Basket Recommendation with Multi-Intent Translation Graph Neural Network

Zhiwei Liu[1], Xiaohan Li[1], Ziwei Fan[1], Stephen Guo[2], Kannan Achan[2], and Philip S. Yu[1]
[1]Department of Computer Science, University of Illinois at Chicago, IL, USA
{zliu213, xli241, zfan20, psyu}@uic.edu
[2]Walmart Labs, CA, USA; {SGuo, KAchan}@walmartlabs.com

***Abstract*—The problem of basket recommendation (BR) is to recommend a ranking list of items to the current basket. Existing methods solve this problem by assuming the items within the same basket are correlated by one semantic relation, thus optimizing the item embeddings. However, this assumption breaks when there exist multiple intents within a basket. For example, assuming a basket contains {*bread, cereal, yogurt, soap, detergent*} where {*bread, cereal, yogurt*} are correlated through the "breakfast" intent, while {*soap, detergent*} are of "cleaning" intent, ignoring multiple relations among the items spoils the ability of the model to learn the embeddings. To resolve this issue, it is required to discover the intents within the basket. However, retrieving a multi-intent pattern is rather challenging, as intents are latent within the basket. Additionally, intents within the basket may also be correlated. Moreover, discovering a multi-intent pattern requires modeling high-order interactions, as the intents across different baskets are also correlated. To this end, we propose a new framework named as Multi-Intent Translation Graph Neural Network (MITGNN). MITGNN models T intents as tail entities translated from one corresponding basket embedding via T relation vectors. The relation vectors are learned through multi-head aggregators to handle user and item information. Additionally, MITGNN propagates multiple intents across our defined basket graph to learn the embeddings of users and items by aggregating neighbors. Extensive experiments on two real-world datasets prove the effectiveness of our proposed model on both transductive and inductive BR. The code[1] is available online.**

***Index Terms*—Recommender System, Basket Recommendation, Graph Neural Network, Multi-intent Pattern**

I. Introduction

The problem of Basket Recommendation (BR) [1]–[4] is to recommend a ranking list of items that users are likely to buy together. It is necessary to model the user-item interactive signal and item-item correlation signal simultaneously [3]. The user-item collaborative filtering (CF) signal assumes that users with similar interaction patterns share semantically similar items, which is important in representing the user and item semantics [5], [6]. The item-item correlation signal is extracted from items within the same basket [1], [3], which is crucial for capturing the basket level item relations. For example, *cereal* and *yogurt* are usually in the same basket as illustrated in Figure 1, as they are complementary with each other [3]. We

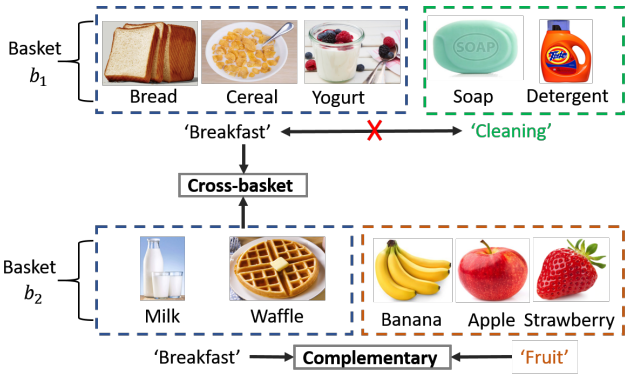


Fig. 1: An example of the multi-intent pattern in baskets. "Breakfast", "Cleaning", and "Fruit" are intents in the baskets. Items within the same intents are correlated. Intents may exhibit "complementary" relations. The "Breakfast" intent relates items across baskets.

can model such item relations to recommend a set of items within the same basket.

Existing methods model item relations in BR by assuming that the items within the same basket are semantically related with each other [1]–[4], [7]. Specifically, Triple2Vec [3] minimizes the distance of the embeddings of items within the same basket. DREAM [1] averages the embedding of items within the same basket so as to model the correlations. Xu, et. al. [7] propose to model the items within the same basket as contexts of each other. BasConv [4] and BGCN [8] learn basket embeddings by aggregating both user embeddings and item embeddings. These works improve the performance of BR as they can model the basket-level semantics. However, in the real-world, the item relations within baskets are rather complex. According to previous works [9]–[12], users' shopping behavior is usually of multiple interests. Regarding the BR problem, we observe that there are often multiple intents, as illustrated in Figure 1. Hence, we should design a new model to handle the multi-intent pattern in the BR problem.

The advantages of modeling a multi-intent pattern are in three folds. Firstly, the multi-intent pattern helps to find **complex semantic relations of items**. For example, in Figure 1, one basket b_1 contains items {*bread, cereal, yogurt, soap,*

*Work is done during the internship at Walmart Labs.
[1]https://github.com/JimLiu96/MITGNN

978-1-7281-6251-5/20/$31.00 ©2020 IEEE

detergent}, where the {*bread, cereal, yogurt*} are correlated through the "breakfast" intent, while {*soap, detergent*} are of "cleaning" intent. Simply assuming items within the same basket are of one semantic relation to each other [1], [3], [4], [13] spoils the ability of the model to comprehend the basket-level semantics. Besides, the **relation between intents** can also be discovered to benefit BR. For example, in Figure 1, basket b_2 has intents "breakfast" and "fruit", which are complementary intents [3], indicating items in one intent should be complements to items in the other. Whereas in b_1, the items in "breakfast" have no relation to the items in "cleaning", though they are in the same basket. As a result, multi-intent relations comprehensively express semantics in baskets. Additionally, the item correlations can be captured more precisely **across different baskets** through multiple intents. For example, in Figure 1, a basket b_2 consists of {*milk, waffle, banana, apple, strawberry*}, where {*milk, waffle*} is of "breakfast" intent, while {*banana, apple, strawberry*} is of "fruit" intent. Though different from b_1 in both item-level and basket-level semantics, it still shares a common intent "breakfast" with basket b_1, and thus we can find the correlation of those items.

In order to discover the multi-intent pattern from a basket, in this paper, we propose a new framework, **M**ulti-**I**ntent **T**ranslation **G**raph **N**eural **N**etwork (**MITGNN**). Inspired by translation-based models [14]–[16] in knowledge graph embedding that translate head entity embeddings into tail entity embeddings to model entity relations, MITGNN learns T different translations [14] from a basket (i.e., head entity) to construct those intents (i.e. tail entities), as illustrated in Figure 3. The translation-based model [14]–[16] is well-known as it can model the relations between entities. The Translation-based model is built upon triples (h, r, q), where r is the relation between head entity h and tail entity q. The embedding of q, viewed as the translation of h, is learned by adding the embedding of h with the embedding of r, i.e., $\mathbf{q} = \mathbf{h} + \mathbf{r}$. Since we model those intents as T tail entities from the same basket entity, we therefore have $\|\mathbf{q}_i - \mathbf{q}_j\| = \|\mathbf{r}_i - \mathbf{r}_j\|$. Hence, the usage of translations in MITGNN can also model the relations among multiple intents. For example, in Figure 3, the distance between different intents indicates the correlation of those intents.

