feature-aware recommendation, and the main focus of our model is on the signed ternary relations.

6.3 Hypergraph Neural Network

Hypergraph is a natural extension of traditional graph; it aims to model high-order correlations among the data. Recent years have witnessed many efforts on building neural network on the hypergraph structures. Reference [13] designs a convolutional operation on hyperedges to capture the correlations during representation learning. Reference [43] degenerates the hyperedge into many pairwise edges and treats the learning problem as a traditional graph optimization problem. To capture different importances of the nodes, Reference [1] designs a hypergraph convolution layer based on attention mechanism for aggregating. In our model, we apply hypergraph to the field of feature-based recommender system, which is shown effective in capturing complex user-item-feature interaction patterns.

7 EXPERIMENTS

In this section, we present our experiments, mainly focusing on three research questions:

- **Q1:** Can our model achieve better performance than the state-of-the-art methods?
- **Q2:** What are the effects of our designed convolutional operation?
- **Q3:** How do different parameters influence our model's performance?

In the following, we begin by introducing the experimental setup and then answer these questions by analyzing the experimental results.

7.1 Experimental Setup

7.1.1 Dataset. As mentioned above, we leverage two real-world datasets—**Amazon**[5] and **Yelp**[6]—for model evaluation. In specific, **Amazon** contains a large number of user review and interaction records, and we select five categories, including Automotive, Instant Video, Baby, Digital Music, and Office for algorithm comparison. From the data statistics in Table 1, we can see the selected datasets cover different characters, e.g., Automotive is a smaller and denser dataset, while Baby is relatively larger, but much more sparse. **Yelp** is also a review-based dataset, but the review contents are about restaurants instead of e-commerce products, which may exhibit different user preference patterns. Since the original data are quite large, we filter it by remaining the users with at least 20 item interactions.

Feature extraction. In our datasets, each piece of user review is transformed into many "(**U**, **I**, **F**, **S**)" quadruplets, where **U** and **I** indicate the user and item related to the review, **F** represents some item feature (e.g., image quality and battery life in Figure 1), and **S** is the emotional polarity from the user to the feature. Following previous studies [15, 33, 48], we extract the features

[5] http://jmcauley.ucsd.edu/data/amazon/.
[6] https://www.yelp.com/dataset/download.

Table 1. Statistics of the Datasets Used in Our Experiments

Dataset	#User	#Item	#Feature	#Interaction	Density
Automotive	1,863	1,340	54	3,520	0.141%
Video	4,099	1,449	142	10,891	0.183%
Office	4,784	2,390	463	34,619	0.302%
Digital	5,394	3,559	584	48,376	0.252%
Baby	19,067	7,008	567	91,184	0.068%
Yelp	10,044	19,091	596	478,146	0.249%

Table 2. Five Most Frequent Features in Each Dataset

Dataset	Top Features (**F**)
Amazon-Auto	car, quality, product, fit, price
Amazon-Video	season, characters, series, acting, episode
Amazon-Office	printer, quality, paper, product, colors
Amazon-Digital	album, song, sound, track, music
Amazon-Baby	baby, size, month, seat, product
Yelp	food, service, staff, taste, chicken

(**F**) and their corresponding sentiments (**S**) based on an open-sourced toolkit called "Sentires"[7] [49]. In general, this tool is a rule-based system. The features (**F**) for each product category are extracted by analyzing the review grammar and morphology, and the sentiment polarities (**S**) are determined by an optimization framework. Since the feature and sentiment extraction is not the focus of this article, we refer readers to References [15, 48] for more detailed illustration.

For an intuitive understanding of our data, we present the most frequently mentioned features for each dataset in Table 2. We can see, while most of the extracted features are meaningful, many of them are just general words instead of item-specific attributes, such as, "food" for Yelp and "baby" for Amazon-Baby. In addition, there are also many noisy features (e.g., "4 × 6" for Amazon-Office) produced from this toolkit. Since these improper features may influence the downstream task (i.e., feature-aware recommendation), and it is not easy to filter them automatically, we remove them in a crowdsourcing manner. In specific, three human annotators were asked to judge whether a feature is proper for the corresponding dataset, and a feature is remained only when more than two people make positive feedback.

