BasConv: Aggregating Heterogeneous Interactions for Basket Recommendation with Graph Convolutional Neural Network

Zhiwei Liu*[†] Mengting Wan[‡] Stephen Guo[§] Kannan Achan[§] Philip S. Yu*

Abstract

Within-basket recommendation reduces the exploration time of users, where the user's intention of the basket matters. The intent of a shopping basket can be retrieved from both user-item collaborative filtering signals and multi-item correlations. By defining a basket entity to represent the basket intent, we can model this problem as a basket-item link prediction task in the User-Basket-Item (UBI) graph. Previous work solves the problem by leveraging user-item interactions and item-item interactions simultaneously. However, collectivity and heterogeneity characteristics are hardly investigated before. Collectivity defines the semantics of each node which should be aggregated from both directly and indirectly connected neighbors. Heterogeneity comes from multi-type interactions as well as multi-type nodes in the UBI graph. To this end, we propose a new framework named **BasConv**, which is based on the graph convolutional neural network. Our BasConv model has three types of aggregators specifically designed for three types of nodes. They collectively learn node embeddings from both neighborhood and high-order context. Additionally, the interactive layers in the aggregators can distinguish different types of interactions. Extensive experiments on two real-world datasets prove the effectiveness of BasConv.

1 Introduction

Shopping for a group of items during a session is a common behavior of users when shopping online [1, 2, 3]. We define a *basket* to contain a set of items which are bought at the same time by the same user [1, 4, 5]. *Within-basket recommendation* [1, 2] is to recommend items for a shopping basket, which can reduce the exploration time of users. In order to make a recommendation, we need to understand the *intent* of the basket. For example, *milk* can be either more related to *bread*

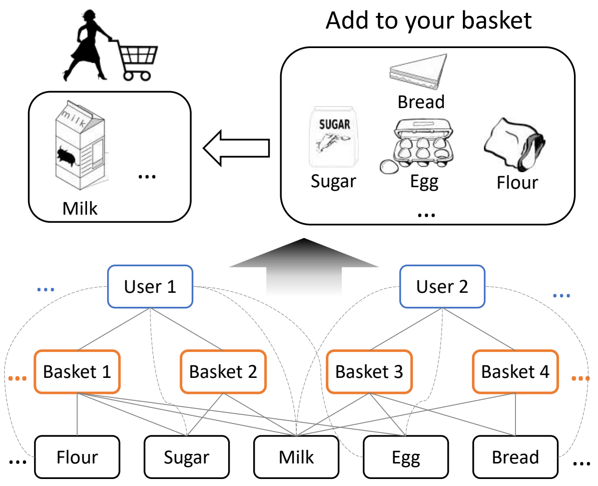


Figure 1: The top figure shows an example basket recommendation for when milk is in the basket, we recommend items to the current basket. The bottom figure is an example of a UBI graph where user-basket, basket-item, and user-item interactions exist.

or to *flour* when the purpose of the corresponding basket is for *breakfast* or for *making a cake*, respectively. Ignoring the intent of a basket compromises the ability of a model to distinguish the different item relationships within the same basket. The intent of the basket should be retrieved from two perspectives: (1) *User-item collaborative filtering* (CF) signals, which model item semantic information as well as the user's personal tastes [6, 7, 8] and (2) *multi-item correlations*, which reveal users' intents for a shopping basket [5, 4, 2, 9], e.g. {*milk*, *flour*, *sugar*, *egg*} contributes to the purpose of making a cake.

As illustrated in Fig. 2a, we introduce the **user-basket-item (UBI) graph** to characterize these two types of structures simultaneously. Unlike the traditional user-item bipartite graph [10, 11, 12], *basket* nodes are incorporated to represent the semantics of users' shopping baskets. The basket recommendation problem can thus be defined as predicting the links between basket nodes and item nodes. On top of this graph, we propose a new framework named **BasConv**

*work is done during internship at Walmart Labs

[†]University of Illinois at Chicago;{zliu213, psyu}@uic.edu

[‡]Airbnb, Inc; mengting.wan@airbnb.com; Work was done while the author was at UC San Diego.

[§]Walmart Labs; {sguo,kachan}@walmartlabs.com to tackle this problem with a graph convolutional neural network (GCNN) [13, 11, 14]. Different from prior work, we are able to address the following two aspects:

- **High-Order Collectivity.** The semantic information of each node can be collected from its neighbors and other relevant nodes through high-order paths. For example, to represent a basket node, information can be aggregated from not only the associated user and the items included in this basket, but also other shopping baskets owned by the same user. In this way, interactions among users, baskets, and items can be holistically modeled.

- **Heterogeneity.** As shown in Fig. 2, by differentiating the types of nodes (i.e., user, basket or item), we build different aggregators to propagate information on the UBI graph. By doing so, heterogeneous relationships (e.g. user-basket and item-basket interactions) can be distinguished.

Specifically, in our proposed framework BasConv, the user aggregator (Fig. 2b) generates the personalized user embeddings by aggregating the information of all connected baskets and items. The basket aggregator (Fig. 2c) summarizes the multi-item correlations inside the corresponding baskets and combines it with the associated user. The item aggregator (Fig. 2d) yields item embeddings by collecting the intents of their corresponding baskets. Moreover, the recursive learning procedure and the multi-layer structure of BasConv capture the high-order information over the UBI graph. The contributions of this paper are as follows:

- **New Framework:** We propose a new framework, BasConv, to tackle the basket recommendation problem. We model the recommendation task as predicting the interactions over the UBI graph. Then, we design heterogeneous aggregators to learn the embedding of each node. Finally, the predictive layer outputs the ranking results from the learned embeddings.