Therefore, MITGNN combines the translation-based model with the GNN model. We construct MITGNN upon our defined basket graph, which consists of user, item, and basket entities. Different from existing methods that learn T relation vectors as free parameters [14]–[17], We integrate item and user information into the intent learning process. Inspired by the design intuition of GNN [18], [19], the relation vector between basket and intents is learned by T-head aggregators. Each head is associated with an intent, aggregating the user and item information of a basket to learn the relation vector. In addition to those intent embeddings, MITGNN learns user embeddings and item embeddings by aggregating from their neighbor embeddings. Since different baskets are connected through user entities and item entities, stacking multiple layers helps to propagate information across different baskets.

The contributions of this paper are:

- **New framework:** We propose a new framework combining the translation-based model with GNN. It can not only propagate information across our defined basket graph, but also model the relations of the learned intents.
- **Multi-intent pattern:** To the best of our knowledge, this is the first work to address the multi-intent pattern in the BR problem. The intents are tail entities translated from basket entities. We use multi-head aggregators to generate the translation vectors.
- **Extensive Experiments:** For the evaluation of our model, we conduct BR experiments on two large-scale real-world datasets. We evaluate the effectiveness of our proposed model under both transductive and inductive BR settings.

II. Related Work

A. Basket Recommendation (BR)

Basket Recommendation (BR) requires not only user-item CF signals [5], [6], [20], but also item-item relationships [3], [21], 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 [21]. Both of these concepts are extensively investigated in previous work [3], [7], [21]. Sceptre [21] is proposed to model and predict relationships between items from the text of their reviews and the corresponding descriptions. Item2vec [13] learns item embeddings from user-generated item sets, i.e., the baskets, based on the word2vec model. BFM [2] and Prod2vec [22] learn one more basket-sensitive embedding for each item, rather than only one embedding, which can help find item relations within the basket. DREAM [1] proposes to use RNN structure to exploit the dynamics of user embeddings so as to complete BR. Triple2vec [3] improves within-basket recommendation via (user, item A, item B) triplets sampled from baskets. Later, Xu, et. al. [7] propose a Bayesian network to unify the context information and high-order relationships of items. BasConv [4] is the first work using heterogeneous graph embeddings to complete BR. All these works simply aggregate the information within the basket, while having no discussion towards basket-level intent mining and exploring little regarding the correlations of those intents.

B. GNNs in recommender systems

Currently, Graph Neural Networks (GNNs) have been widely explored to process graph-structure data [23]–[25]. Motivated by convolutional neural networks, Bruna et al. [26] propose graph convolutions in the spectral domain. Then, GCN [19] simplified the previous graph spectral convolution operations. GraphSAGE [18] inductively generates node embeddings via sampling and aggregating neighbors. GNNs also have proven their effectiveness in recommender systems. GCMC [27] completes the user-item matrix based on spectral GCN [19]. PinSage [28] introduces GraphSAGE [18] into a recommender system on an item-item graph. Spectral CF [20] leverages spectral convolutions over the user-item bipartite

graph. NGCF [6] hierarchically propagates user and item embeddings to model high-order connectivity. DGCF [29] applies dynamic graph to recommender systems. BasConv [4] is the first work that uses a GNN to solve the BR problem. Although these methods achieve significant improvements on some types of task, all ignore the multi-intent pattern in the BR problem. Our proposed MITGNN combines a translation-based model with a GNN model to handle the multi-intent pattern.

C. Intent Mining

Recently, a variety of intent mining methods have been proposed. Temporal Deep Structured Semantic Model (TDSSM) [30] combines users' long-term and short-term preferences to generate their real-time intention. Deep Interest Network (DIN) [9] considers the diversity of user interests, and utilizes the attention mechanism to calculate the relevance between the current advertising commodity and historical commodities clicked by the user. Graph Intention Network (GIN) [10] applies a graph structure to deal with the behavior sparsity problem. IntentGC [11] learns explicit preferences and heterogeneous relationships from user behaviors and item information via a GCN. ASLI [12] models users' intents by leveraging their historical interactions, using a self-attention layer and a temporal convolutional network layer to learn the intent embedding. However, regarding the intent mining in BR problems, all existing methods require improvements. As illustrated in Figure 1, there are multiple intents in a basket, these intents are composed of several items, and they are correlated with each other. Therefore, we should not only use the combination of items to generate multiple intents, but also model the correlations between intents.

III. Preliminary

In this section, we present the preliminaries and definitions related to our Basket Recommendation (BR) problems.

A. Problem Definition

We have a set of items $\mathcal{I} = \{i_1, i_2, \ldots, i_{|\mathcal{I}|}\}$ and a set of users $\mathcal{U} = \{u_1, u_2, \ldots, u_{|\mathcal{U}|}\}$. The BR problem is defined under two different settings based on whether the basket to predict is given during training, i.e. the transductive and inductive BR problems:

Definition 1: **(Transductive and Inductive BR):** Assume a set of shopping baskets $\mathcal{B} = \{b_1, b_2, \ldots, b_{|\mathcal{B}|}\}$, each of which contains a set of items $\mathcal{I}_b \subset \mathcal{I}$ and is associated with a user $u \in \mathcal{U}$. We recommend a ranking list of items ($i^* \in \mathcal{I} \setminus \mathcal{I}_{b^*}$) to complete the current basket b^*. If $b^* \in \mathcal{B}$, it is a transductive BR problem. Otherwise, if $b^* \notin \mathcal{B}$, it is an inductive BR problem.

The goals of the transductive and inductive BR are both to fill the current basket b^* with items not contained in it. However, transductive BR differs from inductive BR in whether the current basket is in the given basket set, or in other words, whether it is in the training data. We present the prediction process of transductive BR and inductive BR on the right-hand side of Figure 2. For transductive BR, $b^* = b$ and is already in the training data, while for inductive BR, we have a new basket b^*. The transductive BR is the same as *within-basket* recommendation [3], [4], and the inductive one is similar to *next-basket* recommendation [1], [3]. The reason why we use 'transductive'('inductive') rather than 'with-in'('next') will be introduced later.

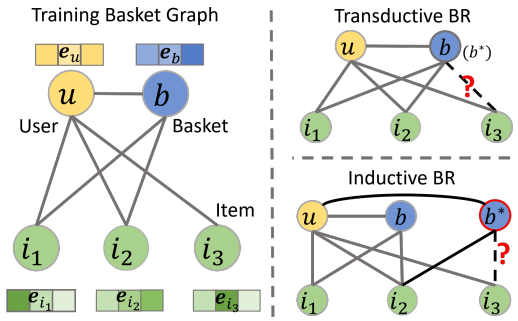


Fig. 2: An example of the basket graph for training (left). The prediction process of transductive BR and inductive BR (right).