7.1.2 Baselines. We select the following representative methods as our baselines:

- **Most Popular (MP) [17]:** This is a non-personalized method, and the recently most popular items are recommended to the users.
- **BPR [26]:** This is a well-known recommender method for modeling user implicit feedback; we use matrix factorization as its predictive function.
- **NCF [16]:** This is a state-of-the-art deep recommender model, where the user and item representations are fed into multiple non-linear layers to predict the final targets.
- **NGCF [35]:** This is a collaborative filtering method based on graph convolution network. The collaborative information is propagated on the user-item bipartite graph.
- **MCCF [37]:** This is a recently proposed graph recommendation method based on attention mechanism.

[7] http://yongfeng.me/code/.

- **HFT [25]:** This is a well-known review-based recommendation method, where the review contents are modeled by a topic model.
- **EFM [48]:** This is a well-known feature-aware recommender model, where the user, item, and feature relations are captured by linear co-matrix factorization technique.
- **LRPPM [8]:** This is a feature-aware recommender model based on ranking objective functions; the user, item, and feature correlations are predicted by tensor factorization.
- **DeepCoNN [51]:** This is a review-based recommender model, and it is the first algorithm that leverages deep learning method to capture user review semantics.

These baselines cover different model characters. BPR, NCF, NGCF, and MCCF are ID-based recommender methods with either shallow or deep architectures. EFM, LRPPM, HFT, and DeepCoNN are review-enhanced models, where HFT, DeepCoNN processes the review information on the document-level, while EFM and LRPPM are designed based on user feature-level preferences. Here, we do not use MTER [33] as a baseline, since it aims to study the effectiveness of user opinion information that is unavailable in our model as well as the other baselines, such as EFM and LRPPM. Because HFT, MCCF, and DeepCoNN are designed for rating prediction, we revise their objective functions into BPR ranking loss [26] to model user-implicit feedback.

7.1.3 Evaluation and Implementation Details. In our experiments, each user's latest 30% interactions are used for model testing, while the remaining are left for training (60% interactions) and validation (10% interactions). The widely used metrics, including Hit Ratio (HR), F_1 and Normalized Discounted Cumulative Gain (NDCG) are utilized for performance evaluation. Among these metrics, HR and F_1 aim to measure how the recommended items overlap with users' actually purchased ones, while NDCG is a ranking-based measurement, and higher-ranked accurate items contribute more to the final results. In our experiment, we recommend 10 items for each user, and the metrics are separately computed for each user. The final result is reported by averaging all the testing users. When implementing our model, the trainable parameters are initialized according to a normal distribution and updated by stochastic gradient descent (SGD) in the later optimization process. The hyper parameters are tuned based on the validation set in a grid search manner. More specifically, the embedding dimension d_0 is tuned in the range of $[10, 30, 50, 70, 90, 150, 200, 250]$. The learning rate and the batch size are determined in $[0.001, 0.005, 0.01, 0.05, 0.1]$ and $[8, 16, 32, 64]$, respectively. We search the weighting parameter α from $[0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1.0]$. The auxiliary embedding size in Equation (7) is tuned in $[10, 30, 50, 70, 90]$, and the regularization coefficient λ is selected from $[0.00001, 0.0001, 0.001, 0.01]$.

7.2 Overall Comparison (Q1)

The results of comparing our model with the baselines can be seen in Table 3. We can see: (i) While MP is a simple method, it sometimes performs well (e.g., on the datasets of Automotive and Baby). This phenomenon is also observed in a recent study [17]. Among the other baselines, NCF, NGCF, and MCCF achieve better performance than BPR in most cases. Basically, BPR captures user-item correlations based on the linear inner product. While NCF depends its prediction on a more powerful neural network, and NGCF equips the embedding process with convolutional operation to more sufficiently leverage CF signals [35]. As a result, both NCF and NGCF are capable of discovering more complex user behavior patterns and produce better recommendation results. (ii) Among the review-enhanced models, EFM and LRPPM perform better than HFT. The reasons can be that: HFT blindly leverages all the review information for user or item profiling. Many irrelevant review contents are introduced into the learning process, which may overwhelm

