Transitivity-Encoded Graph Attention Networks for Complementary Item Recommendations

Jin Shang[1], Yang Jiao[1], Chenghuan Guo[1], Minghao Sun[2†],
Yan Gao[1], Jia Liu[3, 1], Michinari Momma[1], Itetsu Taru[1], Yi Sun[1]
[1]Amazon.com, Seattle, WA, United States
[2]Iowa State University, Ames, IA, United States
[3]The Ohio State University, Columbus, OH, United States
{imjshang, jaoyan, chenghg, yanngao, michi, itetsu, yisun}@amazon.com,
mhaosun@iastate.edu, liu@ece.osu.edu

***Abstract*—In e-commerce recommender systems, providing product suggestions to customers that are often bought together, which is called "complementary recommendation," not only improves customer experience but also boosts business impact. However, in practice, it is highly challenging to efficiently extract the complementary relations between the items due to noisy and low coverage of the co-purchased records in transaction datasets. To address these challenges, graph neural networks (GNN) have been increasingly adopted in complementary item recommendations thanks to their capabilities in integrating side-information and topological structures to extract these complex item relationships. However, most existing GNN-based methods fall short in learning better product complementary representation since they often utilize a simple one-to-one product-to-vector mapping strategy, which fails to describe the transitive logic of complementary items. To overcome this challenge, we propose a new GNN model called transitivity-encoded graph attention networks (TransGAT). To our knowledge, TransGAT is the first method that extends representation space by encoding the behavioral direction into embedding space in GNN and enabling mutual relationship extraction between complementary items. In order to better extract customer's intrinsic behavioral information, we further adopt the substitute information as the guidance by jointly learning complements and substitutes graphs and coupling them together. Moreover, several self-supervised data augmentation strategies are incorporated in our approach. Through evaluations on three real-world datasets, TransGAT consistently surpasses contemporary benchmarks, showcasing its prowess in complementary item recommendations.**

***Index Terms*—complementary recommendation, graph neural networks, graph attention networks, self-supervised learning**

I. Introduction

In today's e-commerce recommender systems, utilizing rich information of different types of products holds a great potential to boost business impacts and improve both customer shopping experiences and business revenue. Notably, many products on an e-commerce service are explicitly or implicitly related via different types of behavior relationships. Among a large variety of relationships between items, one of the most useful is the "complementary" relationship, which can often be used to help jointly recommend a high-quality set of relevant products for a customer to purchase together (e.g. cameras and lens). Thus, how to infer complementary items correctly is of great benefit that not only inspires customers with more potential needs but also induces better customer shopping experiences.

To facilitate the learning of complementary relationship between items, graph neural networks (GNN) has been widely adopted [1]–[4] in the e-commerce recommender system literature thanks to their capabilities in capturing the multiple complex connections over non-Euclidean spaces. For instance, in massive product graphs where the products are represented as nodes and their relationships are represented as edges, one usually defines the the product relationships by "co-purchase" and "co-view" edges for complementary and substitute recommendation tasks [2], [4]–[6]. However, due to the low coverage and noisy data of the co-purchase relationship records in transaction datasets, it remains highly challenging to efficiently extract the complementary relations even with GNN models. More specifically, most existing methods in this area utilize a GNN model as a mapping function $f(x)$ to learn a representation that transforms the product at each node in the graph to an embedded Euclidean space, i.e., each individual product x is transformed into a numerical representation in the form of a vector via a one-to-one mapping. As a result, the learning process is usually done in a pair-wise manner based on the distance between a product pair's representation $d(f(x), f(y))$, where a closer distance between two products indicates a stronger correlation between them. Further, the distance in the embedded space should preserve the product complementary relationship, so that complementary products should stay close to each other while non-complementary products should be separated far away from each other. By optimizing the loss function based on the distance d, one hopes to "converge" to a set of representations for all products (e.g., in the sense of reaching a Nash equilibrium).

However, a key limitation of using these traditional one-to-one mapping approaches for complementary relation learning stems from the potential *non-transitive property* of many complementary relationships. That is, a product's complementary neighbors are not necessarily complementary to each other. For example, a camera lens could have two complementary products that are both cameras. Clearly, these two cameras are not complementary to each other, since most users would typically purchase only one camera out of these two (in fact,

†This work was done when Minghao Sun was an intern with Amazon.com.

these two cameras are substitute items to each other rather being complementary). More formally, given two complementary product pairs (x, z) and (y, z), the traditional one-to-one mapping approaches necessarily yield close Euclidean distances $d(x, z)$ and $d(y, z)$, which further implies that the distance $d(x, y)$ is also close under the representation space in use. However, in reality, products x and y could be substitute, complementary or even non-related. Therefore, existing one-to-one product-to-representation mapping approaches are fundamentally inadequate to precisely describe the complementary relations in e-commerce recommender systems.

To address the non-transitive challenge, some recent works [7], [8] attempted to use multi-objective optimization to learn dual embedding space, while others (e.g., [9]) tried to use direct graph with additional transitivity information. We note that these existing methods suffer a similar limitation that the representations in the projected spaces could still be transitive, which implies that the non-transitive issue could still arise. In addition, the representations in the projected space are usually coupled by multi-objective learning, which is usually intractable since identifying Pareto-optimal solutions in multi-objective optimization is in general NP-hard if the objective functions are non-convex (a typical case in practice).

To address the non-transitive challenge of the complementary relationship between the products and avoid the above pitfalls in the existing works, we propose a new GNN model in this paper called transitivity-encoded graph attention networks (TransGAT). Our basic idea is that, instead of learning embedding in different spaces, we *encode the transitivity property into the representations*, thus the embeddings are learned within the same metric space. Our rationale is that, by encoding the non-transitivity property, the learned representations are able to capture complementary relationship better. The main technical contributions of this paper are summarized as follows:

- We first encode the transitivity directions into the representation space by extending the initial representation with augmented dimensions to reflect the complementary relationship positions of products. As a result, each product is extended with two types of initial inputs and mapped to two embeddings (key and value embeddings) to enable non-transitive embeddings for complementary products. Noted that all the embeddings are still projected into one metric space. By encoding transitivity, a product is extended into two products and embedded in the same metric space.
- We incorporate the product substitute information as guidance via a joint learning framework that helps the GNN model better extract the complementary product relationships. This is achieved by a new knowledge transfer layer that could project the source domain representation into the target domain as a forward knowledge transfer, and the projected representation could also be projected back as a backward knowledge transfer.
- To further enhance the model performance, we adopt both inductive and transductive self-supervised data augmentation strategies to enrich the dataset and reduce the data noise.

Our experiments with real-world datasets demonstrate the effectiveness of the TransGAT approach and show significant gain over performance of complementary relationship prediction.