During an online shopping session, we find that 'add-to-basket' are *asynchronous* behavior of users. Users may add items to the current basket at anytime and save them until checkout. During this period, we can recommend items to the current basket before it is finalized. It suggests that we can use the current basket as the training data, i.e., to make a transductive BR. Additionally, as a result of the latency and infrastructure constraints, it should be better to only consider finalized baskets as training data, and test the BR performance on current baskets, which is inductive BR. Evaluation on both settings contributes to comprehensive analysis of the BR model.

B. Problem Formulation

We solve the BR problem based on a novel graph structure, i.e. the basket graph, which is as follows:

Definition 2: **(Basket Graph):** A basket graph $\mathcal{G}$ is defined as $\mathcal{G} = (\mathcal{V}_u, \mathcal{V}_b, \mathcal{V}_i, \mathcal{E}_{ub}, \mathcal{E}_{bi}, \mathcal{E}_{ui})$, which contains N users, S baskets, and M items. $\mathcal{V}_u$, $\mathcal{V}_b$ and $\mathcal{V}_i$ represent the nodes 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.

Note that each basket may be connected with more than one item, but only with one user. An example of a basket graph is illustrated on the left-hand side in Figure 2. We can learn the graph embedding based on the defined basket graph as shown on the left-hand side of Figure 2. Then, the basket recommendation problem shifts to predicting the edges between the basket nodes and item nodes, which is presented on the right-hand side of Figure 2. For the current basket b^*, we rank the scores of predicting the edges $\{f(v_{b^*}, v_i) | i \in \mathcal{I} \setminus \mathcal{I}_{b^*}\}$, where $f(\cdot, \cdot)$ is the learned predictive function of edges. The ranked scores are the returned results of

our basket recommender system. We explain the reason why we use 'transductive'('inductive') for the BR problem, rather than 'with-in'('next').

In this paper, we need to train the basket node embedding to make a prediction as shown on the right-hand side of Figure 2. Transductive BR is the same as the *within-basket recommendation* [3], [4] setting. However, here we use transductive BR to make it comparable with inductive BR. Inductive BR is similar to *next-basket* recommendation [1], [3]. Under these circumstances, the model has no information for the predictive basket during the training procedure. Hence, previous methods use the associated user embedding [1], [3] to complete BR for the next basket. However, we argue that the next basket is **not empty** during prediction. A user opens a new basket by adding an item to the cart. Then, more items are selected into the current basket. For instance, in the Inductive BR of Figure 2, the current basket b^* is already connected with item i_2. We should use these connected items to make an inference of the embedding of the current basket before prediction. This inference-before-prediction procedure [18] is an inductive procedure. In this paper, we adopt a similar idea as in [18], [31] to train aggregators for the inference of basket embeddings.

IV. Proposed Model

In this section, we present the structure of our proposed MITGNN model, as illustrated in Figure 4. Our model is based on two designing intuitions: 1) the translation-based model [14]–[16] that generates multiple intents as tail entities and models correlations of those intents; 2) the GNN framework [6], [18], [19] that learns node embeddings by aggregating the information from neighbors.

A. Embedding

There exist three different types of nodes, i.e., user, item, and basket, which can be represented as d dimensional vector embeddings, denoted as $\mathbf{e}_u$, $\mathbf{e}_i$, and $\mathbf{e}_b$, respectively. Thus, we can define three embedding matrices. For instance, we define the item embedding layer as $\mathbf{E}_i = [\mathbf{e}_{i_1}, \mathbf{e}_{i_2}, \ldots, \mathbf{e}_{i_{|\mathcal{I}|}}]$, where $|\mathcal{I}|$ denotes the total number of items. We retrieve the embedding by the index of the corresponding node. Since the proposed GNN model is an L-layer structure, we aggregate the embeddings from one layer to the next layer, and use superscripts to denote the layer number, e.g., $\mathbf{e}_i^{(l)}$ represents the embedding of item i at the l-th layer. The embeddings of the entities at the 0-th layer are randomly initialized and trainable.

B. Multi-Intent Module

This is one of the main components of MITGNN. The Multi-intent module discovers the multi-intent pattern for baskets. There are two major parts: one is the **multi-intent generator** that learns multiple intents by modeling them as tail entities translated from basket entities; the other is the **multi-intent aggregator** that propagates those intents for the usage of next layer.

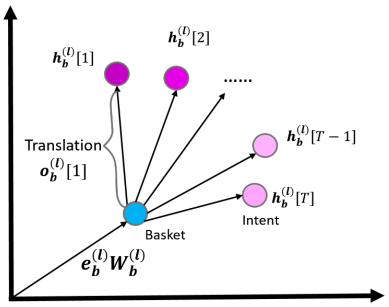


Fig. 3: The multiple intents translated from the basket.

1) Multi-intent generator: Assuming that there are T different intents for the basket b, we define a T-head intent generator to retrieve T hidden intents from basket b. Intuitively, we view the intents as different translations [14] from the same basket, which is illustrated in Figure 3. The translation-based module is inspired by existing translation-based models [14]–[16] to learn the embeddings of entities and relations in knowledge graphs. The translation-based module in MITGNN treats T hidden intents as tail entities translated from the head basket entity. Thus, we model the embedding of the t-th hidden intent of basket b at $l-$th layer $\mathbf{h}_b^{(l)}[t]$, as:

$$\mathbf{h}_b^{(l)}[t] = \mathbf{e}_b^{(l)}\mathbf{W}_b^{(l)} + \mathbf{o}_b^{(l)}[t], \quad t = 1, 2, \ldots, T \quad l = 0, 1, ..., L \tag{1}$$

where $\mathbf{e}_b^{(l)}\mathbf{W}_b^{(l)}$ is mapping the basket information to intent space and $\mathbf{o}_b^{(l)}[t]$ denotes the relation vector between the head basket entity and the $t-$th tail intent entity. $\mathbf{W}_b^{(l)} \in \mathbb{R}^{d\times d}$ is the weight matrix that propagates the basket embedding from $l-$th layer. The translation $\mathbf{o}_b^{(l)}[t]$ models the relation between the basket and $t-$th intent. By using this translation-based model, it is easy to show that the distance (i.e. correlations) of different intents is:

$$\mathbf{h}_b^{(l)}[m] - \mathbf{h}_b^{(l)}[n] = \mathbf{o}_b^{(l)}[m] - \mathbf{o}_b^{(l)}[n], \tag{2}$$

where $1 \le m, n \le T$.

Previous methods [14]–[16] set the relation vectors $\mathbf{o}_b^{(l)}$ as free parameters. However, in BR problem, relation vectors between intents and baskets depend on the information carried by users and items. Thus, we use T-head aggregators at different layers to learn it by aggregating neighboring nodes' information. Each t-th head is corresponding to t-th intent as:

$$\mathbf{o}_b^{(l)}[t] = \mathbf{e}_u^{(l)}\mathbf{W}_1^{(l)}[t] + \sum_{i\in\mathcal{N}_i(b)} \mathbf{e}_i^{(l)}\mathbf{W}_2^{(l)}[t], \tag{3}$$

where $\mathbf{e}_u$ and $\mathbf{e}_i$ denotes the embedding of user and item, respectively. $\mathcal{N}_i(b)$ denotes the neighboring items of basket b, i.e., the items within the basket b. As denoted in Eq. (3), each head is associated with a pair of weight matrix $(\mathbf{W}_1[t], \mathbf{W}_2[t])$, where $\mathbf{W}_1[t], \mathbf{W}_2[t] \in \mathbb{R}^{d\times d}$. Thus, we only introduce T pairs of weight matrices for generating T intents. $\mathbf{W}_1$ is to aggregate the user information, while $\mathbf{W}_2$ is to aggregate the item information.