Table 3. Performance Comparison between the Baselines and Our Model

Dataset	@10	MP	BPR	NCF	NGCF	MCCF	HFT	EFM	LRPPM	DeepCoNN	SHCN
Automotive	F_1	3.282	3.030	3.331	3.734	3.813	3.121	3.261	3.311	4.233*	**4.996**
	HR	18.05	16.67	18.13	22.83	23.09	17.01	18.06	19.06	24.61*	**27.77**
	NDCG	9.773	7.403	8.112	11.91	12.23	7.015	7.115	8.335	**12.78***	12.65
Video	F_1	5.538	6.593	6.881	6.985	7.021	6.632	6.863	6.661	7.304*	**7.449**
	HR	30.65	35.88	37.11	37.50	38.11	36.34	37.90	35.66	39.11*	**40.32**
	NDCG	15.72	16.45	18.91	19.16	19.76	17.44	18.13	17.16	19.63*	**21.62**
Office	F_1	1.267	4.165	4.671	4.802	4.791	4.442	4.745	4.667	4.899*	**5.755**
	HR	7.059	20.34	24.13	25.88*	25.02	21.70	23.06	23.18	25.32	**31.17**
	NDCG	3.871	10.03	10.98	11.02	10.96	10.91	10.34	10.08	11.21*	**15.30**
Digital Music	F_1	1.404	7.220	8.123	9.580*	9.167	7.011	7.281	8.218	8.485	**10.05**
	HR	8.000	37.14	40.93	49.71*	48.04	37.42	39.71	40.97	41.14	**52.57**
	NDCG	3.311	17.68	23.18	25.75*	24.99	18.39	18.72	21.01	23.32	**27.08**
Baby	F_1	5.415	4.949	4.771	4.804	4.984	5.413	5.633	5.513	5.946*	**6.683**
	HR	29.62	28.27	25.17	26.24	27.32	28.24	30.74	29.04	32.33*	**36.45**
	NDCG	13.22	11.95	11.09	12.83	13.03	13.10	14.33	13.11	17.33*	**19.10**
Yelp	F_1	4.655	11.98	14.21	15.02	14.91	13.16	14.16	14.26	16.11*	**17.09**
	HR	26.61	66.42	69.01	71.72*	70.01	65.12	27.12	68.23	69.12	**72.62**
	NDCG	7.454	26.00	28.91	30.33	29.10	27.12	28.88	29.98	31.22*	**34.40**

All the numbers are percentage value with "%" omitted. For each metric on different datasets, we use bolded fonts and * to label the best performance and the best baseline performance, respectively.

useful information and limit the recommendation performance. In contrast, EFM and LRPPM only remain user feature-level preferences, which profile users and items in a more clean and structured manner. Interestingly, DeepCoNN obtains better performance than EFM and LRPPM, we speculate that the powerful deep architecture can compensate the weakness brought by the noisy review information, which manifests that non-linear architecture is quite important for user review modeling. (iii) Encouragingly, our model achieves the best performance in most cases. This observation positively answers the first research question (**Q1**) and verifies the effectiveness of our designed architecture. Since the review information can more comprehensively profile the users and items, SHCN achieves better performance than review-free models, such as NCF and NGCF. Comparing with previous review-based methods (e.g., EFM, LRPPM), SHCN can enhance the user/item/feature embeddings with their neighbor information, which finally leads to improved results.

7.3 Study on the Convolutional Operation (Q2)

In this section, we investigate the effects of the convolutional layers. The parameters are tuned in the ranges as mentioned above. We present the performances of HR and NDCG on the Amazon datasets, and the results on F_1 and Yelp are similar and omitted.

7.3.1 Effects of the Number of Convolutional Layers. The number of convolutional layers determines how many hops the information can be propagated in the graph. We run our model by setting the layer number L as $\{0, 1, 2, 3, 4\}$, respectively.[8] The results are shown in Table 4. We can see: convolutional operation is indeed useful for our task, since SHCN-0 ($L = 0$) exhibits the worst performance across all the datasets. The best performance is achieved when $L = 2$ or $L = 3$, too small or large layer number leads to inferior results. We speculate that increasing L can enhance

[8] $L = 0$ indicates no convolutional layers.