II. Related Works

In this section, we provide a quick overview on the state of the art of complementary item recommendation in Section II-A. Then, in Section II-B, we provide necessary background of GNN in complementary recommendations. Lastly, we will discuss some recent studies using directed graphs to help GNN models extract the transitivity property in Section II-D

A. Complementary Item Recommendation

In recent years, an extensive line of research focuses on extracting the complementary relationship between products from various types of features, including but not limited to content features (images and text) and behavior features (co-view and co-purchase by the customers). For example, McAuley et al. [5] incorporated product descriptions and reviews as well as other features between products (e.g., price difference, average rating difference, and manufacture difference) and integrate them into a topic model, which transforms complementary item recommendations into a link prediction task. Later, by making the assumption that the complementary relationship is determined by both perceived feature matching and item quality matching, Zhang et al. [10] proposed a neural complementary recommender that jointly learns complementary item relationships and user preferences. In contrast, Rakesh et al. [11] proposed a linked variational autoencoder to learn the latent features of products by conditioning on the observed relationship between them. In [1], Hao et al. also proposed a P-companion approach that first uses an encoder-decoder network to predict multiple complementary product types and then projects the embedding of query product to each predicted complementary product type subspace, which further learns the complementary relationship based on the distant supervision labels.

B. GNN for Complementary Product Recommendation

Graph neural networks [12], [13] (e.g., GraphSAGE and Graph Attention Networks (GAT)) have been used in a wide range of applications as they are able to capture the local information from non-Euclidean structures. As products on a recommender system can be viewed as nodes and the relationship between nodes can be modeled as edges, GNN-based models are ideal for product relationship predictions and recommendations [14]–[17]. For complementary product recommendation, Yan et al. [3] proposed an approach that uses a graph attention network and a sequential behavior transformer to model product relations and user preferences. An active line of research [2], [4], [6], [18] exploit the substitute information to build GNN-based models to jointly learn both complementary and substituable relationships between products in order to extract the intrinsic correlations to boost the prediction performance of complementary product recommendation.

We note that all aforementioned approaches utilize a GNN model to map each node to a single embedding. This strategy works fine with transitive relationships such as substitutable relationship (substitutable products of a product are typically substituable to each other). However, these existing approaches become problematic when it comes to non-transitive product relationships. For example, the complementary products of a product are not necessarily complementary to each other. In this case, however, the "one-to-one" mapping strategy will yield close embeddings for this product's complementary neighbors, which could violate the ground-truth relations and mislead the model to learn wrong embeddings for these non-transitive product relationship prediction.

C. Learning Dual Embedding Space for Complementary Item Recommendations

To address the transitivity chanllenges in Section II-B, some studies propose to learn two sets of item embeddings for complementary recommendation. For example, the methods proposed in [8] and [7] learn two encoders, where each encoder projects the same product initial embedding into different individual spaces, i.e. an item-in embedding and item-out embedding. They use the item-in embedding captures the knowledge-aware feature between complement products, while using the item-out embedding to capture the user-item interactions. Subsequently, both embeddings are learned together via multi-objective optimization. However, within each learned embedding space, the representations are still symmetric and transitive. In addition, the computational procedures for exploring the Pareto front is intractable since identifying Pareto-optimal solutions in multi-objective optimization is in general NP-hard if the objective functions are non-convex (a typical case in practice).

D. Directed-Graph-Based Approaches for Transitivity-Aware Relationship Extraction

Recently, there have also been studies focusing on using directed graphs models to facilitate the transitivity property extraction to address the transitivity chanllenges. For example, Ou et al. [19] proposed the HOPE algorithm, which uses directed graphs to preserve asymmetric transitivity by approximating high-order proximities. Similarly, Zhou et al. [20] introduced the asymmetric proximity preserving (APP) graph embedding method via random walk with restart, which captures both asymmetric and high-order similarities between node pairs. In a more recent study, Virinchi et al. [9] proposed Blade, a GNN model that utilizes two separate embeddings for each product with in-neighbor and out-neighbor graphs to capture lexical information for substitutable product recommendations. They sampled and aggregated features from its in-neighbor and out-neighbor node embeddings separately and used the graph attention network to update the central node. We note, however, that the aforementioned methods require having datasets that cover the transitivity relationship information, which may not attainable in real-world scenarios, since one usually uses co-purchase as the ground truth for complementary relationship.

TABLE I: Notations.

Notation	Description
$\mathcal{G}$	input graph
$\mathbf{V}, \mathcal{E}$	node/edge set of $\mathcal{G}$
$v_i^{(k)}$	i-th node for key sub space in relationship graph
$e_{ij}^{(kv)}$	edge for i as key and j as value
$e_{ij}^{(vk)}$	edge for i as value and j as key
$\boldsymbol{h}_i^{(k)}$	i-th node input embedding for key sub space
$\hat{\boldsymbol{h}}_i^{(k)}$	i-th node embedding before knowledge transfer
$\ddot{\boldsymbol{h}}_i^{(k)}$	i-th node embedding after knowledge transfer
N	the number of products
M	number of relationship product pairs
d, d'	dimensions of input/output graph embeddings
$\alpha_{ij}^{(vk)}$	attention score between $(v_i^{(k)}, v_j^{(v)})$ for node $v_i^{(k)}$
γ', γ, λ	weight parameters in knowledge transfer layer
$\boldsymbol{a}$	shared weight in graph attention
$\mathcal{N}_i^{(k)}$	sampled neighbors in value sub space for node $v_i^{(k)}$
$W^{(k)}, W^{(v)}$	key/value weight matrices for graph attention

When the data only contains undirected relationship, the Blade model will be reduced to GAT.

III. Model

In this section, we will present our TransGAT model. We first discuss how we handle the non-transitive property in complementary product recommendation in III-A. Then, we present our TransGAT model with updating rules in III-B. Next, in III-C, we present the overall framework with knowledge transfer from product substitue information. Finally, we explain how we adopt self-supervise learning for data augmentation in III-D. The most important notations in this paper are summarized in Table I.

A. Transitivity Encoding

Most e-commerce recommmender systems for complementary product recommendation, as shown in Fig. 1(a), assume a machine learning model corresponding to a representation function $f(\cdot)$ to learn a representation under a metric space (X, d), where X is the representation set and d is a metric on X. Given three products A, B, C and their learned representations $f(R_A)$, $f(R_B)$, $f(R_C) \in X$ (here R_A, R_B, R_C are the input embeddings of A, B, and C), the distance of product representation pair $d(f(R_B), f(R_C))$, satisfies the triangle rules: $d(f(R_B), f(R_C)) \leq d(f(R_B), f(R_A)) + d(f(R_A), f(R_C))$.