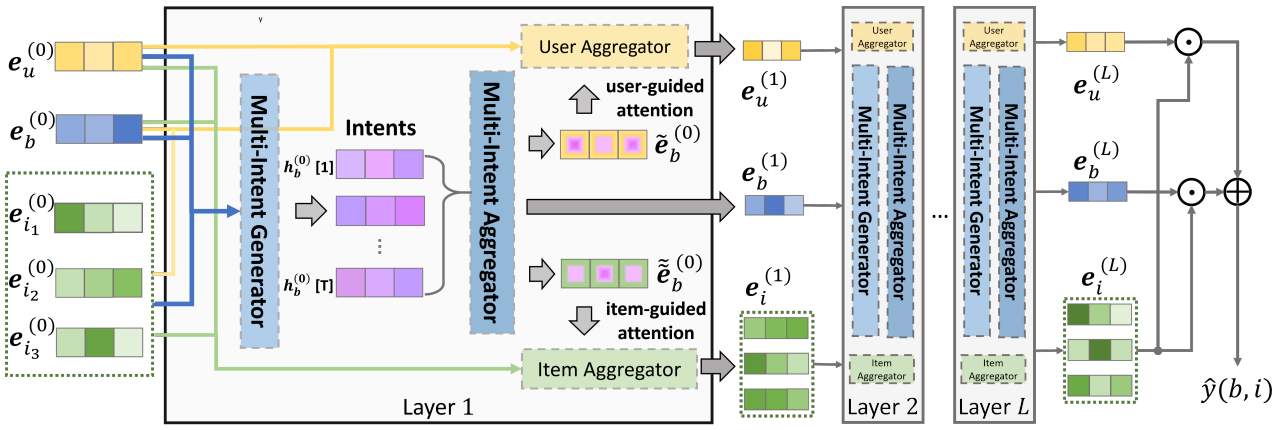


Fig. 4: The MITGNN framework. MITGNN is of L-layer structure. In the first layer, the multi-intent signals can be captured by the proposed Multi-Intent Generator, which generates T hidden intent $\mathbf{h}_b[t]$ from the input embeddings. Then, the Multi-Intent Aggregator digests intents and outputs three embeddings. One is the basket embedding $\mathbf{e}_b$ for the usage of the next layer. The other two embeddings are the user-guided basket embedding $\tilde{\mathbf{e}}_b$ and item-guided basket embedding $\tilde{\tilde{\mathbf{e}}}_b$ for the usage of the user aggregator and item aggregator, respectively. User and item aggregators output the user and item embeddings, respectively.

2) Multi-intent Aggregator: Since the generated multiple intents are hidden embeddings, those intents should be further aggregated to the next layer to learn node embeddings. Firstly, we should aggregate the multiple intents into the basket embedding for the usage of next layer. As different intents contribute unequally to the basket information, we adopt the idea in attention mechanism [32], [33] and propose an self-attention aggregator for aggregating the embeddings of intents as one basket embedding $\mathbf{e}_b^{(l+1)}$:

$$\mathbf{e}_b^{(l+1)} = \sigma\left(\sum_{t=1}^{T} \gamma_{bt}^{(l)} \mathbf{h}_b^{(l)}[t]\right), \tag{4}$$

where $\gamma_{bt}^{(l)}$ is the attention weight indicating the importance of $t-$th intent to the current basket b at $l-$th layer, which is:

$$\gamma_{bt}^{(l)} = \frac{\exp\left(\sigma\left(\{\mathbf{h}_b^{(l)}[t] \oplus \mathbf{e}_b^{(l)}\}\mathbf{a}_b^\top\right)\right)}{\sum_{t=1}^{T}\exp\left(\sigma\left(\{\mathbf{h}_b^{(l)}[t] \oplus \mathbf{e}_b^{(l)}\}\mathbf{a}_b^\top\right)\right)}, \tag{5}$$

where $\mathbf{a}_b \in \mathbb{R}^{2d}$ is the weights for the basket-guide attention layer. The $\sigma(\cdot)$ in Eq. (4) and Eq. (5) denotes the activation function, where we use LeakyReLU[2] [34]. $\oplus$ denotes the concatenation of two embeddings. Eq. (5) is intuitively the softmax of the information between each intent and the basket.

Secondly, since GNN framework requires the basket embeddings to learn both user embeddings and item embeddings, the embeddings of those hidden intents of the basket b should be aggregated into one user-guided basket embedding and one item-guided basket embedding. Therefore, we propose to use type-guided attention mechanism to aggregate the intents into two **type-guided basket embeddings**. The designing intuition is that the correlations between users and intents are distinct from the correlations between items and intents. Because the former is to discover the preference of users to the intents [35], [36], while the later is to model the relation of items to the intents [4]. Hence, we need to apply different weights for each intent vector during intent aggregation. We also use the the self-attention mechanism as type-guided attention like in Eq. (4):

$$\tilde{\mathbf{e}}_b^{(l)} = \sigma\left(\sum_{t=1}^{T} \alpha_{bt}^{(l)} \mathbf{h}_b^{(l)}[t]\right), \quad \tilde{\tilde{\mathbf{e}}}_b^{(l)} = \sigma\left(\sum_{t=1}^{T} \beta_{bt}^{(l)} \mathbf{h}_b^{(l)}[t]\right), \tag{6}$$

where $\tilde{\mathbf{e}}_b^{(l)}$ is the user-guided basket embedding generated by aggregating the hidden intent embedding with user-guided attention weights $\alpha_{bt}^{(l)}$, and $\tilde{\tilde{\mathbf{e}}}_b^{(l)}$ is the item-guided basket embedding generated by aggregating the hidden intent embedding with item-guided attention weights $\beta_{bt}^{(l)}$. The weights $\alpha_{bt}^{(l)}$ and $\beta_{bt}^{(l)}$ are calculated from the user-guided and item-guided attention, respectively:

$$\alpha_{bt}^{(l)} = \frac{\exp\left(\sigma\left(\{\mathbf{h}_b^{(l)}[t] \oplus \mathbf{e}_u^{(l)}\}\mathbf{a}_u^\top\right)\right)}{\sum_{t=1}^{T}\exp\left(\sigma\left(\{\mathbf{h}_b^{(l)}[t] \oplus \mathbf{e}_u^{(l)}\}\mathbf{a}_u^\top\right)\right)}, \tag{7}$$

$$\beta_{bt}^{(l)} = \frac{\exp\left(\sigma\left(\{\mathbf{h}_b^{(l)}[t] \oplus \bar{\mathbf{e}}_i^{(l)}\}\mathbf{a}_i^\top\right)\right)}{\sum_{t=1}^{T}\exp\left(\sigma\left(\{\mathbf{h}_b^{(l)}[t] \oplus \bar{\mathbf{e}}_i^{(l)}\}\mathbf{a}_i^\top\right)\right)}, \tag{8}$$

where $\mathbf{a}_u \in \mathbb{R}^{2d}$ and $\mathbf{a}_i \in \mathbb{R}^{2d}$ are respectively the weights for user-guided and item-guided attention layer, and $\bar{\mathbf{e}}_i^{(l)}$ is the average of all the item embeddings. Note that the user-guided basket embedding $\tilde{\mathbf{e}}_b^{(l)}$ and item-guided basket embedding $\tilde{\tilde{\mathbf{e}}}_b^{(l)}$ are all intermediate embeddings, which means they are not required to store by the model. By leveraging the user-guided and item-guided basket embeddings, we can further design the user and item aggregators.