Table 4. Effects of the Number of Convolutional Layers

Dataset	Automotive		Video		Office		Digital Music		Baby	
@10(%)	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
SHCN-0	13.70	6.54	16.67	7.22	20.59	9.76	50.86	26.37	7.31	3.13
SHCN-1	23.61	11.71	36.29	19.03	25.29	10.08	51.42	26.74	26.71	12.02
SHCN-2	**27.77**	**12.65**	**40.32**	**21.62**	**31.17**	**15.30**	**52.57**	**27.08**	36.45	19.10
SHCN-3	25.00	11.57	40.29	20.37	21.76	9.06	52.00	26.89	**37.01**	**19.42**
SHCN-4	23.61	11.09	40.30	21.08	26.47	12.49	50.28	26.37	35.99	18.88

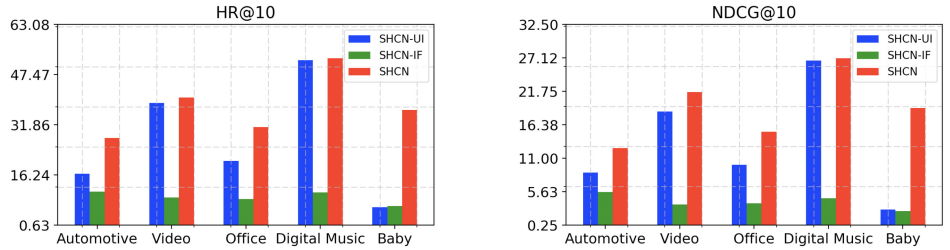

Fig. 4. Comparison between different propagation paths.

the utilization of CF signals, while with the aggregation path becoming longer, there are also more opportunities to introduce noises into the modeling process [35, 36].

7.3.2 Effects of Different Propagation Paths. There are both user-item and item-feature convolutional paths in our model; we respectively investigate their effects for the performance in this section. In the experiments, we conduct convolutional operation along only one type of these propagation paths (we name them as SHCN-UI and SHCN-IF, respectively), and their comparison with our final model can be seen in Figure 4. We can see, SHCN-UI performs better than SHCN-IF across different datasets in most cases. We speculate that Top-N recommendation is more relevant with the user-item interactions, thus the convolutional operation between the users and items may play a more significant role on the final evaluation metrics. While only using item-feature convolutional path does not perform well, it is also valuable in the modeling process, which can be seen from the superior performance of our final model SHCN. This observation manifests that different propagation paths may have their own contributions to the final results; our model can effectively combine them for further improving the performance.

7.3.3 Effects of the Dual-embedding Mechanism. In this experiment, we would like to study whether the dual-embedding mechanism can bring us with superior performance. We compare our model with its two variations. The first method directly drops the negative edges, and the information is only propagated along the positive ones (named as SHCN-P). For the second model, we do not distinguish positive and negative edges, and the convolutional operation is conducted based on a unified embedding mechanism (named as SHCN-U). The results of comparing these methods are shown in Figure 5. We can see: The performance ranking between SHCN-P and SHCN-U interchanges across different datasets (i.e., SHCN-P is better for Video and Baby, and SHCN-U wins on Automotive, Office, and Digital Music), while our model can consistently outperform both of them, which verifies the effectiveness of our designed dual-embedding scheme. We argue that, in real-world scenarios, negative sentiments are useful in revealing user preference, blindly dropping them, like SHCN-P, may lose some valuable information and limit

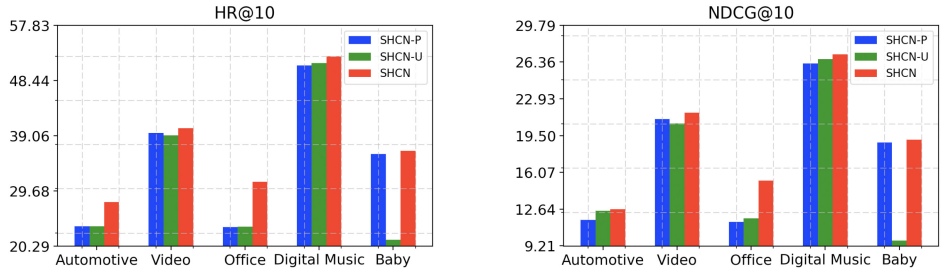

Fig. 5. Effectiveness of the dual-embedding mechanism.