Suppose that there are two complementary pairs (A, B), (A, C) as the given complementary relationship labels, the conventional approach attempts to minimize the distance $d(f(R_A), f(R_B))$ and $d(f(R_A), f(R_C))$. By optimizing the loss function in a pairwise fashion, one hopes to achieve a convergence for all product representations (e.g., in the sense of Nash equilibrium). However, the distance $d(f(R_B), f(R_C))$ is bounded by $d(f(R_B), f(R_C)) \leq d(f(R_B), f(R_A)) + d(f(R_A), f(R_C))$ due to the triangle inequality. Meanwhile, the true relationship between the product pair (B, C) could be either substitutable, complementary, or non-relational. Thus, the conventional one-to-one mapping approach may violate

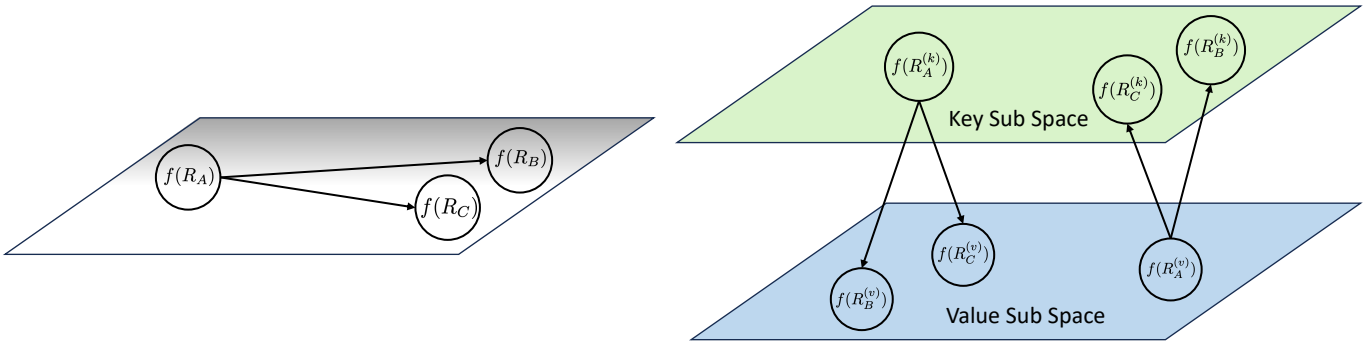

Fig. 1: Comparison of the traditional one-to-one mapping (a) and transitivity encoding (b) approaches to learn the representations for complementary products.

the reality and prevents the model from learning a better representation for complementary products, which in turn confuses the model in the learning process and degrade overall the quality of the learned representations.

To address the above non-transitivity challenge in complementary product recommendations, we propose a new transitivity encoding approach. Instead of constructing a one-to-one product-to-vector mapping, we first augment the product initial embeddings with additional dimensions that are shown in Fig. 2. For example, given an initial product embedding $R_A = [0.3, 0.5, 0.2]$ for product A, we consider the position of product A in the complementary relation rather than just itself. For a complementary product pair (A, B), we define the first position in the pair as the key, (i.e., A is key) and the second position is the value (i.e., B is value). Then, we encode its position in the complementary product pairs into the representation that we are learning. Assume the additional dimensions for the key embedding $R^{(k)} = [1.0, 0.0]$ and the value embedding $R^{(v)} = [0.0, 1.0]$, we concatenate them with the product initial embedding R_A before feeding them into the model. Thus, product A eventually has two final initializations: 1) $R_A^{(k)} = [R^{(k)} \| R_A] = [1.0, 0.0, 0.3, 0.5, 0.2]$ for product A as key in complementary pair (A, B) and 2) $R_A^{(v)} = [R^{(v)} \| R_A] = [0.0, 1.0, 0.3, 0.5, 0.2]$ for product A as value in complementary pair (B, A), where $\|$ represents the concatenate operation. Similarly, we have the same initializations for B and all the other products in our product graph. Thus, each product will have both key and value embeddings as inpput into the representation function $f(\cdot)$. For the remaining part of the product initial embedding, just as shown in Fig. 2, we adopt the product indices' one-hot embedding as well as image and content features and concatenate them. For features, we adopt a self-supervised technique to enrich our data, which will be discussed later in Section III-D.

The rationale of our transitivity encoding approach originates from how to learn a better word representation. It is well-known that word2vec does not perform well when the word has multiple meanings and only one vector is used to represent it. However, BERT and subsequent large language models could achieve a much better performance since they learn the representations not only from the word itself, but also from the context in the sentence where the word belongs to. By adopting our transitivity encoding approach as shown in Fig. 1(b), even though product pairs (A, B) and (A, C) are complementary and the distances of their learned embeddings $d(f(R_A^{(k)}), f(R_B^{(v)}))$, $d(f(R_A^{(k)}), f(R_C^{(v)}))$, $d(f(R_B^{(k)}), f(R_A^{(v)}))$, and $d(f(R_C^{(k)}), f(R_A^{(v)}))$ are close, the distances of $d(f(R_B^{(k)}), f(R_C^{(v)}))$ and $d(f(R_C^{(k)}), f(R_B^{(v)}))$, which denote the complementary relation of product pair (B, C), could still be large. This is because the distances of $d(f(R_B^{(k)}), f(R_C^{(v)}))$ and $d(f(R_C^{(k)}), f(R_B^{(v)}))$ will not constitute a complementary relationship and they remain dependent on the learning process of the model. As will be shown later, this transitivity encoding approach will lead to a better learned representation for complementary recommendations.

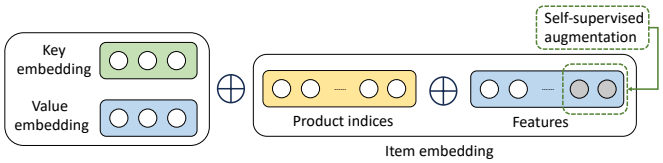

Fig. 2: Product initial embedding construction.

It is also worth noting that the product relationship could be directed in some scenarios. For example, consider the product pair (A, B) where B is complementary to A while A is not complementary to B. In this case, instead of minimizing both $d(f(R_A^{(k)}), f(R_B^{(v)}))$ and $d(f(R_B^{(k)}), f(R_A^{(v)}))$, the model is able to only minimize the distance of $d(f(R_A^{(k)}), f(R_B^{(v)}))$. As a result, our transitivity-encoding approach is suitable for both directed and undirected relationships. Compared with several models that could handle directed relationship only or undirected relationship only, our approach is more general and easy to deploy in real-world recommender systems.

B. Graph Attention with Key-Value Updating

In this paper, we adopt the graph attention networks (GAT) [13] as the backbone network in our TransGAT model.

However, since we have two types of product nodes in the graph, the node embedding updating rules are more complex compared to the standard GAT. Given a complementary product set that contains N products with M complementary pairs, we build a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the node set $\mathcal{V}$ has $2N$ product nodes in total. Here, each product $i \in \{1, ..., N\}$ has two product nodes $v_i^{(k)}$ and $v_i^{(v)}$. Similarly, the edge set $\mathcal{E}$ has $2M$ edges in total, each of which represents a complementary relationship between the two nodes this edge connects. Here, we assume the complementary relationship is undirected. Thus, each complementary product pair (i, j) contain two edges: one is $e_{ij}^{(kv)} = (v_i^{(k)}, v_j^{(v)})$ for product i as key and product j as value, and the other is $e_{ij}^{(vk)} = (v_i^{(v)}, v_j^{(k)})$ for product i as value and product j as key. We note that it is straightforward to apply our TransGAT model to directed relationship settings if datasets with such information are provided.