- **Heterogeneous Interactions:** Various interactive layers in BasConv retrieve the heterogeneous interactions on the UBI graph. The interactive layers explicitly model the user-item, basket-item, and user-basket interactive signals.

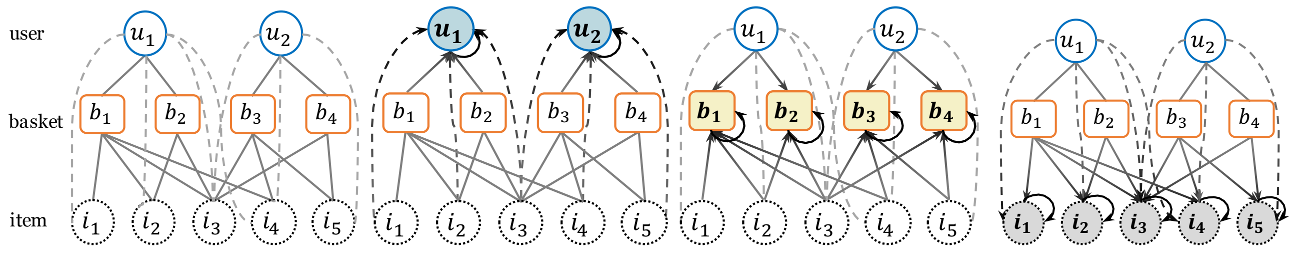
- **Heterogeneous Aggregators:** We design three different aggregators for user, basket and item entities, which are built upon heterogeneous interactive layers to retrieve both heterogeneous nodes and heterogeneous linkage signals in the UBI graph. The same type of aggregators in the same layer share common training parameters.

2 Related Work

In this section, we review two research areas: basket recommendation and graph convolutional neural network (GCNN)-based recommender systems.

2.1 Basket Recommendation. Basket recommendation requires not only the user-item CF signals [7, 10, 11], but also the item-item relationships [9, 2], e.g., the complementary and substitution relationships. Item A is a substitute for item B if A can be purchased instead of B, while item A is complementary to item B if it can be purchased in addition to B [9]. Both of these concepts are extensively investigated in the previous work [9, 2, 5]. Sceptre [9] is proposed to model and predict relationships between items from the text of their reviews and the corresponding descriptions. Item2vec [15] learns the item embedding from the user-generated item sets, i.e. the baskets, based on the word2vec model. BFM [1] learns one more basket-sensitive embedding for each item rather than only one embedding, which can help to find item relation w.r.t. the current shopping basket. Prod2vec [16] applies the same idea to learn the distributed representation of items and support the recommendation of the ad in *Yahoo! Mail*. Triple2vec [2] improves the within-basket recommendation via (user, item A, item B) triplets sampled from baskets, where item A and item B have a complementary relationship. Later, [5] proposed a Bayesian network to unify the context information and high-order relationships of items, learning context-aware dual embeddings of items. We argue that these works have no discrimination towards heterogeneous types of interactions and explore little on the collectivity pattern in the basket recommendation problem.

2.2 GCNN-based Recommender System. By defining the user-item interaction as a bipartite-graph [10, 11], we can apply recently developed graph convolutional [13, 12] models to design recommender systems [17, 12, 10]. GCN [13] is proposed to learn the graph embeddings from spectral graph convolutions. Based on this idea, GC-MC [17] predicts the links between users and items by applying the graph convolutional network as the graph encoder. Graphsage [12] learns the graph embedding by aggregating the information of neighbors, which is extended as a large-scale recommender system, namely PinSage [18]. SpectralCF [10] designs a spectral convolutional filter to model the CF signals in user-item bipartite graphs. NGCF [11] explicitly models the high-order CF signal in the user-item bipartite graph. It designs a multi-layer graph convolutional network by constructing, then aggregating the messages over the graph. However, none



(a) UBI Example (b) User Aggregator (c) Basket Aggregator (d) Item Aggregator

Figure 2: We present a UBI graph example in (a). The user, basket, and item aggregator examples are illustrated in (b), (c) and (d), respectively. The aggregated nodes are shadowed with associated colors.

of the previous methods study the heterogeneity in the graph. Our proposed BasConv model can capture heterogeneous interaction signals. To our best knowledge, BasConv is the first GCNN model to solve the basket recommendation task.

3 Preliminary and Definition

We have a set of items $I = \{i_1, i_2, \ldots, i_{|I|}\}$ and a set of users $U = \{u_1, u_2, \ldots, u_{|U|}\}$. Given a partial shopping basket b which contains a set of items $I_b \subset I$ and is associated with a user $u \in U$, we recommend items ($i^* \in I \setminus I_b$) to complete the current basket b. The recommendations are based on the intent of the basket, which can be inferred from the semantics of items within b as well as user u's preferences. In particular, we define a user-basket-item (UBI) graph to represent interactions among users, baskets and items.