Table 5. Effects of Different Hyper-edge Corruption Methods

Dataset	Automotive		Video		Office		Digital Music		Baby	
@10(%)	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
SHCN-A	25.00	**12.83**	38.71	21.34	22.94	11.27	51.42	26.14	33.74	18.40
SHCN-B	20.83	10.11	38.30	19.92	16.47	8.16	49.14	26.55	30.92	15.97
SHCN-C	26.39	11.33	39.51	20.13	22.35	10.91	50.86	26.74	33.08	17.41
SHCN	**27.77**	12.65	**40.32**	**21.62**	**31.17**	**15.30**	**52.57**	**27.08**	**36.45**	**19.10**

the recommendation performance. Even if we had incorporated all the sentimental information, positive and negative edges still vastly differ in semantics, and it is unreasonable to ignore their discriminations, which is verified by the inferior performance of SHCN-U. Our proposed sub-embeddings can encode different sentiments separately, and the carefully designed aggregation strategy directly handles the signed edges without relaxing the problem like SHCN-U, which utilizes user sentimental preferences in a more effective manner.

7.3.4 Effects of Different Hyper-edge Corruption Methods. Careful readers may also be interested in whether our hyper-edge extending method is reasonable. To answer this question, we compare our model with three other types of corruption strategies. Obviously, each user-item-feature hyper-edge can be potentially extended into three ordinary edges, which connect the user-item, item-feature, and user-feature, respectively. In our experiment, the first comparing method makes connections between the user-item and user-feature (SHCN-A), the second one builds edges for the user-feature and item-feature (SHCN-B), and the last one connects each pair of the user, item and feature (SHCN-C). Similar to SHCN, information in the baselines can only be propagated along the same type of relations. The comparison results are shown in Table 5. We can see, in most cases, our model can outperform the other variations, which verifies the superiority of our hyper-edge corruption method. One interesting observation is that SHCN-C does not perform better than our model. Since SHCN-C only adds user-feature connections based on SHCN, it suggests that the direct modeling between the users and features maybe not be important for our data, which is consistent with our intuitions in Section 4.1.1 and agrees with the results observed in the previous study [15].

7.4 Parameter Analysis (Q3)

In this section, we analyze how different parameters influence our model's performance. We begin by studying the impact of the embedding dimension d_0, and then the weighting parameter α is investigated to show its effects. In the experiments, d_0 and α are tuned in the ranges as mentioned

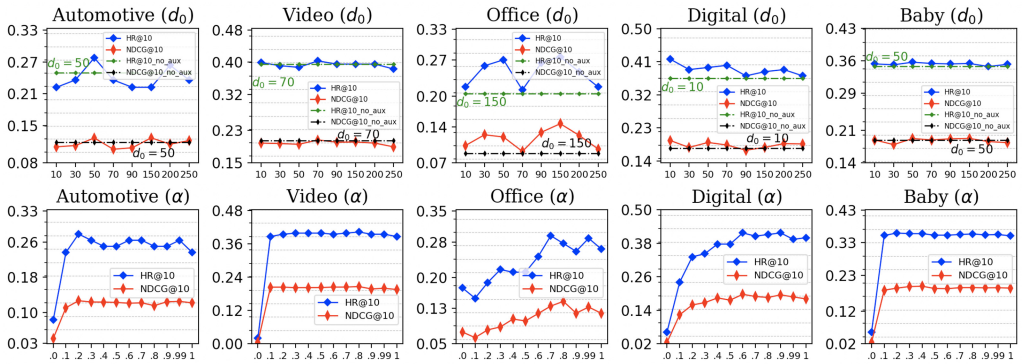

Fig. 6. Parameter analysis. The first line shows the effects of different embedding size d_0. We also present the performance of our model without using the auxiliary embedding by fixing d_0 as the optimal value. The second line shows the impact of the weighting parameter α.

above, and the other parameters are fixed as default values. The results are reported based on the same datasets as Section 7.3.