Different from a normal GAT layer, given the input node embeddings for both key and value (i.e., $\{\boldsymbol{h}_i^{(k)}\}_{i=1}^N \in \mathbb{R}^d$ and $\{\boldsymbol{h}_j^{(v)}\}_{j=1}^N \in \mathbb{R}^d$), the output of a layer in TransGAT can be written as: $\{\hat{\boldsymbol{h}}_i^{(k)}\}_{i=1}^N \in \mathbb{R}^{d'}$ and $\{\hat{\boldsymbol{h}}_j^{(v)}\}_{j=1}^N \in \mathbb{R}^{d'}$. Instead of directly using its sampled neighbors to update the central node's embedding, we use the neighbors in the value subspace to update the central node in key subspace and vice versa, which is shown in Fig. 3. Here, we name our GAT-variant layer as the **key-value GAT backbone layer**, which is the backbone of our TransGAT model in Section III-C and Fig.4. In order to compute the attention scores, we adopt a similar setting akin to GAT, which only calculates the attention of the central node's sampled first-order neighbors, and computes the linear transformation by two shared weighted matrices $W^{(k)}$, $W^{(v)} \in \mathbb{R}^{d'} \times \mathbb{R}^d$. Then, the *softmax* normalized attention scores is calculated as:

$$\begin{cases} \alpha_{ij}^{(kv)} = \frac{\exp(\text{LeakyReLu}(\boldsymbol{a}^\top [W^{(v)}\boldsymbol{h}_i^{(v)} \| W^{(k)}\boldsymbol{h}_j^{(k)}]))}{\sum_{j' \in \mathcal{N}_i^{(k)}} \exp(\text{LeakyReLu}(\boldsymbol{a}^\top [W^{(v)}\boldsymbol{h}_i^{(v)} \| W^{(k)}\boldsymbol{h}_{j'}^{(k)}]))}, \\ \alpha_{ij}^{(vk)} = \frac{\exp(\text{LeakyReLu}(\boldsymbol{a}^\top [W^{(k)}\boldsymbol{h}_i^{(k)} \| W^{(v)}\boldsymbol{h}_j^{(v)}]))}{\sum_{j' \in \mathcal{N}_i^{(v)}} \exp(\text{LeakyReLu}(\boldsymbol{a}^\top [W^{(k)}\boldsymbol{h}_i^{(k)} \| W^{(v)}\boldsymbol{h}_{j'}^{(v)}]))}, \end{cases} \tag{1}$$

for neighbors j as key and central node i as value ($\alpha_{ij}^{(kv)}$) and neighbors j as value and central node i as key ($\alpha_{ij}^{(vk)}$), respectively, where $\boldsymbol{a} \in \mathbb{R}^{2d'}$ is the shared parameter to perform self-attention on nodes and $\|$ denotes the concatenation operation. The updating process is shown in Fig. 3. The sampled first-order neighbors in the key subspace $\mathcal{N}_i^{(k)}$ are used for updating central node $v_i^{(v)}$ in its value subspace, while the sampled first-order neighbors in the value subspace $\mathcal{N}_i^{(v)}$ are used for updating central node $v_i^{(k)}$ in the key subspace. It is worth noting that one also needs to include the central nodes themselves into the update following from the attention mechanism, which will preserve the previous learned information.

Upon obtaining the normalized attention scores, we have the final output of our key-value GAT backbone layers by first adopting an attention score based on a weighted linear combination of all the neighbor embeddings, and then applying an activation function $\sigma(\cdot)$ to induce nonlinearity. Thus, we have the output key and value embeddings for product i as follows:

$$\begin{cases} \hat{\boldsymbol{h}}_i^{(k)} = \sigma(\sum_{j \in \mathcal{N}_i^{(v)}} \alpha_{ij}^{(vk)} W^{(v)} \boldsymbol{h}_j^{(v)} + \alpha_{ii}^{(kk)} W^{(k)} \boldsymbol{h}_i^{(k)}), \\ \hat{\boldsymbol{h}}_i^{(v)} = \sigma(\sum_{j \in \mathcal{N}_i^{(k)}} \alpha_{ij}^{(kv)} W^{(k)} \boldsymbol{h}_j^{(k)} + \alpha_{ii}^{(vv)} W^{(v)} \boldsymbol{h}_i^{(v)}), \end{cases} \tag{2}$$

where the second term on the right-hand-side of each equation in (2) corresponds to the central node itself as illustrated by Fig. 3.

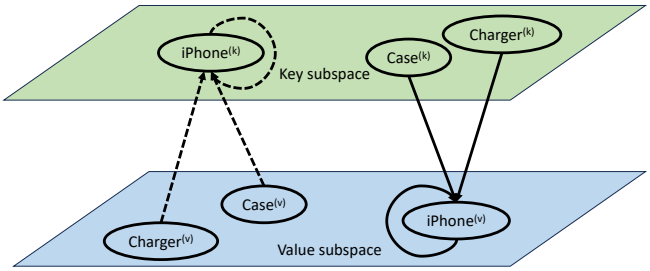

Fig. 3: Key-value GAT backbone layer: updating between the key and value subspace. Here, we use sampled first-order neighbors $\mathcal{N}_i^{(k)}$ = {charge$^{(k)}$, case$^{(k)}$} in the key subspace to update node $v_i^{(v)}$ = {iPhone$^{(v)}$} in the value subspace, and use the sampled first-order neighbors $\mathcal{N}_i^{(v)}$ = {charge$^{(v)}$, case$^{(v)}$} in the value subspace to update node $v_i^{(k)}$ = {iPhone$^{(k)}$} in the key subspace. Noted that the central nodes themselves are also included during the update. Both the updating rules that are described in Eq. (2) mathematically constitute the **key-value GAT backbone layer** component shown in Fig. 4.

C. Joint Learning with Knowledge Transfer

With the key-value GAT backbone layer introduced in the previous subsection, we are now in a position to present our TransGAT model that consists of and is built upon the key-value GAT backbone layers. The TransGAT model structure is illustrated in Fig. 4. Inspired by the observations in [2], [4], [6] that jointly learning substitutable and complementary relationships significantly boosts both complementary and substitute product recommendation performances, we propose a *two-tower-type* architecture. Here, each tower learns the complementary and substitutable relationship separately, and then the learned knowledge is transferred via the knowledge transfer layer after the key-value GAT backbone layer.

The knowledge transfer layer is similar to that in [2], which leverages the ideas of dual learning [21] and Cycle-GAN [22]. Specifically, the knowledge transfer layer first uses a linear transformation to project the learned complementary embeddings into the substitute space and vice-versa. Next, the projected embeddings are projected back to their original spaces.

Since we have two subspaces for complementary embeddings, we also apply the transitivity encoding strategy to the

substitute graph learning tower, so that the complementary embeddings could be easily transferred with substitute embeddings. As a result, after each key-value GAT backbone layer, we have a modified knowledge transfer layers as a forward transfer layer:

$$\begin{cases} \ddot{\boldsymbol{h}}_s^{(k)} = \gamma_1' \hat{\boldsymbol{h}}_s^{(k)} + (1-\gamma_1') \cdot g_{c\to s}^{(k)}(\hat{\boldsymbol{h}}_c^{(k)}), \\ \ddot{\boldsymbol{h}}_c^{(k)} = \gamma_2' \hat{\boldsymbol{h}}_c^{(k)} + (1-\gamma_2') \cdot g_{s\to c}^{(k)}(\hat{\boldsymbol{h}}_s^{(k)}), \\ \ddot{\boldsymbol{h}}_s^{(v)} = \gamma_3' \hat{\boldsymbol{h}}_s^{(v)} + (1-\gamma_3') \cdot g_{c\to s}^{(v)}(\hat{\boldsymbol{h}}_c^{(v)}), \\ \ddot{\boldsymbol{h}}_c^{(v)} = \gamma_4' \hat{\boldsymbol{h}}_c^{(v)} + (1-\gamma_4') \cdot g_{s\to c}^{(v)}(\hat{\boldsymbol{h}}_s^{(v)}), \end{cases} \tag{3}$$