Definition 1. ***(UBI Graph).*** *A UBI graph is defined as $\mathcal{G} = (\mathcal{V}_u, \mathcal{V}_b, \mathcal{V}_i, \mathcal{E}_{ub}, \mathcal{E}_{bi}, \mathcal{E}_{ui})$. $\mathcal{V}_u$, $\mathcal{V}_b$ and $\mathcal{V}_i$ represent the vertices of user, basket and item, respectively. $\mathcal{E}_{ub}$, $\mathcal{E}_{bi}$ and $\mathcal{E}_{ui}$ denote the edges between users and baskets, between baskets and items and between users and items, respectively. Each basket is connected to one user exclusively.*

The UBI graph is an extension of the user-item bipartite graph [10] with one more basket entity. The user-basket part, as illustrated in Fig. 2a, is of a tree structure, while the user-item and basket-item part are both of bipartite graph structure. We define three different types of interaction matrices from the UBI graph, i.e., $\mathbf{R}_{ub}$, $\mathbf{R}_{bi}$ and $\mathbf{R}_{ui}$ for the user-basket interaction matrix, basket-item interaction matrix, and user-item interaction matrix, respectively. We show the user-basket interaction matrix as

$$\mathbf{R}_{ub}(r, j) = \begin{cases} 1 & \text{if } (u_r, b_j) \in \mathcal{E}_{ub} \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

We define the other two interaction matrices $\mathbf{R}_{bi}$ and $\mathbf{R}_{ui}$ in the same way, as in Eq. (3.1) from the UBI graph with entity substitution, i.e. $\mathbf{R}_{bi}(j, k) = 1$ when $(b_j, i_k) \in \mathcal{E}_{bi}$ and $\mathbf{R}_{ui}(r, k) = 1$ when $(u_r, i_k) \in \mathcal{E}_{ui}$.

4 BasConv Model

In this section, we present the structure of our proposed BasConv model. BasConv has two major parts, the embedding layer and the heterogeneous aggregators. BasConv has three different types of aggregators, i.e., the basket, user, and item aggregator. The stucture of BasConv is presented in Fig. 3.

4.1 Embedding. We use d dimensional embedding vectors e_u, e_i, $e_b \in \mathbb{R}^d$ to describe the user, item, and basket entities in the UBI graph, respectively. We can define three embedding matrices to form a look-up table for the embeddings of each type of entity, e.g., the user embedding matrix, $\mathbf{E}_u = \left[\mathbf{e}_{u_1}, \mathbf{e}_{u_2}, \ldots, \mathbf{e}_{u_{|U|}}\right]$ where $|U|$ represents the total number of users in the graph. The embedding matrices represent the information of the entities in the UBI graph. They are propagated along with the structural information of the UBI graph into the next layer of GCNN. At each layer, we refine the embeddings of all nodes by leveraging both heterogeneous and high-order interactions of the UBI graph. Hence, we use superscript to denote the layer number, e.g., $\mathbf{E}_u^{(0)}$ for the initial embedding of users and $\mathbf{E}_u^{(l)}$ for the embedding of users at the l-th layer. The number of parameters in BasConv at each layer is independent of the number of nodes in the graph and linear to the dimension of embeddings, which will be analyzed in detail later.

4.2 Heterogeneous Aggregator. In this section, we present the three different aggregators of the BasConv, i.e., basket aggregator, user aggregator, and item aggregator. As showed in Fig. 2, these aggregators are built upon the following propagation layers.

- **Self-Propagation Layers.** Each type of node has a self-propagation layer within its corresponding aggregator, i.e., basket-self-propagation layer, user-self-propagation layer, and item-self-propagation layer. In order to reduce the complexity of BasConv, we share the self-propagation layer for all types of nodes, denoted as δ. In each aggregator, the l-th self-propagation layer $\delta^{(l)}$ retrieves the self information from l layer ($l \geq 0$). For example, the basket-self-propagation can be calculated as

$$\delta^{(l)}(\mathbf{e}_u^{(l)}) = \mathbf{e}_u^{(l)}\mathbf{W}_{sp}^{(l)}. \tag{4.2}$$

The user and item self-propagation layers can be defined similarly.

- **Interactive Layers.** We further introduce three interactive layers in these aggregators for the user-basket (ψ), user-item (γ), and item-basket (η) interactions respectively. For example, we have the user-basket interactive layer

$$\psi^{(l)}\left(\mathbf{e}_u^{(l)}, \mathbf{e}_b^{(l)}\right) = \frac{1}{p_u}\mathbf{e}_u^{(l)} \odot \mathbf{e}_b^{(l)}\mathbf{W}_{ub}^{(l)}, \tag{4.3}$$

where $\mathbf{e}_u^{(l)}$ and $\mathbf{e}_b^{(l)}$ are interchangeable, p_u is the normalized factor w.r.t. the degree of the corresponding node, and $\mathbf{W}_{ub}^{(l)}$ is a shared trainable $d \times d$ matrix. Similarly, we can define the user-item interactive layer $\gamma^{(l)}$, the item-basket interactive layer $\eta^{(l)}$, and have the corresponding parameter matrix $\mathbf{W}_{ui}^{(l)}$, $\mathbf{W}_{ib}^{(l)}$.

On top of these propagation layers, we are able to formally define the three aggregators as follows.

4.2.1 Basket Aggregator. As showed in Fig. 2c, the aggregated information of basket node b at l-th layer $\mathbf{h}_b^{(l)}$ should be aggregated from both the embeddings of all the items connected with the basket and the corresponding user embedding:

$$\begin{aligned}\mathbf{h}_b^{(l)} = \; & \overbrace{\delta^{(l)}\left(\mathbf{e}_b^{(l)}\right)}^{\text{basket-self-propagation}} + \overbrace{\psi^{(l)}\left(\mathbf{e}_{u_b}^{(l)}, \mathbf{e}_b^{(l)}\right)}^{\text{user-basket interaction}} \\ & + \underbrace{\sum_{i \in \mathcal{N}_i(b)} \eta^{(l)}\left(\mathbf{e}_b^{(l)}, \mathbf{e}_i^{(l)}\right)}_{\text{item-basket interaction}}.\end{aligned} \tag{4.4}$$