7.4.1 Influence of the Embedding Dimension d_0. Embedding size is important for the representation learning framework, since it directly determines the model expressiveness. The impact of this parameter for our model is shown in the first line of Figure 6. We can see: In most cases, our model can almost reach the best performance when d_0 is moderate or even very small. This observation is interesting: It manifests that a more powerful model does not necessarily lead to better results. We speculate that, in the field of feature-aware recommendation, user preference data are usually very sparse, and little parameters are enough for fitting the training set; the redundant dimensions make the model more flexible and hard to learn, which limits the model generalization capability on the testing set. In addition, when d_0 is optimal, abandoning auxiliary embedding leads to inferior performance (labeled by dotted lines), which manifests that explicitly modeling the review-irrelevant information is helpful in our model.

7.4.2 Influence of the Weighting Parameter α. Our objective function (i.e., Equation (8)) is composed of two major components (i.e., $\mathcal{L}_1$ and $\mathcal{L}_2$), and the weighting parameter α determines how to balance them in the learning process. To study the influence of α, we examine the model performance by tuning α in the range of $[0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1.0]$. From the result in Figure 6, we can see: The optimal α varies across different datasets. For Automotive and Baby, small α leads to better performance, while on the datasets of Video, Office, and Digital, larger α can be more appropriate. By setting $\alpha = 0.0$ or $\alpha = 1.0$, we actually only optimize $\mathcal{L}_1$ or $\mathcal{L}_2$ in the training process. We can see neither of them can achieve the best performance, which manifests that both optimization goals are useful for the final results. Essentially, our objective function is a multi-task learning framework, and the shared parameters are forced to learn from both user item-level ($\mathcal{L}_2$) and feature-level ($\mathcal{L}_1$) preferences. The user-item modeling is more general and direct to the final task. The feature interactions can reveal more detail and comprehensive user personalities (i.e., different users may have individual feature preferences even for the same item). By taking the best of both worlds, our model finally obtains superior recommendation performance.

7.5 Visualization

In this section, we provide some intuitive understandings for our designed convolutional operation. We based our experiment on the Baby dataset, and the comparison was conducted between

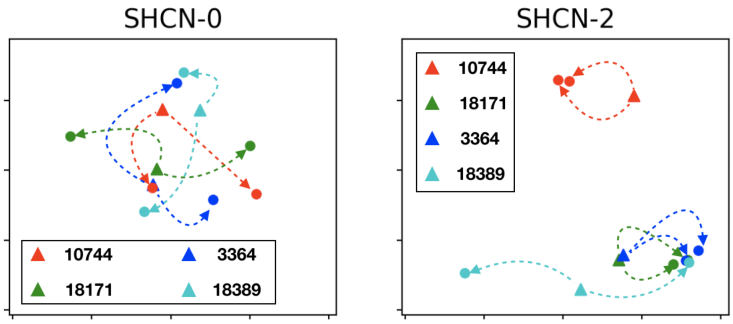

Fig. 7. Visualizing the effect of the convolutional operation. The left figure is our model without any convolutional operation, and the right one is enhanced by two convolutional layers. Each triangle represents a user, and the dotted lines link the users to their interacted items, which are labeled by circles.

our model without any convolutional operation (i.e., SHCN-0) and the one with two convolutional layers[9] (i.e., SHCN-2). We projected the learned embeddings into a 2D space based on t-SNE [24], and four users as well as their interactions in the testing set are presented for analysis. The results are shown in Figure 7, we can see, comparing with SHCN-0, the user embeddings learned from SHCN-2 are more closer to their purchased items' embeddings, which means the convolutional enhanced embeddings can more accurately reflect unseen user-item connectivities. This observation is consistent with Table 4, which, from the qualitative perspective, explains the superiority of the convolutional operation. By explicitly propagating collaborative information on the user-item-feature graph, the users and their correspondingly interacted items seem more likely to form clusters (e.g., users 10,744, 18,171 and 3,364) in most cases.