and a backward transfer layer:

$$\begin{cases} \ddot{\boldsymbol{h}}_s^{(k)} = (1-\gamma_1-\lambda_1) \cdot \hat{\boldsymbol{h}}_s^{(k)} + \gamma_1 g_{c\to s}^{(k)}(\hat{\boldsymbol{h}}_c^{(k)}) + \lambda_1 g_{c\to s}^{(k)}(\ddot{\boldsymbol{h}}_c^{(k)}), \\ \ddot{\boldsymbol{h}}_c^{(k)} = (1-\gamma_2-\lambda_2) \cdot \hat{\boldsymbol{h}}_c^{(k)} + \gamma_2 g_{s\to c}^{(k)}(\hat{\boldsymbol{h}}_s^{(k)}) + \lambda_2 g_{s\to c}^{(k)}(\ddot{\boldsymbol{h}}_s^{(k)}), \\ \ddot{\boldsymbol{h}}_s^{(v)} = (1-\gamma_3-\lambda_3) \cdot \hat{\boldsymbol{h}}_s^{(v)} + \gamma_3 g_{c\to s}^{(v)}(\hat{\boldsymbol{h}}_c^{(v)}) + \lambda_3 g_{c\to s}^{(v)}(\ddot{\boldsymbol{h}}_c^{(v)}), \\ \ddot{\boldsymbol{h}}_c^{(v)} = (1-\gamma_4-\lambda_4) \cdot \hat{\boldsymbol{h}}_c^{(v)} + \gamma_4 g_{s\to c}^{(v)}(\hat{\boldsymbol{h}}_s^{(v)}) + \lambda_4 g_{s\to c}^{(v)}(\ddot{\boldsymbol{h}}_s^{(v)}), \end{cases} \tag{4}$$

where γ', γ, λ are the learned weight parameters for integration, and $g_{s\to c}^{(k)}(\cdot)$, $g_{s\to c}^{(v)}(\cdot)$, $g_{c\to s}^{(k)}(\cdot)$, $g_{c\to s}^{(v)}(\cdot)$ are the projection functions that typically correspond to single-layer neural networks in practice. The subscripts s and c here mean substitutable and complementary, respectively.

The knowledge transfer layers enable the aggregation of product representations across complementary and substitutable structures. As shown in Fig. 4, we can stack several layers of key-value GAT and knowledge transfer structures (we use two in our experiments as we find it achieves the best performance and complexity trade-off).

After we obtain the final output embeddings, we use the cross-entropy loss to minimize the distance between the labelled complementary and substitutable products. Thus, for product i, we have the complementary and substitutable losses as follows:

$$\begin{cases} \mathcal{L}_c(i) = & -\sum_{j\in\mathcal{N}_{c,i}^{(v)}} \log(\sigma(\ddot{\boldsymbol{h}}_{c,i}^{(k)} \cdot \ddot{\boldsymbol{h}}_{c,j}^{(v)})) \\ & -\sum_{j\in\mathcal{N}_{c,i}^{(k)}} \log(\sigma(\ddot{\boldsymbol{h}}_{c,i}^{(v)} \cdot \ddot{\boldsymbol{h}}_{c,j}^{(k)})) \\ & -\sum_{j'\in\Omega_{c,i}^{(v)}} \log(\sigma(-\ddot{\boldsymbol{h}}_{c,i}^{(k)} \cdot \ddot{\boldsymbol{h}}_{c,j'}^{(v)})) \\ & -\sum_{j'\in\Omega_{c,i}^{(k)}} \log(\sigma(-\ddot{\boldsymbol{h}}_{c,i}^{(v)} \cdot \ddot{\boldsymbol{h}}_{c,j'}^{(k)})), \\ \mathcal{L}_s(i) = & -\sum_{j\in\mathcal{N}_{s,i}^{(v)}} \log(\sigma(\ddot{\boldsymbol{h}}_{s,i}^{(k)} \cdot \ddot{\boldsymbol{h}}_{s,j}^{(v)})) \\ & -\sum_{j\in\mathcal{N}_{s,i}^{(k)}} \log(\sigma(\ddot{\boldsymbol{h}}_{s,i}^{(v)} \cdot \ddot{\boldsymbol{h}}_{s,j}^{(k)})) \\ & -\sum_{j'\in\Omega_{s,i}^{(v)}} \log(\sigma(-\ddot{\boldsymbol{h}}_{s,i}^{(k)} \cdot \ddot{\boldsymbol{h}}_{s,j'}^{(v)})) \\ & -\sum_{j'\in\Omega_{s,i}^{(k)}} \log(\sigma(-\ddot{\boldsymbol{h}}_{s,i}^{(v)} \cdot \ddot{\boldsymbol{h}}_{s,j'}^{(k)})), \end{cases} \tag{5}$$

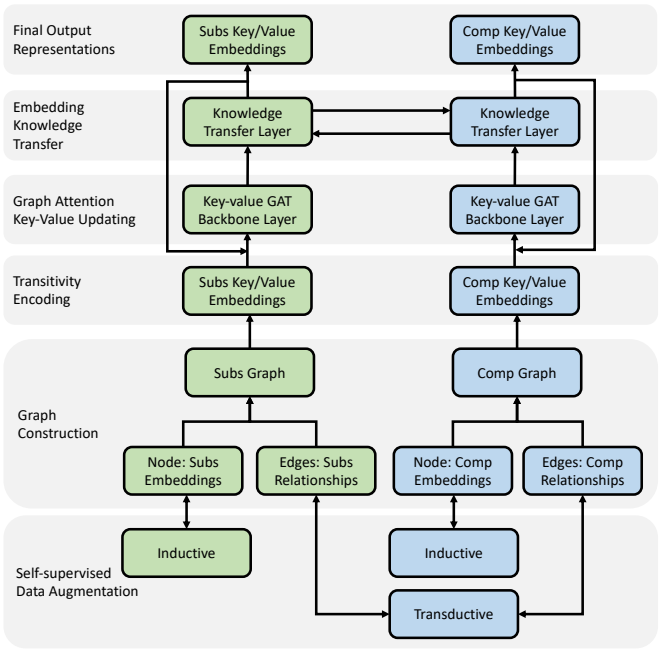

Fig. 4: Model structure of TransGAT.

where $\mathcal{N}_{c,i}^{(k)}$ and $\mathcal{N}_{c,i}^{(v)}$ are the sampled neighboring nodes of the central node i in key and value subspaces, respectively, that are actually positive samples for complementary relation; and $\Omega_{c,i}^{(k)}$ and $\Omega_{c,i}^{(v)}$ are the sampled negative nodes. The substitutable neighbors also follow the same notation format as mentioned above. Since we have N products in total, the final average loss of our TransGAT model can be written as:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(\mathcal{L}_c(i) + \mathcal{L}_s(i)). \tag{6}$$