[2]Other activation function can be used, e.g. Sigmoid.

C. User and Item Aggregators

By stacking multiple layers, we propagate the node information to its surrounding neighbors from one layer to the next layer. In other words, the node embedding at $(l+1)-$th layer is aggregated from the $l-$th layer embedding of its neighbors. Previously, we introduce how to aggregate the basket embeddings. In this section, we present user aggregator and item aggregator for learning user and item embeddings.

User Aggregator: Each user is both interacted with several baskets and the corresponding items. Hence, to learn the $(l+1)$-th layer's user embedding $\mathbf{e}_u^{(l+1)}$, the user aggregator should sum both the self-information of u and Neighboring Information (NI_u) of u:

$$\mathbf{e}_u^{(l+1)} = \sigma\left(\mathbf{e}_u^{(l)} + NI_u^{(l)}\right). \tag{9}$$

Inside the activation function $\sigma(\cdot)$ are two terms. The first term $\mathbf{e}_u^{(l)}$ denotes the user information propagated from $l-$th layer. The second term $NI_u^{(l)}$ denotes the Neighbor Information of user u at $l-$th layer, which is:

$$NI_u^{(l)} = \frac{1}{|\mathcal{N}_b(u)|}\sum_{b\in\mathcal{N}_b(u)}\tilde{\mathbf{e}}_b^{(l)} + \frac{1}{|\mathcal{N}_i(u)|}\sum_{i\in\mathcal{N}_i(u)}\mathbf{e}_i^{(l)}, \tag{10}$$

where $\mathcal{N}_b(u)$ and $\mathcal{N}_i(u)$ denote the neighboring baskets and neighboring items of user u, respectively. There are two terms on the right hand side in Eq. (10). The first term is the neighbor information from baskets. The information of basket b is retrieved as multiple intents, which are already aggregated to user-guided basket embedding $\tilde{\mathbf{e}}_b^{(l)}$ as in Eq. (6). Since each user has more than one basket, we average the user-guided basket embeddings for all neighboring baskets. The second term is the neighboring information from items. We average all the embeddings of the items connected to user u.

Item Aggregator: Each item is connected by a set of users and also has edges with a set of baskets. Therefore, an item aggregator should sum both the self-information of an item i and the neighboring information NI_i of it to learn the item embedding $\mathbf{e}_i^{(l+1)}$:

$$\mathbf{e}_i^{(l+1)} = \sigma\left(\mathbf{e}_i^{(l)} + NI_i^{(l)}\right). \tag{11}$$

Inside the activation function $\sigma(\cdot)$ are two terms. The first term $\mathbf{e}_i^{(l)}$ denotes the item information propagated from $l-$th layer. The second term $NI_i^{(l)}$ denotes the Neighbor Information of item i at $l-$th layer, which is:

$$NI_i^{(l)} = \frac{1}{|\mathcal{N}_b(i)|}\sum_{b\in\mathcal{N}_b(i)}\tilde{\tilde{\mathbf{e}}}_b^{(l)} + \frac{1}{|\mathcal{N}_u(i)|}\sum_{i\in\mathcal{N}_u(i)}\mathbf{e}_u^{(l)}, \tag{12}$$

where $\mathcal{N}_b(i)$ and $\mathcal{N}_u(i)$ denote the neighboring baskets and neighboring users of item i, respectively. There are two terms on the right hand side in Eq. (12). The first term denotes neighboring information from baskets. The information of basket b is retrieved as multiple intents, which are already aggregated to item-guided basket embedding $\tilde{\tilde{\mathbf{e}}}_b^{(l)}$ as in Eq. (6). Since each item exists in more than one basket, we average the item-guided basket embeddings for all neighboring baskets. The second term is the neighboring information from users. We average all the embeddings of users connected to item i.

By leveraging the user and item aggregators, we can update the embeddings from layer 0 to layer L and output the final prediction by using the predictive layer.

D. Predictive Layer

To rank a list of items for recommending a basket, we should output the score between basket b and the items i. We obtain the representation of the associated user u, the basket b, and the item i by inferring and concatenating the embedding at each layer. Thus, the final representation of u, b, and i respectively are $\mathbf{e}_u^* = \mathbf{e}_u^{(0)}\|\mathbf{e}_u^{(1)}\|\ldots\|\mathbf{e}_u^{(L)}$, $\mathbf{e}_b^* = \mathbf{e}_b^{(0)}\|\mathbf{e}_b^{(1)}\|\ldots\|\mathbf{e}_b^{(L)}$, and $\mathbf{e}_i^* = \mathbf{e}_i^{(0)}\|\mathbf{e}_i^{(1)}\|\ldots\|\mathbf{e}_i^{(L)}$. Concatenation of embeddings from all layers stores the information from the initial embeddings to high-order aggregated embeddings. They are passed into a predictive function. In this paper, the predictive function is defined as the sum of two dot products:

$$\hat{y}(b,i;u) = \mathbf{e}_u^*\mathbf{e}_i^{*\top} + \mathbf{e}_b^*\mathbf{e}_i^{*\top}, \tag{13}$$

where the first term $\mathbf{e}_u^*\mathbf{e}_i^{*\top}$ predict the score between user and item, and the second term $\mathbf{e}_b^*\mathbf{e}_i^{*\top}$ predict the score between basket and item. The combination of those two scores measures how likely we should recommend the item i to basket b. The user u is the associated user of basket b.

E. Optimization

We use BPR loss [37] to optimize the parameters of our MITGNN model. Given a basket b, we sample the positive sample i as the items within the current basket and negative sample j from the other items. Additionally, we add a regularization term for the parameters. Hence, the final optimization loss function is:

$$\mathcal{L} = -\sum_{(b,i,j)\in\mathcal{S}}\log\sigma\big(\hat{y}(b,i) - \hat{y}(b,j)\big) + \lambda\|\Theta\|_2^2, \tag{14}$$

where $\mathcal{S}$ is the set of the generated samples, Θ denotes all the parameters requiring for training, and λ is the regularization factor. Between each layer, we add a normalization layer and dropout-out layer to the embeddings, which stabilizes the training. We optimize the model with the mini-batch Adam [38] optimizer in Tensorflow.