8 CONCLUSION AND FUTURE WORK

Recommendation based on user review information has attracted increasing attention in the recent research communities. This article proposes to reformulate feature-aware recommendation with a signed hypergraph convolutional network. We develop a novel convolution operation for signed hyper-edges based on the basic characters of feature-aware recommendation and leverage neural models to capture the correlations among the users, items, and features. By the designed model, we can more sufficiently leverage the collaborative filtering information and discover more complex user-item-feature relations, which enable us to significantly improve the recommendation performance. We conduct extensive experiments based on real-world datasets to demonstrate the superiorities of our proposed model.

For the field of graph convolutional network, this article proposes the first solution to deploy convolutional operation on signed hypergraphs. In the future, we plan to extend this method with attention mechanism to discriminate different node importances and apply our model to other domains, such as social network and neural language processing.

From the perspective of personalized recommendation, our proposed method provides a promising avenue for formulating and modeling user feature-level sentiments via using the signed hypergraph. Actually, there is much room for improvement. For example, we can build personalized information aggregation paths to capture user diverse personalities. And also, we can design more advanced methods for corrupting hyper-edges, or even consider each hyper-edge as a whole for information propagation.

[9]SHCN-2 is selected because it can achieve the best performance according to Table 4.

ACKNOWLEDGMENTS

We gratefully appreciate the anonymous reviewers for their valuable and detailed comments that greatly helped to improve the quality of this article. Special thanks go to Associate Editor Hui Fang for her hard work in reviewing and providing important feedback.

REFERENCES

[1] Song Bai, Feihu Zhang, and Philip H. S. Torr. 2019. Hypergraph convolution and hypergraph attention. *arXiv preprint arXiv:1901.08150* (2019).

[2] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).

[3] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *J. Mach. Learn. Res.* 3, Jan. (2003), 993–1022.

[4] Yixin Cao, Zhiyuan Liu, Chengjiang Li, Juanzi Li, and Tat-Seng Chua. 2019. Multi-channel graph neural network for entity alignment. In *Proceedings of the Meeting of the Association for Computational Linguistics*. 1452–1461.

[5] Rose Catherine and William Cohen. 2017. Transnets: Learning to transform for recommendation. In *RecSys*. 288–296.

[6] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural attentional rating regression with review-level explanations. In *Proceedings of the World Wide Web Conference*. 1583–1592.

[7] Xu Chen, Hanxiong Chen, Hongteng Xu, Yongfeng Zhang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2019. Personalized fashion recommendation with visual explanations based on multimodal attention network: Towards visually explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. 765–774.

[8] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. 2016. Learning to rank features for recommendation over multiple categories. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 305–314.

[9] Xu Chen, Yongfeng Zhang, Qingyao Ai, Hongteng Xu, Junchi Yan, and Zheng Qin. 2017. Personalized key frame recommendation. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. 315–324.

[10] Xu Chen, Yongfeng Zhang, and Zheng Qin. 2019. Dynamic explainable recommendation based on neural attentive models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 53–60.

[11] Tyler Derr, Yao Ma, and Jiliang Tang. 2018. Signed graph convolutional networks. In *Proceedings of the IEEE International Conference on Data Mining*.

[12] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *Proceedings of the World Wide Web Conference*. ACM, 417–426.

[13] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3558–3565.

[14] Junyu Gao, Tianzhu Zhang, and Changsheng Xu. 2017. A unified personalized video recommendation via dynamic recurrent neural networks. In *Proceedings of the 25th ACM International Conference on Multimedia*. 127–135.

[15] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. TriRank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 1661–1670.

[16] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the World Wide Web Conference*. 173–182.

[17] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2020. A re-visit of the popularity baseline in recommender systems. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*.

[18] Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[19] 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 *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. 275–284.

[20] Lei Li, Li Chen, and Yongfeng Zhang. 2020. Towards controllable explanation generation for recommender systems via neural template. In *Companion Proceedings of the Web Conference 2020*. ACM/IW3C2, 198–202.

[21] Lei Li, Yongfeng Zhang, and Li Chen. 2020. Generate neural template explanations for recommendation. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*.

[22] Donghua Liu, Jing Li, Bo Du, Jun Chang, and Rong Gao. 2019. DAML: Dual attention mutual learning between ratings and reviews for item recommendation. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. 344–352.