In Eq. (4.4), the aggregated information at the l-th layer for basket b is denoted as $\mathbf{h}_b^{(l)}$, which is aggregated from the basket-self-propagation layer δ, the user-basket interactive layer ψ and the basket-item interactive layer η. Each basket connects with only one corresponding user, thus only one basket-associated user u_b's embedding $\mathbf{e}_{u_b}^{(l)}$ passes into the user-basket interactive layer ψ along with basket embedding $\mathbf{e}_b^{(l)}$. The basket aggregator collects all the items connected with the basket b, which is denoted as the item neighbor function $\mathcal{N}_i(b)$. The basket-item interactive layer computes the interactive information of each item with the basket. Summing up the interaction for all the items with the current basket aggregates the item semantics for the basket. The aggregated information is passed to an activation function to output the embedding of basket b at $(l+1)$-th layer as

$$\mathbf{e}_b^{(l+1)} = \sigma\left(\mathbf{h}_b^{(l)}\right). \tag{4.5}$$

The computational flow and structural details for the basket aggregator are presented in Fig. 3. We show how to compute the basket embedding $\mathbf{b}_2^{(l)}$ by aggregating the information from the previous two layers. First, the aggregators in $(l-2)$-th layer generate the $(l-1)$-th embedding of b_2 and its neighbors by aggregating the embeddings of all corresponding neighbor entities. Then the basket aggregator $\mathbf{Agg}_b^{(l-1)}$ aggregates the embeddings of b_2's neighbors and b_2 from previous layer. Finally, $\mathbf{Agg}_b^{(l-1)}$ outputs the l-th embedding of b_2. We present the structure of the basket aggregator on the right-hand side in Fig. 3. User embedding $\mathbf{u}_1^{(l-1)}$ and basket embedding $\mathbf{b}_2^{(l-1)}$ are input together into the user-basket interactive layer. Meanwhile, all the connected items are input along with a basket to the basket-item interactive layer. Then they have summed up with the output from the self-propagation layer and input to activation function σ to output the embedding.

4.2.2 User Aggregator. A user u connects with several baskets, thus the aggregated information of user u at the l-th layer, denoted as $\mathbf{h}_u^{(l)}$, should be learned by aggregating all the basket's information. Moreover, the user-item interaction is important to capture the user's personalized and item semantic information. Hence, we define user aggregator as:

$$\begin{aligned}\mathbf{h}_u^{(l)} = \; & \overbrace{\delta^{(l)}\left(\mathbf{e}_u^{(l)}\right)}^{\text{user-self-propagation}} + \overbrace{\sum_{b \in \mathcal{N}_b(u)} \psi^{(l)}\left(\mathbf{e}_u^{(l)}, \mathbf{e}_b^{(l)}\right)}^{\text{user-basket interaction}} \\ & + \underbrace{\sum_{i \in \mathcal{N}_i(u)} \gamma^{(l)}\left(\mathbf{e}_u^{(l)}, \mathbf{e}_i^{(l)}\right)}_{\text{user-item interaction}}.\end{aligned} \tag{4.6}$$

The user-self-propagation layer δ propagates the self information from the previous layer to the next layer. Each user u connects with a set of baskets, which is denoted as $\mathcal{N}_b(u)$. We sum up all the user-basket interactions from user-basket interactive layer ψ to aggregate the information from baskets to the user. In

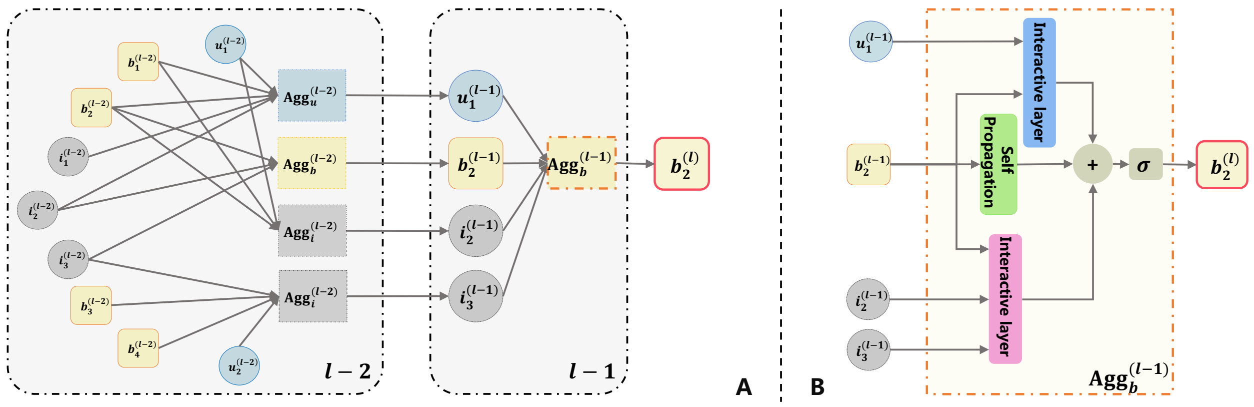


Figure 3: On the left part **A**, we present the structure of **BasConv** with an example of aggregating the l-th embedding of basket b_2, i.e., $\mathbf{b}_2^{(l)}$, based on the UBI example in Fig. 2. In part A, three types of aggregators in the $(l-2)$-th layer aggregate the corresponding embeddings in the $(l-2)$ layer and output the embedding for the $(l-1)$-th layer. Then, the basket aggregator $\mathbf{Agg}_b^{(l-1)}$ produces the embedding $\mathbf{b}_2^{(l)}$ by aggregating the embeddings of b_2 and its neighbor nodes in the $(l-1)$-th layer. On the right part **B**, we show the inner structure of basket aggregator $\mathbf{Agg}_b^{(l-1)}$. The blue block and pink block represent the user-basket and item-basket interactive layers, respectively.

addition to baskets, user u is also connected with a set of items, denoted as $\mathcal{N}_i(u)$. The user-item interactive layer γ learns the user-item interactive information, which is aggregated for all the corresponding items to user u. Then, the aggregated user information is input into an activation function to generate the user embedding as $\mathbf{e}_u^{(l+1)} = \sigma\left(\mathbf{h}_u^{(l)}\right)$.