F. Inductive Basket Recommendation

As we mentioned before, inductive BR is also crucial in real-world applications. Hence, we should evaluate the MITGNN model under the inductive setting. However, under the inductive setting, we do not have basket information until the testing process. In other words, we rank a list of items to a new basket b^*, but without knowing its existence or which items are in it during training. The problem is that we have no trained embedding of b^*. Hence, we should make an inference of the embedding of this new basket b^*. Recalling that in Eq. (4), we learn the basket embedding at the $(l+1)$-th layer by attentively aggregating the embeddings of multiple

intents. However, the embeddings of intents and the attention weights require the embedding of the basket at the l-th layer, as illustrated in Eq. (1) and Eq. (5). Therefore, we should firstly compute the embedding of b^* at the 0-th layer as:

$$\mathbf{e}_{b^*}^{(0)} = \mathbf{e}_u^{(0)} + \frac{1}{|\mathcal{N}_i(b^*)|} \sum_{i \in \mathcal{N}_i(b^*)} \mathbf{e}_i^{(0)}, \tag{15}$$

where $\mathbf{e}_u^{(0)}$ is the embedding of the corresponding user at the 0-th layer, and $\mathbf{e}_i^{(0)}$ is the embedding of items within b^*. We use $\mathcal{N}_i(b^*)$ denotes the items within the new basket $b^{(*)}$. The other variables are already trained and set as constants during testing. Thus, we can calculate the L-layer embedding of b^* by inferring with Eq. (4), Eq. (1), and Eq. (5) simultaneously.

V. Experiments

A. Dataset

We conduct experiments on two real-world datasets, the *Instacart* dataset and the dataset collected from the *Walmart* online grocery shopping website[3].

- **Instacart** is an online grocery shopping dataset, which is published by *instacart.com* [4]. 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.

After filtering baskets with less than 30 items and users with less than 5 baskets for the Instacart data, the final Instacart data contains $65,859$ nodes and $1,271,195$ interactions. We filter baskets with less than 40 items and users with less than 5 baskets for the Walmart data, leading to $85,315$ nodes and $1,225,155$ interactions in total. The details of the data statistics are summarized in Table III.

B. Experimental Setting

We evaluate performance based on Top-**K** ranking results, with the evaluation metrics being *Recall*@**K**, *Hit Ratio*@**K**, and *NDCG*@**K** [5], [39]. Experiments use **K**=$\{20, 40, 60, 80, 100\}$ for comparison. We first compute the scores between the basket b^* and all potential items. Then, we sort the items based on the scores. Finally, we compute the evaluation metrics by comparing the ranking results with the true items within basket b^*. The overall comparison is conducted under two settings: one is transductive BR; the other one is inductive BR, as we introduced in Sec. III. We compare MITGNN with the following baselines:

- **BPR-MF** [37]: we merge the items within the baskets for each user and sample positive and negative items for the user. BPR-MF completes the BR based solely on user embeddings.
- **Triple2vec** [3]: we sample triplets from baskets and train the user and item embeddings.
- **DREAM** [1]: it train user embeddings by modeling the dynamics of baskets.
- **GC-MC** [27]: we adopt the idea in [27] using the GCN model [19] to predict the basket-item relations in the graph.
- **R-GCN** [40]: A GNN model to complete the relations between nodes, which is used here to predict the basket-item relations.
- **NGCF** [6]: we modify it to predict the basket-item interactions. However, it has no module for the multi-intent pattern.

For BPR-MF, since it has no basket embedding to evaluate transductive and inductive basket recommendation, we still use the user embedding as the embedding for the next basket. Triple2vec adopts the idea that the basket embedding is the sum of the user embedding and the average of embeddings of items within the basket.

C. Parameter Settings

All models are validated on the performance of Recall@100. The embedding size d is 64 for all models. The learning rate for all models is searched from $\{10^{-5}, 5 \times 10^{-5}, 10^{-4}, 5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}\}$. The regularization factor λ is searched from $\{10^{-5}, 10^{-4}, \ldots, 10^{-1}\}$. For the layer number of GCMC, R-GCN, NGCF, and MITGNN, we search the layer number from $\{1, 2, 3, 4\}$. The number of intents for MITGNN is selected from $\{1, 2, 3, 4, 5\}$. We choose the MITGNN model with 3-layer structure for both datasets with learning rate of 0.0005 for Instacart data and 0.001 for Walmart data, regularization factor of 0.0001 for Instacart and 0.00001 for Walmart data, The number of intents T is tuned as 3 and 4 for Instacart and Walmart data, respectively. The details of the tuning process for the hyperparameters are discussed in Sec. V-F.

D. Transductive Recommendation

In this section, we conduct experiments regarding the transductive BR problem. For both datasets, 80% of items within the basket are used as training data. The remaining 20% of items within the basket are used for testing. The performance comparison on two datasets is presented in Figure 5. We have the following observations:

- MITGNN performs the best against all other baselines. On Instacart data, compared with the strongest baseline on each metric, MITGNN on average improves the Recall, HR, and NDCG by 12.03%, 9.43%, and 9.71%, respectively. On Walmart data, MITGNN on average improves the Recall, HR, and NDCG by 17.82%, 17.17%, and 10.75%, respectively. The improvements suggest that the multi-intent pattern is important in the BR problem.
- NGCF outperforms almost any other baseline on both datasets. It aggregates the neighboring information and learns multi-layer embeddings, rather than only using the last layers' embedding like GC-MC and R-GCN. However, R-GCN beats NGCF for *Hit Ratio*@80 and *Hit Ratio*@100, which suggests different relations should also be modeled. MITGNN performs the best as it can model

[3]https://www.walmart.com/grocery

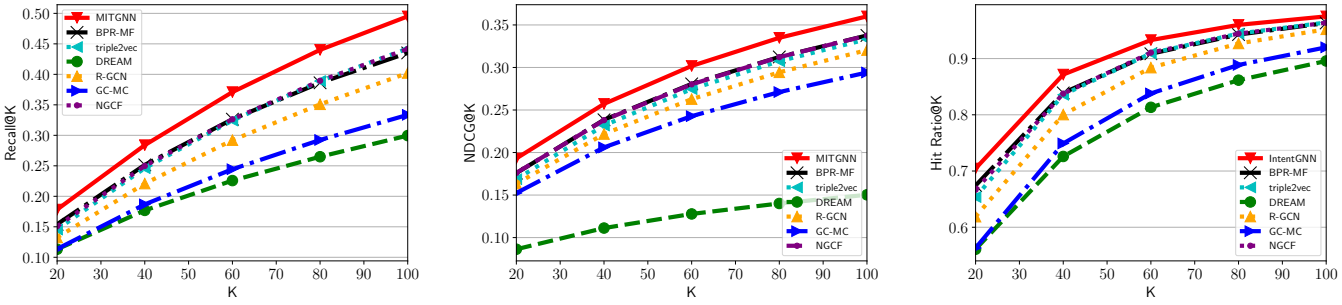
[4]https://www.instacart.com/datasets/grocery-shopping-2017