For prediction, we can compute the inner product score of the learned embeddings, and then use the mean to obtain the predicted scores of complementary and substitutable relationships for a product pair (i, j) as follows:

$$\begin{cases} p_c(i,j) = \frac{1}{2}[\sigma(\ddot{\boldsymbol{h}}_{c,i}^{(k)} \cdot \ddot{\boldsymbol{h}}_{c,j}^{(v)}) + \sigma(\ddot{\boldsymbol{h}}_{c,i}^{(v)} \cdot \ddot{\boldsymbol{h}}_{c,j}^{(k)})], \\ p_s(i,j) = \frac{1}{2}[\sigma(\ddot{\boldsymbol{h}}_{s,i}^{(k)} \cdot \ddot{\boldsymbol{h}}_{s,j}^{(v)}) + \sigma(\ddot{\boldsymbol{h}}_{s,i}^{(v)} \cdot \ddot{\boldsymbol{h}}_{s,j}^{(k)})]. \end{cases} \tag{7}$$

D. Self-Supervised Data Augmentation

In recent year, Self-supervised learning (SSL) has gained more and more interest as a solution to tackle the data sparsity and noisy challenges in recommender systems [23]–[25]. Since co-purchase and co-view records are often used as the ground truth label for complementary and substitutable items, respectively, low coverage and noisy data have also become two major challenges for learning a better representation. To address these challenges, we adopt two self-supervised learning data augmentation strategies to enrich and denoise our labelled data before we construct the graph structure. By doing so, we can take advantage of both inductive and transductive approaches in self-supervised learning as illustrated in Fig. 4. In what follows, we will zoom into these two approaches.

TABLE II: Dataset statistics.

	Electronic	Toys & Games	Sports & Outdoors
Items	260144	238347	390953
Sub. Edges	2338810	1323370	2225258
Comp. Edges	871266	1693304	2594038

1) The Transductive Approach: Since we have the information substitutable products, we can extend the connections with the following three rules, which are inspired by [4], [6]:

- *Rule #1:* If product A and B are substitute to each other, product A's substitutable neighbors are also B's substitutes.
- *Rule #2:* If product A and B are substitute to each other, product A's complementary neighbors are also B's complements.
- *Rule #3:* If product A and B are complementary to each other, product A's substitutable neighbors are also B's complements.

Instead of utilizing them in the learning process, we directly extend the complementary and substitutable relationships before constructing the graph based on them. Since all nodes already exist in the graph and we only increase the edges between them, this augmentation approach is transductive.

2) The Inductive Approach: Our inductive approach is similar to the recent self-supervised learning approaches in [23], [26]–[28], which mask and randomize some of the features to enrich the data. In our setting, we adopt the image embeddings of the product as well as some other features (e.g., title, brand name, size, color, and price). We mask the color feature and replace it with some random numbers to create some dummy product nodes, and let them still retain the complementary and substitutable relationships with other products. Since we extend the dataset by adding new synthetic products nodes, this data augmentation approach is inductive.

IV. Experiments

In this section, we will first describe the datasets we use, baseline methods, and the evaluation metrics. Then, we show the main results on complementary product recommendations as well as substitutable product recommendations. Lastly, we will conduct various ablation studies and investigate the impacts of our proposed transitivity encoding technique.

A. Experimental Setup

1) Dataset: We conduct our experiments using the public Amazon review dataset [29], [30] with three categories: Electronics, Toys & Games, and Sports & Outdoors. Following previous studies [2], [4]–[6], we use the co-purchase and co-view relationships provided in the dataset as the ground truth labels of complementary and substitutable products. The statistics of these three categories are listed in Table II.

2) Baseline Methods: We compare TransGAT with the following state-of-the-art baseline approaches:

- **GAT:** Graph Attention Network [13] is a widely used graph neural network model. GAT derives the hidden representation of each node by employing self-attention mechanisms over its neighboring nodes. In our experimental setup, we uses two separate GAT models for both complementary and substitutable tasks to update the node representations.
- **GraphSAGE:** GraphSAGE [12] uses graph convolutional networks (GCN) to learn the representations (focusing more on large graphs). GraphSAGE uses the *sample and aggregate* strategy over the local neighborhood nodes. In our experiments, we also use two separate GraphSAGE models for both complementary and substitutable tasks.
- **HAN:** Heterogeneous Graph Attention Network (HAN) [31] captures complex structures and rich semantics within a heterogeneous graph. HAN utilizes both node-level attention and semantic-level attention mechanisms to learn the significance of individual nodes and meta-paths. Similar to the setings in GAT and GraphSAGE, in our experiments, we also use two separate HAN models for both complementary and substitutable tasks.
- **HAT:** Hyperbolic Graph Attention Network (HAT) is the state-of-the-art model [4] to adopt GNN on both complementary and substitutable product recommendations. HAT operates within hyperbolic spaces and performs both feature aggregation and transformation. HAT incorporates the attention mechanism to aggregate node features within a hyperbolic space.
- **DEGCN**: Decoupled Graph Convolutional Network (DEGCN) [2] generates item embeddings within distinct embedding spaces associated with both complementary and substitutable product relationships. DEGCN also provides a knowledge transfer structure to jointly learn the complements and substitutes.

3) Parameter Settings: For fair comparisons, we use the same input graph with self-supervised data augmentation and the same dimension of node embedding in our experiments. We also provide results of ablation studies on data augmentation in Section IV-C. We set the dimension of hidden node embedding to 64 and the dimension of the output to 32. We used the Adam algorithm for optimization except for HAT, because HAT is on a hyperbolic space. For HAT, we use the Riemann-SGD optimizer. We set the batch size as 1024 and the learning rate as 0.01. All methods are implemented by the DGL library [32] and PyTorch.

4) Evaluation Metrics: We use inner product in Eq. (7) to obtain the score of two product nodes to predict the complementary and substitutable product relationships. For each product node, we randomly select one edge for each in both substitute and complement graphs to construct the test dataset, and then use the remaining edges for training. In the testing stage, for each product, we follow [4] while combine its ground truth paired node with 10000 sampled negative nodes, and use HR@K (Hit Rate@K), MRR@K (Mean Reciprocal Rank@K) and NDCG@K (Normalized Discounted Cumulative Gain@K) as the evaluation metrics. We report the average numbers for all these performance metrics.

TABLE III: Overall performance of TransGAT compared with state-of-art baseline methods for both complement and substitute on three categories of public Amazon dataset.