4.2.3 Item Aggregator.

We follow the same rule as in the basket aggregator and user aggregator in the previous section to learn the item aggregated information:

$$\begin{aligned}\mathbf{h}_i^{(l)} = \; & \overbrace{\delta^{(l)}\left(\mathbf{e}_i^{(l)}\right)}^{\text{item-self-propagation}} + \overbrace{\sum_{u\in\mathcal{N}_i(u)} \gamma^{(l)}\left(\mathbf{e}_u^{(l)}, \mathbf{e}_i^{(l)}\right)}^{\text{user-item interaction}} \\ & + \underbrace{\sum_{b\in\mathcal{N}_b(i)} \eta^{(l)}\left(\mathbf{e}_b^{(l)}, \mathbf{e}_i^{(l)}\right)}_{\text{item-basket interaction}}. \end{aligned} \tag{4.7}$$

Item-self-propagation layer retrieves the information from the embedding of item i in the previous layer. We aggregate all the information from the user-item interactive layer and basket-item interactive layer by incorporating the interaction among the item and the corresponding neighboring users, as well as neighboring baskets, respectively. Embedding is generated by passing the aggregated information to an activation function as $\mathbf{e}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\right)$.

4.2.4 Matrix Form.

Given parameter matrices $\mathbf{W}_{sp}^{(l)}$, $\mathbf{W}_{ub}^{(l)}$, $\mathbf{W}_{ui}^{(l)}$, $\mathbf{W}_{ib}^{(l)}$, the above three aggregators can be represented in explicit matrix forms. Specifically, we have the following updating rules for the user embedding matrices

$$\begin{aligned}\mathbf{E}_u^{(l+1)} = & \sigma\Big(\mathbf{E}_u^{(l)}\mathbf{W}_{sp}^{(l)} + \tilde{\mathbf{R}}_{ub}\mathbf{E}_b^{(l)} \odot \mathbf{E}_u^{(l)}\mathbf{W}_{ub}^{(l)} \\ & + \tilde{\mathbf{R}}_{ui}\mathbf{E}_i^{(l)} \odot \mathbf{E}_u^{(l)}\mathbf{W}_{ui}^{(l)}\Big),\end{aligned} \tag{4.8}$$

the item embedding matrices

$$\begin{aligned}\mathbf{E}_i^{(l+1)} = & \sigma(\mathbf{E}_i^{(l)}\mathbf{W}_{sp}^{(l)} + \tilde{\mathbf{R}}_{bi}^\top\mathbf{E}_b^{(l)} \odot \mathbf{E}_i^{(l)}\mathbf{W}_{bi}^{(l)} \\ & + \tilde{\mathbf{R}}_{ui}^\top\mathbf{E}_u^{(l)} \odot \mathbf{E}_i^{(l)}\mathbf{W}_{ui}^{(l)}),\end{aligned} \tag{4.9}$$

and the basket embedding matrices

$$\begin{aligned}\mathbf{E}_b^{(l+1)} = & \sigma(\mathbf{E}_b^{(l)}\mathbf{W}_{sp}^{(l)} + \tilde{\mathbf{R}}_{ub}^\top\mathbf{E}_u^{(l)} \odot \mathbf{E}_b^{(l)}\mathbf{W}_{ub}^{(l)} \\ & + \tilde{\mathbf{R}}_{bi}\mathbf{E}_i^{(l)} \odot \mathbf{E}_b^{(l)}\mathbf{W}_{bi}^{(l)}).\end{aligned} \tag{4.10}$$

$\tilde{\mathbf{R}}$ is the normalized matrix of previously defined interaction matrix $\mathbf{R}$ in Sec. 3, i.e., $\tilde{\mathbf{R}} = \mathbf{D}^{-1}\mathbf{R}$ and $\mathbf{D}$ is the corresponding diagonal degree matrix. For example, $\tilde{\mathbf{R}}_{ub}$ is a user-basket interaction matrix, and $\tilde{\mathbf{R}}_{ub} = \mathbf{D}_{ub}^{-1}\mathbf{R}_{ub}$ where $D_{ii} = \sum_{j=1}^{|B|}\tilde{\mathbf{R}}_{ub}(i,j)$.

4.3 Model Prediction.

BasConv outputs the embeddings of users, baskets, and items after L-layer graph convolutions. We concatenate embeddings from all layers to incorporate the information from neighborhoods as well as high-order interactions. For example, the output embedding of basket b is $\mathbf{e}_b^* = \mathbf{e}_b^{(0)}\|\mathbf{e}_b^{(1)}\|\cdots\|\mathbf{e}_b^{(L)}$,

Table 1: Dataset Statistics

Dataset	*#User*	*#Item*	*#Basket*	*Avg. Basket*	*Avg. Size*	*#Interactions*
Instacart	$22,168$	$40,044$	$65,672$	2.96	37.0	$2,495,695$
Walmart	$44,218$	$77,599$	$130,707$	2.96	52.5	$6,997,572$

where $\|$ stands for the concatenation operation. The same concatenation is applied for both user embeddings $\mathbf{e}_u^*$ and item embeddings $\mathbf{e}_i^*$. With the embedding, we can provide recommendations for the basket b with the candidate items based on their predicted preference scores. For an item i, this prediction is defined as

$$\hat{y}(b,i) = \mathbf{e}_{u_b}^{*\top}\mathbf{e}_i^* + \mathbf{e}_b^{*\top}\mathbf{e}_i^*. \tag{4.11}$$

The first term captures the user-item signals, recommending the items of the corresponding user's interests. The second term models the relationship between the intent of the basket and the embedding of the item.