TABLE I: Inductive Basket Recommendation on Instacart

Method	*Recall*@**20**	*Recall*@**60**	*Recall*@**100**	*HR*@**10**	*HR*@**20**	*HR*@**30**	*NDCG*@**20**	*NDCG*@**60**	*NDCG*@**100**
BPR-MF	0.12185	0.27980	0.38629	0.88981	0.97004	0.98898	0.29311	0.46678	0.56721
Triple2vec	0.12532	0.27423	0.37476	0.88326	0.96694	0.98795	0.27538	0.45588	0.56051
DREAM	0.10841	0.22782	0.30663	0.83231	0.92101	0.95107	0.19380	0.21817	0.25982
GC-MC	0.09804	0.20555	0.27952	0.83333	0.93974	0.97624	0.27794	0.43145	0.51974
R-GCN	0.10147	0.21788	0.29679	0.84676	0.94180	0.97314	0.27375	0.43339	0.52435
NGCF	0.12389	0.27346	0.37504	0.89360	0.96832	0.99036	0.28978	0.46565	0.56724
MITGNN	**0.13321**	**0.30079**	**0.41234**	**0.90461**	**0.97417**	**0.99277**	**0.32173**	**0.49197**	**0.59701**

TABLE II: Inductive Basket Recommendation on Walmart

Method	*Recall*@**20**	*Recall*@**60**	*Recall*@**100**	*HR*@**10**	*HR*@**20**	*HR*@**30**	*NDCG*@**20**	*NDCG*@**60**	*NDCG*@**100**
BPR-MF	0.05339	0.12636	0.17972	0.57442	0.76543	0.85107	0.18325	0.32260	0.40903
Triple2vec	0.04077	0.11186	0.15998	0.39972	0.68647	0.81179	0.12562	0.27088	0.35430
DREAM	0.02913	0.06931	0.10127	0.40463	0.59959	0.70738	0.05675	0.06522	0.08233
GC-MC	0.03041	0.08527	0.12605	0.38272	0.61263	0.73072	0.12383	0.25075	0.33155
R-GCN	0.04133	0.11531	0.17078	0.44631	0.70137	0.82320	0.12500	0.26917	0.36091
NGCF	0.05233	0.12651	0.18135	0.55416	0.76660	0.86117	0.17709	0.31654	0.40540
MITGNN	**0.05757**	**0.15062**	**0.22466**	**0.61472**	**0.80992**	**0.90030**	**0.19086**	**0.33013**	**0.42062**



(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

Fig. 5: Performance of transductive basket recommendation w.r.t. different **K** on two datasets.

TABLE III: Dataset Statistics

Dataset	*#Edges*	*#Baskets*	*#Users*	*#Items*	*Density*
Instacart	1,271,195	32,201	2,904	30,754	0.0293%
Walmart	1,225,155	27,797	7,110	50,408	0.0168%

the multi-intent pattern and learn the relations between intents.

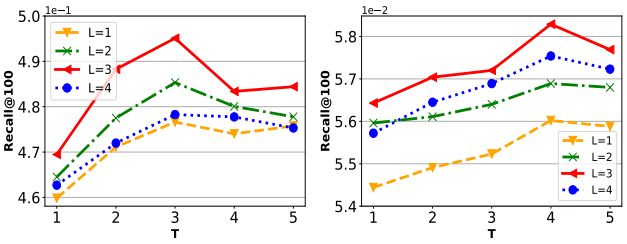
- Triple2vec and NGCF perform similarly to BPR-MF, which only models the interactions of user-item. This suggests that direct retrieval of item relations from the basket cannot improve the performance by only modeling user-item relations. However, they still outperform other direct aggregation methods such as R-GCN on the Instacart dataset, which indicates that the CF signal and the user factor are important for basket recommendation on the Instacart dataset. On Walmart data, R-GCN performs comparably with BPR-MF and triple2vec, which implies the difference of relations should be modelled.
- DREAM performs the worst among all the models as we find that personalized information is more important than sequential patterns in both datasets, which limits the effectiveness of the RNN structure in the DREAM model.
- The performance on Instacart is better than that of the Walmart. The first reason is that the Walmart data is much sparser than the Instacart. The former's density is 0.0168%, while the latter's density is 0.0293%. The

second reason is that each user in the Instacart data on average owns 11.1 baskets. However, compared with only 3.9 baskets in the Walmart data. The scarcity of personalized information leads to the poor performance.

E. Inductive Recommendation

In this section, we conduct inductive BR experiments. We predict items in the most recent baskets of users, where those baskets are not presented during training. Note that, for the current basket b^*, we perform inference of the basket embedding e_{b^*} as introduced in Sec. IV-F. We take out 5 items for aggregation and set the rest as ground truth. During testing, MITGNN learns basket embeddings before prediction. Performance is presented in Table I for Instacart data and Table II for Walmart data. Due to the space limitations, we only present the performance of Recall and NDCG@$\{20, 60, 100\}$ and the performance of Hit Ratio (HR)@$\{10, 20, 30\}$. The best results are in bold and the second-best results are underlined. We have the following observations:

- On Instacart data, compared with the strongest baseline on each metric, MITGNN improves the Recall, HR, and NDCG on average by 6.85%, 0.63%, and 6.80%, respectively. On Walmart data, MITGNN improves the Recall, HR, and NDCG on average by 16.92%, 5.74%, and 6.41%, respectively. The improvement comes from two perspectives: 1) our model aggregates the neighbors' embeddings to learn the new basket embedding, thus utilizing the item information in the new basket; 2) MITGNN discovers the multi-intent pattern in the basket graph and preserves the semantics in node embeddings, which strengthens the ability to solve BR problems.
- DREAM performs worst on Walmart data, which suggests that user's shopping patterns are not related to sequential patterns. On Instacart data, DREAM also performs worse than others, except for GC-MC. It implies that sequential patterns are not useful in BR. Since BPR-MF performs much better than DREAM, we should utilize CF signals, rather than sequential patterns.
- R-GCN and GC-MC perform worse than BPR-MF. It suggests that direct mean-pooling the neighboring information cannot assist in modeling the useful information for the inductive BR task. NGCF models the interactive information rather than direct mean-aggregation, thus performs better than R-GCN and GC-MC. MITGNN retrieves the interactive information for the BR task, therefore performs the best.
- Triple2vec directly averages the item embeddings for new baskets, but its performance is worse than BPR-MF, indicating that we need to train a basket aggregator for inductive learning. Our MITGNN trains the embeddings and the aggregators. During inference time of the basket embedding, we aggregate the embeddings of neighbors of the basket. Hence, it performs better.
- The value of each metric for the inductive BR in Table II is higher than the corresponding value for the transductive BR in Figure 5 on Walmart data. The reason is that for inductive BR, we only take out five items for aggregation, and each basket still contains at least 87.5% items as ground truth, compared with only 20% for tranductive BR. Some of these items have already been interacted with users in training baskets, thus are easy to predict. Hence, the values are higher on inductive BR as compared to transductive BR.