Model	Electronic			Toys & Games			Sports & Outdoors		
	HR@10	MRR@10	NDCG@10	HR@10	MRR@10	NDCG@10	HR@10	MRR@10	NDCG@10
Complementary									
GAT	0.6724	0.3436	0.4184	0.7005	0.3520	0.4278	0.6335	0.3049	0.3781
GraphSAGE	0.6652	0.3345	0.4177	0.6964	0.3708	0.4461	0.6270	0.3121	0.3829
HAN	0.5542	0.2434	0.3106	0.6621	0.3362	0.4101	0.5475	0.2326	0.3001
HAT	0.5017	0.1855	0.2512	0.6397	0.2932	0.3692	0.5418	0.2138	0.2830
DEGCN	0.6094	0.2596	0.3345	0.6561	0.3041	0.3816	0.6243	0.2628	0.3216
TransGAT (Ours)	**0.7658**	**0.4269**	**0.5078**	**0.7735**	**0.4424**	**0.5221**	**0.7744**	**0.4157**	**0.4998**
Substitute									
GAT	0.7092	0.3468	0.4254	0.7962	0.4336	0.5153	0.7845	0.4564	0.5322
GraphSAGE	0.7415	0.3919	0.4696	0.7953	0.4511	0.5315	0.7609	0.4258	0.5019
HAN	0.6148	0.2692	0.3422	0.7302	0.3686	0.4479	0.6445	0.3131	0.3761
HAT	0.6676	0.2973	0.3760	0.7258	0.3464	0.4285	0.6530	0.2892	0.3363
DEGCN	0.6915	0.3014	0.3842	0.7440	0.3404	0.4268	0.6586	0.2725	0.3603
TransGAT (Ours)	**0.8689**	**0.5363**	**0.6162**	**0.8682**	**0.5062**	**0.6098**	**0.8721**	**0.5576**	**0.6344**

TABLE IV: Ablation study results.

Electronic	Complementary			Substitute		
	HR@10	MRR@10	NDCG@10	HR@10	MRR@10	NDCG@10
GAT (No-Enrich)	0.6786	0.3083	0.3894	0.6055	0.2588	0.3315
GAT	0.6724	0.3436	0.4184	0.7092	0.3468	0.4254
GAT+Knowledge Transfer	0.6825	0.3577	0.4323	0.7279	0.3638	0.4434
GAT+Transitivity Encoding	0.7603	0.3959	0.4803	0.7962	0.4211	0.5047
TransGAT (Ours)	**0.7658**	**0.4269**	**0.5078**	**0.8689**	**0.5363**	**0.6162**
TransGraphSAGE	0.7445	0.3879	0.4703	0.8573	0.4942	0.5786

B. Main Experimental Results

The average results for predicting substitutable and complementary product relationships are presented in Table III. As shown in the table, our TransGAT method outperforms all the baseline methods across all three datasets. Particularly, we achieve significant improvements on the complement product prediction due to our proposed transitivity encoding technique. Our results also demonstrate that we obtain a better representation that can distinguish and predict the complementary relationship precisely, which showcases the superiority of our approach over the conventional one-to-one product-to-vector mapping technique that is adopted by all the baseline methods.

It is also worth mentioning that, surprisingly, the substitutalbe product task also enjoys the benefits from our proposed transitivity encoding technique, and achieves significant improvements over baseline methods. In theory, the substitutable product relationship is transitive since a product's substitutable neighbors are typically substitutable to each other. Thus, it is quite surprising that our proposed transitivity encoding technique is beneficial for substitutable product recommendations. This can be explained by the fact that we are using the co-view records of the products as the ground-truth label of substitutable products in our experiments. In practice, co-view records are typically quite noisy (a lot of co-view records are actually not exactly substitutable products: some are complementary and some are not even related). In other words, a product's substitutable neighbors are not necessarily substitutable to each other. Thus, substitutable relationship may not be transitive if one uses co-view records as the label for substitutable products. Therefore, our proposed transitivity encoding technique help resolve this challenge in the substitute task in our experiments, which yields improved representations in the substitutable product recommendation task.

C. Ablation Studies

To further understand the effectiveness of our TransGAT model, we conduct ablation studies on how each component of the TransGAT model benefits the final performance and how much performance gain they contribute to. We use the following additional settings for our TransGAT model that involve the knowledge transfer layer and dual embedding:

- **GAT (No-Enrich):** Since all baseline methods and TransGAT use the same input graph that is already enriched by self-supervised data augmentation, we test using a vanilla graph as the input to see the differences in results.
- **GAT + Knowledge Transfer:** This setting corresponds to the GAT model that adopts the joint learning framework as shown in Fig. 4. The only difference here comparing with our TransGAT is that this model does not have the transitivity encoding component and still uses the traditional one-to-one product-to-vector mapping.
- **GAT + Transitivity Encoding:** This setting corresponds to the GAT model that adopts the joint learning framework as shown in Fig. 4. The only difference here comparing to our TransGAT model is that it does not have the embedding knowledge transfer component. The output of the graph

attention key-value updating component is directly sent to the final output representation component and we only enforce the correlation between complementary and substitutable by optimizing their losses simultaneously. Here we use two separate GAT + Transitivity Encoding models for both complementary and substitutable tasks.

- **TransGraphSAGE:** To illustrate the generalization of our approach, we use GraphSAGE with transitivity encoding rather than GAT as our backbone network, and we also use the knowledge transfer layer to jointly learn both complementary and substitutable product tasks, which is similar to our TransGAT model and the structure is the same as Fig. 4. We only change the GAT to GraphSAGE.

The average ablation study results are summarized in Table IV. Our ablation study results clearly show that self-supervised data augmentation is beneficial in general, which evidenced by the performance improvement from GAT (No-Enrich) to GAT. The knowledge transfer layer and the joint learning strategy also boost the performance on all metrics, which can be seen from the performance improvements from GAT to GAT+Knowledge Transfer. The transitivity encoding strategy significantly helps and contributes the most in model performance improvements. Also, the comparable results from TransGraphSAGE prove the significant impact of learning a better representation for complementary product recommendation by encoding transitivity. The TransGraphSAGE results also show that our TransGAT approach is a general framework, which can be applied on other models with product relationships that are non-transitive.

D. TransGAT with Direct Graph Settings

As mentioned earlier in Section III-B, it is straightforward to apply our TransGAT model to directed relationship settings if datasets with such information are available. This is because such symmetric/asymmetric information is simply product pair labels, which can be easily handled by our TransGAT model. For example, suppose that product A's co-purchase items are $[B, C]$. In the symmetric setting, we can label product key/value embedding pairs $(A^k, B^v), (A^k, C^v), (A^v, B^k), (A^v, C^k)$ as complementary pairs. By contrast, in the asymmetric settings, we can only label product key/value embedding pairs $(A^k, B^v), (A^k, C^v)$ as complementary pairs. Then, our TransGAT model learns the geometric information from connected nodes. As a result, node A^k is connected to node B^v and C^v and node A^v are connected to node B^k and C^k in symmetric settings, while in asymmetric settings, only node A^k is connected to node B^v and C^v.

We have also conducted experiments with our model in asymmetric settings, which can be represented as directed graphs by assuming the data are asymmetric. The results are shown in Table V. We can observe that the results for the asymmetric graph setting are not as strong as those of the symmetric dataset setting. Since the Amazon review dataset is sparse and does not contain the transitivity information between products, there could be a large number of missing links between those items that have complementary relationships.

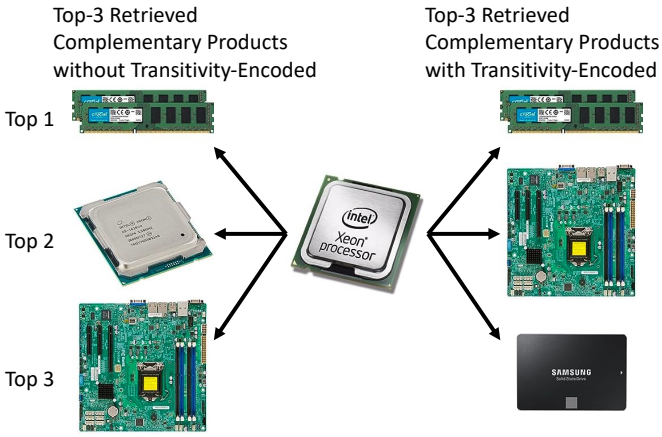

Fig. 5: Example for the impacts of transitivity encoding.