4.4 Optimization. We optimize the model based on BPR loss [6]. We sample a positive item i and a negative item j for a partially given basket b, where the positive item is sampled within the basket and the negative item is sampled from items outside the basket. The final BPR loss is:

$$\mathcal{L} = -\sum_{(b,i,j)\in\mathcal{S}} \log\sigma\big(\hat{y}(b,i) - \hat{y}(b,j)\big) + \lambda\|\Theta\|_2^2, \tag{4.12}$$

where the first term denotes the BPR interaction loss, and the second term denotes the regularization to the trainable parameters (λ is the regularizing factor). The trainable parameters consist of the embedding parameters and the aggregator parameters. We use Xavier [19] initialization. Note for the embedding parameters, we only train the initial user embedding and the item embedding, i.e., only $\{\mathbf{E}_u^{(0)}, \mathbf{E}_i^{(0)}\}$ are trainable. The initial basket embedding matrix is fixed with all zeros (i.e., $\mathbf{E}_b^{(0)} = \mathbf{0}$) because of its extremely large number. Our models are trained in Tensorflow with the batch-wise Adam [20] optimizer.

5 Experiment

5.1 Datasets. We conduct experiments on two real-world datasets, the *Instacart* dataset[1] and the dataset collected from the *Walmart* online grocery shopping website[2].

- **Instacart** is an online grocery shopping dataset, which is published by *instacart.com* [21]. It contains over 3 million grocery transaction records from over 200 thousand users on around 50 thousand items.
- **Walmart Grocery** is an online service provided by *walmart.com* for shopping groceries. We sampled 100 thousand users, whose transaction data are retrieved to conduct the experiment.

We filter transactions based on their basket sizes to fulfill the requirement of adequate basket signals. Baskets with less than 30 items and 40 items for *Instacart* dataset and *Walmart* dataset (respectively) are removed. The statistics of the preprocessed datasets are summarized in Table 1.

5.2 Experimental Settings

5.2.1 Evaluation Metrics. We recommend items to complete the partially given baskets. We evaluate the performance of models by the Top-K recommendation metrics, i.e., the *Recall@K*, *HR@K*, and *NDCG@K* [7, 11]. The default K is 100.

5.2.2 Baselines. We compare our model with the following methods in recommender systems:

- **ItemPop**: From some previous work [2, 4], in real-world, users always buy some popular items. Thus, we design a baseline *itemPop* that considers the user-wise *frequency* of items in the training data as the ranking criteria for recommendation.
- **BPR-MF** [6]: This is a standard method of modeling user-item interactions with bayesian personalized ranking (BPR) loss. We merge the baskets w.r.t. the same user and sample positive and negative items. We recommend items within baskets based on the corresponding user embeddings and item embeddings.
- **GC-MC** [17]: It is the recent GCNN model to complete the rating matrix. We adopt the idea in [17] using the GCN model [13] to complete the UBI graph. We recommend baskets with items based on the corresponding user embedding. We use one GCN layer connected by an MLP layer structure for predicting the links.
- **NGCF** [11]: This is a state-of-the-art GCNN method that explicitly models the high-order inter-

[1] `https://www.instacart.com/datasets/grocery-shopping-2017`

[2] `https://grocery.walmart.com/`

Table 2: Within-Basket Recommendation Comparison

Data	Method	Recall	NDCG	HR
Insta.	ItemPop	0.1490	0.1604	0.5704
	BPR-MF	0.1687	0.1800	0.6380
	triple2vec	0.1711	0.1855	0.6543
	GC-MC	0.1758	0.1871	0.6529
	NGCF	0.1887$*$	0.2013$*$	0.71729$*$
	BasConv	**0.2092**	**0.2281**	**0.7712**
	Improv.%	**7.34%**	**1.74%**	**9.89%**
Wal.	ItemPop	0.0490	0.0586	0.2732
	BPR-MF	0.0430	0.0620	0.2619
	triple2vec	0.0462	0.0663	0.2713
	GC-MC	0.0392	0.0706	0.2874
	NGCF	0.0492$*$	0.0722$*$	0.2903$*$
	BasConv	**0.0530**	**0.0841**	**0.3394**
	Improv.%	**7.72%**	**16.48%**	**16.91%**

action of users and items with a multi-layer neural network. However, it has no distinction for the heterogeneous interactions.

- **Triple2vec** [2]: It is one of the most recent works to address the within-basket recommendation task. We sample the triplets from baskets and train the user and item embeddings.

5.2.3 Parameter Settings. For a fair comparison, the embedding size is set to 64 across different methods. The hyper-parameters of BasConv are selected based on the recommendation performance (*Recall*) on the validation set. The learning rate is selected from $\{10^{-5}, 5\times10^{-5}, 10^{-4}, 5\times10^{-4}, 10^{-3}, 5\times10^{-3}\}$. Our model converges best when learning rate is 5×10^{-4} and 10^{-3} for the Instacart and Walmart dataset, respectively. The layer size is selected from {1,2,3,4} for NGCF and BasConv.

5.3 Within-Basket Recommendation

5.3.1 Overall Comparison. In this section, we compare the performance of different models on the within-basket recommendation task. We split 80% items of each basket as training data and the remaining 20% as test data for both of the datasets. All the models are trained on the training data and the hyperparameters are tuned based on the validation data, which is 20% randomly masked data from the training data. We report the within-basket recommendation results on the test data in Table 2. The highest value is in bold, and the second-highest value is with a star($*$).