(a) Instacart (b) Walmart

Fig. 6: The performance of transductive BR w.r.t. the number of intents T and the number of layers L.

F. Effect of Intents and Layers

In this section, we discuss the effects w.r.t. the number of the intent T and the number of layers L. We discuss the results by presenting the *Recall@***100** of transductive BR on two datasets in Figure 6. In each figure, we draw four lines representing the number of layers increasing from $L = 1$ to $L = 4$. On each line are five points, indicating the number of intents varying from $T = 1$ to $T = 5$. On both datasets, when the number of intents $T > 1$, the performance is always better than $T = 1$. This proves that generating multiple hidden intents provides useful information. Moreover, we conclude that we should propagate high-order information in the BR problem, since better evaluation results are obtained when the number of layers $L > 1$ as compared to $L = 1$.

For the performance on Instacart in Figure 6a, we observe that the model consistently outputs the best results when $T = 3$, indicating the number of hidden intents on Instacart should be 3. As we increase the number of intents from $T = 1$ to $T = 5$, the performance curve goes up and then drops after reaching the peak. We also observe that as the number of layers increases from $L = 1$ to $L = 3$, performance also increases, which suggests that higher-order signals provide useful information for the BR problem. However, when increasing the number from $L = 3$ to $L = 4$, the performance drops dramatically, even worse than $L = 2$. It is consistent to our choice of the number of layers.

For the performance on Walmart in Figure 6b, there is gradual improvement when we increase T from 1 to 4, which proves the effectiveness of the generated hidden intents. We also observe the results are always the best when the number of intents $T = 4$, compared with other numbers of intents. The best number of T for Walmart data is 4, which is greater than $T = 3$ for Instacart. We find that each basket contains more items in Walmart data and is more diverse than Instacart data.

It implies more hidden intents are able to model more diversity for baskets. Moreover, $L = 3$ generates the highest value against other number of layers. It implies that the higher order provides useful information to the BR problem. Similarly, if we increase the layer number from 3 to 4, the performance drops. It implies too much higher-order information spoils the model's ability for BR.

VI. Conclusion

In this paper, we discuss the multi-intent pattern in BR problems, under both transductive and inductive settings. It is important to model the correlation between intents. Hence, we combine a translation-based model with a GNN model, which is the MITGNN. MITGNN models the intents as vectors translated from basket embeddings. Additionally, MITGNN has a GNN structure with multi-head aggregation to learn the relation vectors between intents and the basket. Moreover, MITGNN has a type-guided attention mechanism to attentively aggregate the hidden intents as user-guided and item-guided basket embeddings for the usage of a user aggregator and item aggregator, respectively. MITGNN is of multi-layer structure and can learn node embeddings by aggregating neighbors. It not only generates the embeddings of entities for transductive BR, but also trains aggregators for inductive BR. Extensive experiments on two large-scale real-world datasets prove the effectiveness of MITGNN on both transductive and inductive BR. Our study on the effect of the number of intents validates the benefits of modeling multi-intent patterns in BR problems.

VII. Acknowledgements

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

References

[1] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A dynamic recurrent model for next basket recommendation," in *SIGIR*. ACM, 2016, pp. 729–732.

[2] D. Le, H. W. Lauw, and Y. Fang, "Basket-sensitive personalized item recommendation," in *IJCAI*, C. Sierra, Ed. ijcai.org, 2017, pp. 2060–2066.

[3] 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.

[4] Z. Liu, M. Wan, S. Guo, K. Achan, and P. S. Yu, "Basconv: Aggregating heterogeneous interactions for basket recommendation with graph convolutional neural network," in *SDM*. SIAM, 2020, pp. 64–72.

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

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

[7] D. Xu, C. Ruan, J. Cho, E. Körpeoglu, S. Kumar, and K. Achan, "Knowledge-aware complementary product representation learning," in *WSDM*, 2020, pp. 681–689.

[8] J. Chang, C. Gao, X. He, Y. Li, and D. Jin, "Bundle recommendation with graph convolutional networks," *arXiv preprint arXiv:2005.03475*, 2020.

[9] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *SIGKDD*. ACM, 2018, pp. 1059–1068.

[10] F. Li, Z. Chen, P. Wang, Y. Ren, D. Zhang, and X. Zhu, "Graph intention network for click-through rate prediction in sponsored search," in *SIGIR*, 2019, pp. 961–964.

[11] J. Zhao, Z. Zhou, Z. Guan, W. Zhao, W. Ning, G. Qiu, and X. He, "Intentgc: a scalable graph convolution framework fusing heterogeneous information for recommendation," in *KDD*, 2019, pp. 2347–2357.

[12] M. M. Tanjim, C. Su, E. Benjamin, D. Hu, L. Hong, and J. McAuley, "Attentive sequential models of latent intent for next item recommendation," in *The Web Conference*, 2020, pp. 2528–2534.

[13] O. Barkan and N. Koenigstein, "Item2vec: neural item embedding for collaborative filtering," in *MLSP*. IEEE, 2016, pp. 1–6.

[14] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NIPS*, 2013, pp. 2787–2795.

[15] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *AAAI*, 2014.

[16] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *AAAI*, 2015.

[17] R. He, W.-C. Kang, and J. McAuley, "Translation-based recommendation," in *RecSys*, 2017, pp. 161–169.

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

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

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

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

[22] 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.

[23] Z. Liu, Y. Dou, P. S. Yu, Y. Deng, and H. Peng, "Alleviating the inconsistency problem of applying graph neural network to fraud detection," in *SIGIR*, 2020.

[24] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *CIKM*, 2020.

[25] Z. Liu, X. Li, L. He, H. Peng, and P. S. Yu, "Heterogeneous similarity graph neural network on electronic health record," in *BigData*. IEEE, 2020.

[26] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *ICLR*, 2014.

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

[28] 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*. ACM, 2018, pp. 974–983.

[29] X. Li, M. Zhang, S. Wu, Z. Liu, L. Wang, and P. S. Yu, "Dynamic graph collaborative filtering," in *ICDM*, 2020.

[30] Y. Song, A. M. Elkahky, and X. He, "Multi-rate deep learning for temporal recommendation," in *SIGIR*, 2016, pp. 909–912.

[31] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," *arXiv preprint arXiv:2002.07962*, 2020.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 5998–6008.

[33] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[34] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, vol. 30, no. 1, 2013, p. 3.

[35] Q. Liu, S. Wu, and L. Wang, "Deepstyle: Learning user preferences for visual recommendation," in *SIGIR*, 2017, pp. 841–844.

[36] W. Yu, H. Zhang, X. He, X. Chen, L. Xiong, and Z. Qin, "Aesthetic-based clothing recommendation," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 649–658.

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

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[39] P. Resnick and H. R. Varian, "Recommender systems," *Communications of the ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[40] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *ESWC*. Springer, 2018, pp. 593–607.