Hence, the symmetric setting in our experiments could help enrich the information and lead to good performance.

TABLE V: Performance between symmetric and asymmetric graph setting with Amazon Electronic dataset.

Amazon Electronic	HR@10	MRR@10	NDCG@10
asymmetric	0.6464	0.2880	0.3669
symmetric	**0.7658**	**0.4269**	**0.5078**

E. The Impact of Transitivity Encoding

Lastly, we study the impact of transitivity encoding on the performance of our TransGAT model. Consider the example with the CPU processor shown in the middle of Fig. 5. The top-3 retrieved complementary products identified by TransGAT with and without transitivity encoding are shown on the right and left sides, respectively. We can see that the memory, motherboard, and solid-state drive are the top 3 retrieved products complementary to the query product, which is a CPU processor. However, TransGAT without transitivity encoding returns a CPU processor (cf. the second in the list), which is actually a substitute product to the query product. This shows that the representation without using transitivity encoding cannot distinguish the complementary relationship precisely. The processor in the second place of the left list is a complement of memory and motherboard, which are first and third items in the left list.

V. Conclusion

In this paper, we proposed a new transitivity-encoded graph attention networks (TransGAT) model for learning complementary products. TransGAT is built upon a new transitivity-encoding strategy to address the non-transitive challenge of complementary product recommendation and learn an accurate representation. We also build a joint learning framework that takes the substitute information as guidance to help learn the complementary representation by knowledge transfer. We also utilized several self-supervised learning data augmentation

techniques to enhance the TransGAT model performance. Our experiments on the public Amazon dataset demonstrated the effectiveness of TransGAT and showed significant performance gain in predicting complementary products. Future works include generalizing the transitivity encoding technique into heterogeneous graphs, replacing inner product with contrastive loss metrics to learn the representation, and exploring the possibility of model structure changes that could handle non-transitive relationship rather than only encoding them into embeddings.

REFERENCES

[1] J. Hao, T. Zhao, J. Li, X. L. Dong, C. Faloutsos, Y. Sun, and W. Wang, "P-companion: A principled framework for diversified complementary product recommendation," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2517–2524.

[2] Y. Liu, Y. Gu, Z. Ding, J. Gao, Z. Guo, Y. Bao, and W. Yan, "Decoupled graph convolution network for inferring substitutable and complementary items," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2621–2628.

[3] A. Yan, C. Dong, Y. Gao, J. Fu, T. Zhao, Y. Sun, and J. McAuley, "Personalized complementary product recommendation," in *Companion Proceedings of the Web Conference 2022*, 2022, pp. 146–151.

[4] Z. Zhou, T. Wang, L. Hou, X. Zhou, M. Ma, and Z. Ding, "Decoupled hyperbolic graph attention network for modeling substitutable and complementary item relationships," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 2763–2772.

[5] J. McAuley, R. Pandey, and J. Leskovec, "Inferring networks of substitutable and complementary products," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 785–794.

[6] Z. Wang, Z. Jiang, Z. Ren, J. Tang, and D. Yin, "A path-constrained framework for discriminating substitutable and complementary products in e-commerce," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018, pp. 619–627.

[7] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Product knowledge graph embedding for e-commerce," in *Proceedings of the 13th international conference on web search and data mining*, 2020, pp. 672–680.

[8] D. Xu, C. Ruan, J. Cho, E. Korpeoglu, S. Kumar, and K. Achan, "Knowledge-aware complementary product representation learning," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 681–689.

[9] S. Virinchi and A. Saladi, "Blade: Biased neighborhood sampling based graph neural network for directed graphs," in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 2023, pp. 42–50.

[10] Y. Zhang, H. Lu, W. Niu, and J. Caverlee, "Quality-aware neural complementary item recommendation," in *Proceedings of the 12th ACM conference on recommender systems*, 2018, pp. 77–85.

[11] V. Rakesh, S. Wang, K. Shu, and H. Liu, "Linked variational autoencoders for inferring substitutable and supplementary items," in *Proceedings of the twelfth acm international conference on web search and data mining*, 2019, pp. 438–446.

[12] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

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

[14] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 974–983.

[15] L. Zheng, Z. Zhen, W. Wang, C. Dong, M. Momma, and Y. Sun, "Heterogeneous graph neural networks with neighbor-sim attention mechanism for substitute product recommendation," 2021.

[16] T. Jian, F. Yang, Z. Zuo, W. Wang, M. Momma, T. Zhao, C. Dong, Y. Gao, and Y. Sun, "Multi-task gnn for substitute identification," in *Companion Proceedings of the Web Conference 2022*, 2022, pp. 228–231.

[17] T. Zhou, M. Momma, C. Dong, F. Yang, C. Guo, J. Shang, and J. Liu, "Multi-task learning on heterogeneous graph neural network for substitute recommendation," 2023.

[18] L. Dai, Y. Yin, C. Qin, E. Chen, and H. Xiong, "Decomposing complementary and substitutable relations for intercorporate investment recommendation," in *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2022, pp. 909–914.

[19] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 1105–1114.

[20] C. Zhou, Y. Liu, X. Liu, Z. Liu, and J. Gao, "Scalable graph embedding for asymmetric proximity," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.

[21] D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T.-Y. Liu, and W.-Y. Ma, "Dual learning for machine translation," *Advances in neural information processing systems*, vol. 29, 2016.

[22] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[23] C. Huang, X. Wang, X. He, and D. Yin, "Self-supervised learning for recommender system," in *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, 2022, pp. 3440–3443.

[24] J. Yu, H. Yin, X. Xia, T. Chen, J. Li, and Z. Huang, "Self-supervised learning for recommender systems: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 1, pp. 335–355, 2023.

[25] Y. Jiao, F. Yang, Y. Chen, Y. Gao, J. Liu, and Y. Sun, "Rethinking sequential relationships: Improving sequential recommenders with inter-sequence data augmentation," in *Companion Proceedings of the ACM on Web Conference 2024*, 2024, pp. 641–645.

[26] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[27] T. Gao, X. Yao, and D. Chen, "Simcse: Simple contrastive learning of sentence embeddings," *arXiv preprint arXiv:2104.08821*, 2021.

[28] T. Yao, X. Yi, D. Z. Cheng, F. Yu, T. Chen, A. Menon, L. Hong, E. H. Chi, S. Tjoa, J. Kang *et al.*, "Self-supervised learning for large-scale item recommendations," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 4321–4330.

[29] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015, pp. 43–52.

[30] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *proceedings of the 25th international conference on world wide web*, 2016, pp. 507–517.

[31] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The world wide web conference*, 2019, pp. 2022–2032.

[32] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," *arXiv preprint arXiv:1909.01315*, 2019.