BasConv improves the Recall, NDCG, and Hit Ratio 7.72% (7.34%), 15.9% (1.74%) and 18.60% (9.89%) respectively on the Walmart (Instacart) dataset. The improvement comes from two aspects: (1) BasConv aggregates the information from high-order structure via heterogeneous aggregators, which capture multi-type node information. (2) The heterogeneous interactive layers explicitly model the multi-type interactions in the UBI graph. The results prove that our proposed BasConv model can retrieve the high-order collectivity and the heterogeneity pattern from the UBI graph, and hence improve the performance of recommendation.

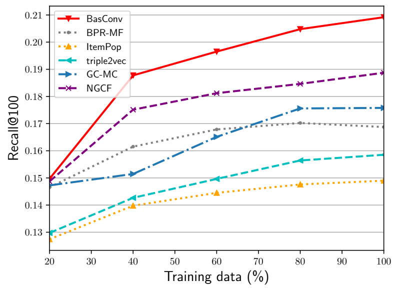
Table 3: Effects of Layer Number

Data	Method	Recall	NDCG	HR
Insta.	BasConv-1	0.1799	0.2221	0.7666
	BasConv-2	0.1818	0.2235	0.7699
	BasConv-3	**0.2092**	**0.2281**	**0.7712**
	BasConv-4	0.1868	0.2238	0.7605
Wal.	BasConv-1	0.0499	0.0728	0.2892
	BasConv-2	**0.0530**	**0.0841**	**0.3394**
	BasConv-3	0.0500	0.0837	0.3305
	BasConv-4	0.0496	0.0830	0.3272

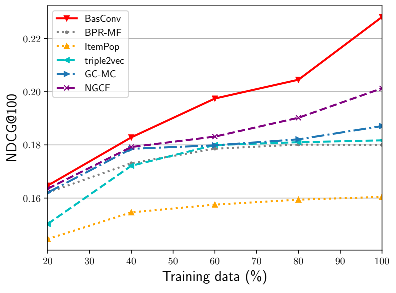
Although user-wise item popularity (ItemPop) lacks generalization power, as it is able to memorize users' simple shopping patterns, it still on both datasets as it explicitly models the interactions of user-items. However, since all existing methods ignore the collectivity and heterogeneity of the basket recommendation problem, they can be outperformed by BasConv in terms of all three evaluation metrics.

5.3.2 Sensitivity Analysis. Data sparsity can spoil the performance of recommender systems since using limited interactions might be insufficient to comprehensively retrieve entity semantics. We thus investigate the sensitivity of models w.r.t. the number of interactions of basket-items. We compare BasConv with other methods when training data is sampled $\{20\%, 40\%, 60\%, 80\%, 100\%\}$ from the training data used in the previous section to study the sensitivity of different methods. The results are presented in Fig. 4. We observe that BasConv consistently outperforms other methods as the data volume increases, which proves that high-order collectivity and heterogeneity is important to retrieve the semantics of the UBI graph. Also, we find that all methods have a tendency to perform better when data size increases, but BasConv improves faster than other models.

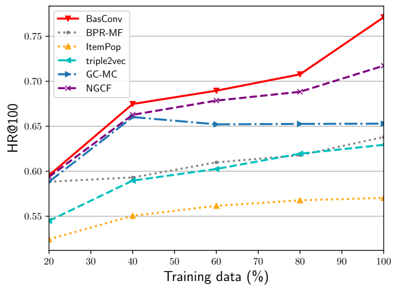
5.4 Study of BasConv. BasConv has a multi-layer structure which collects the high-order interactions in the UBI graph. The layer number is an important hyper-parameter for BasConv. We conduct experiments



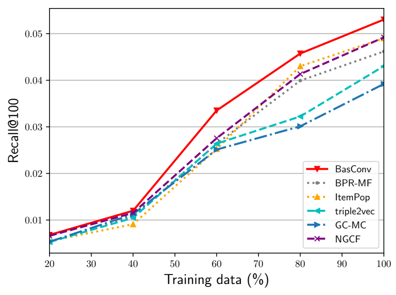
(a) Recall on Instacart



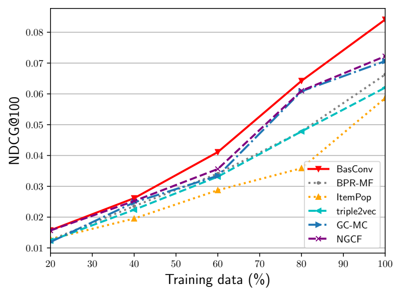
(b) NDCG on Instacart



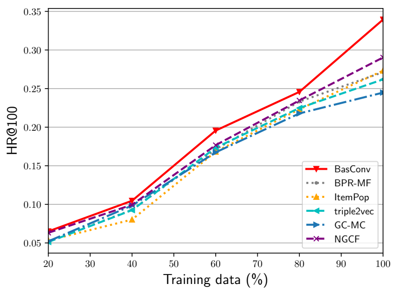
(c) Hit Ratio on Instacart



(d) Recall on Walmart



(e) NDCG on Walmart



(f) Hit Ratio on Walmart

Figure 4: Model Performance w.r.t. different percentage of training data on two datasets.

on a different number of layers of BasConv to investigate the model performance change. The layer number is chosen from {1,2,3,4}, and the results are displayed in Table 3. We find that on the Instacart dataset, BasConv performs best when layer number is 3, while on Walmart dataset, BasConv performs best when the layer size is 2. A possible reason for this performance drop could be higher-order interactions may contain less relevant information thus introducing noises in the model.