[23] Yang Liu, Xiangji Huang, Aijun An, and Xiaohui Yu. 2008. Modeling and predicting the helpfulness of online reviews. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM'08)*. IEEE Computer Society, 443–452. DOI: https://doi.org/10.1109/ICDM.2008.94

[24] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *J. Mach. Learn. Res.* 9, Nov. (2008), 2579–2605.

[25] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*. 165–172.

[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. 452–461.

[27] 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 *Proceedings of the 11th ACM Conference on Recommender Systems*. 297–305.

[28] Omer Tal, Yang Liu, Jimmy Huang, Xiaohui Yu, and Bushra Aljbawi. 2019. Neural attention frameworks for explainable recommendation. *IEEE Trans. Knowl. Data Eng.* (2019).

[29] Yunzhi Tan, Min Zhang, Yiqun Liu, and Shaoping Ma. 2016. Rating-boosted latent topics: Understanding users and items with ratings and reviews. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Vol. 16. 2640–2646.

[30] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Multi-pointer co-attention networks for recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2309–2318.

[31] Ke Tu, Peng Cui, Xiao Wang, Fei Wang, and Wenwu Zhu. 2018. Structural deep embedding for hyper-networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[32] Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. 2018. Shine: Signed heterogeneous information network embedding for sentiment link prediction. In *Proceedings of the International Conference on Web Search and Data Mining*. 592–600.

[33] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. 2018. Explainable recommendation via multi-task learning in opinionated text data. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 165–174.

[34] Pengfei Wang, Yu Fan, Long Xia, Wayne Xin Zhao, Shaozhang Niu, and Jimmy Huang. 2020. KERL: A knowledge-guided reinforcement learning model for sequential recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 209–218.

[35] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 165–174.

[36] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5329–5336.

[37] Xiao Wang, Ruijia Wang, Chuan Shi, Guojie Song, and Qingyong Li. 2019. Multi-component graph convolutional collaborative filtering. *arXiv preprint arXiv:1911.10699* (2019).

[38] Xiaolong Wang, Yufei Ye, and Abhinav Gupta. 2018. Zero-shot recognition via semantic embeddings and knowledge graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6857–6866.

[39] Libing Wu, Cong Quan, Chenliang Li, Qian Wang, Bolong Zheng, and Xiangyang Luo. 2019. A context-aware user-item representation learning for item recommendation. *ACM Trans. Inf. Syst.* 37, 2 (2019), 1–29.

[40] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A neural influence diffusion model for social recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 235–244.

[41] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 346–353.

[42] Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 285–294.

[43] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Anand Louis, and Partha Talukdar. 2018. HyperGCN: Hypergraph convolutional networks for semi-supervised classification. *arXiv preprint arXiv:1809.02589* (2018).

[44] Handong Zhao, Yingqiang Ge, Xu Chen, Qiaoying Huang, Shijie Geng, Zhou Qin, Gerard de Melo, S. Muthukrishnan, Yongfeng Zhang, Yikun Xian, and Zuohui Fu. 2020. CAFE: Coarse-to-fine neural symbolic reasoning for explainable recommendation. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*.

[45] 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 *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 974–983.

[46] Yongfeng Zhang. 2015. Incorporating phrase-level sentiment analysis on textual reviews for personalized recommendation. In *Proceedings of the 8th ACM International Conference on Web Search and Data Mining*. 435–440.

[47] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W. Bruce Croft. 2017. Joint representation learning for Top-N recommendation with heterogeneous information sources. In *Proceedings of the ACM Conference on Information and Knowledge Management*. 1449–1458.

[48] 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 *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*. 83–92.

[49] Yongfeng Zhang, Haochen Zhang, Min Zhang, Yiqun Liu, and Shaoping Ma. 2014. Do users rate or review? Boost phrase-level sentiment labeling with review-level sentiment classification. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1027–1030.

[50] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A deep reinforcement learning framework for news recommendation. In *Proceedings of the World Wide Web Conference*. 167–176.

[51] Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*. 425–434.

[52] Yu Zhu, Ziyu Guan, Shulong Tan, Haifeng Liu, Deng Cai, and Xiaofei He. 2016. Heterogeneous hypergraph embedding for document recommendation. *Neurocomputing* 216 (2016), 150–162.

Received January 2020; revised August 2020; accepted September 2020