5.5 Case Study. In this section, we show the ranking results of BasConv in Table 4 for a real-world case study[3]. In this table, we show a portion of the training items in the given basket at the top, the test items (i.e., the ground truth) in the same basket in the middle, and the recommended items at the bottom. We observe the following two potential intents in the given basket: (1) the 'Lemon Verbena Dish Soap' and the 'Sweeper Dry Sweeping Cloth Refills' may indicate the user's intention of *cleaning* and (2) the 'Homestyle Belgian Waffles' may indicate the user's intention of *breakfast*. BasConv can identify both of them and recommends *milk* and *eggs* for the breakfast intention and *dishwasher detergent* for cleaning, which are all verified by the ground truth. We have successfully predicted 3 products in the top 7 recommendations while others preserve reasonable explanations. For instance, the 'Brioche Slider Buns' is also related to the breakfast purpose and 'Pasta' is recommended as 'Cheddar' is in the given basket. This real-world use case justifies that BasConv is able to identify the intent of the basket, model the semantics of items, and thus provide a satisfying recommendation.

6 Conclusion

In this paper, we introduced the user-basket-item (UBI) graph where the basket entity represents the intent of the shopping basket. Upon this, we formulated the within-basket recommendation problem as a link prediction problem. In order to solve the high-order collectivity and heterogeneity challenges in identifying basket intent, we introduced three types of aggregators to incorporate heterogeneous interactive signals and collectivity semantics. Extensive experiments on Instacart and Walmart datasets demonstrated the effectiveness of BasConv in modeling high-order collectivity and heterogeneity. The real-world case study validated that our proposed model could identify the intents of the current shopping basket.

[3]We delete some irrelevant descriptive words to save space.

Table 4: A case study on instacart dataset. We find two different intents in the given basket—*cleaning* (in red) and *breakfast* (in blue). The correctly predicted items are underlined.

Items within the partially given basket (Condition)
Lemon Verbena Dish Soap
Sweeper Dry Sweeping Cloth Refills
Homestyle Belgian Waffles
Lightly Salted Kettle Potato Chips - Sea Salt
Baked Rice and Corn Puffs, Aged White Cheddar
Chocolate Brownie Kid Z Bar
...
Test items purchased in the same basket (Ground Truth)
Grade A Large Brown Eggs
Unsweetened Original Milk
Gala Apples
Scent Dishwasher Detergent
Free & Gentle Fabric Softener Dryer Sheets
Unscented Liquid Laundry Detergent
Recommended items to the partically given basket (Prediction)
Organic Chocolate Chip ZBar Kids Energy Snack
Scent Dishwasher Detergent
Brioche Slider Buns
Penne Rigate 41 Pasta
Unsweetened Original Milk
Cage Free 100% Liquid Egg Whites
Grade A Large Brown Eggs

7 Acknowledgements

This work is supported in part by NSF under grants III-1526499, III-1763325, III-1909323, and CNS-1930941.

References

[1] D. Le, H. W. Lauw, and Y. Fang, "Basket-sensitive personalized item recommendation," in *IJCAI*, 2017, pp. 2060–2066.

[2] M. Wan, D. Wang, J. Liu, P. Bennett, and J. McAuley, "Representing and recommending shopping baskets with complementarity, compatibility and loyalty," in *CIKM*, 2018, pp. 1133–1142.

[3] W.-C. Kang, E. Kim, J. Leskovec, C. Rosenberg, and J. McAuley, "Complete the look: Scene-based complementary product recommendation," in *CVPR*, 2019, pp. 10 532–10 541.

[4] J. Bai, C. Zhou, J. Song, X. Qu, W. An, Z. Li, and J. Gao, "Personalized bundle list recommendation," in *The World Wide Web Conference*, 2019, pp. 60–71.

[5] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Modeling complementary products and customer preferences with context knowledge for online recommendation," *arXiv preprint arXiv:1904.12574*, 2019.

[6] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *UAI*, 2009, pp. 452–461.

[7] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *WWW*, 2017, pp. 173–182.

[8] S. Rendle, "Factorization machines," in *ICDM*, 2010, pp. 995–1000.

[9] J. McAuley, R. Pandey, and J. Leskovec, "Inferring networks of substitutable and complementary products," in *SIGKDD*, 2015, pp. 785–794.

[10] L. Zheng, C. Lu, F. Jiang, J. Zhang, and P. S. Yu, "Spectral collaborative filtering," in *RecSys*, 2018, pp. 311–319.

[11] X. Wang, X. He, M. Wang, F. Feng, and T. Chua, "Neural graph collaborative filtering," in *SIGIR*, 2019, pp. 165–174.

[12] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017, pp. 1024–1034.

[13] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[14] Z. Liu, L. Zheng, J. Zhang, J. Han, and P. S. Yu, "Jscn: Joint spectral convolutional network for cross domain recommendation," *arXiv preprint arXiv:1910.08219*, 2019.

[15] O. Barkan and N. Koenigstein, "Item2vec: Neural item embedding for collaborative filtering," in *RecSys*, 2016.

[16] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp, "E-commerce in your inbox: Product recommendations at scale," in *SIGKDD*, 2015, pp. 1809–1818.

[17] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.

[18] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *SIGKDD*, Y. Guo and F. Farooq, Eds., 2018, pp. 974–983.

[19] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *IJCAI*, 2010, pp. 249–256.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[21] "The instacart online grocery shopping dataset 2017," in *https://www.instacart.com/datasets/grocery-shopping-2017*, Accessed on Sep. 2019.