Decoupled Graph Convolution Network for Inferring Substitutable and Complementary Items

Yiding Liu[1], Yulong Gu[2], Zhuoye Ding[2], Junchao Gao[2], Ziyi Guo[2], Yongjun Bao[2], Weipeng Yan[2]
[1]Baidu Inc., Beijing, China [2]JD.com, Beijing, China
{liuyiding.tanh,guyulongcs}@gmail.com
{dingzhuoye,gaojunchao5,guoziyi,baoyongjun,paul.yan}@jd.com

ABSTRACT

Inferring substitutable and complementary items is an important and fundamental concern for recommendation in e-commerce websites. However, the item relationships in real-world are usually heterogeneous, posing great challenges to conventional methods that can only deal with homogeneous relationships. More specifically, for this problem, there is a lack of in-depth investigation on 1) decoupling item semantics for modeling heterogeneous item relationships, and at the same time, 2) incorporating mutual influence between different relationships. To fill this gap, we propose a novel solution, namely Decoupled Graph Convolutional Network (DecGCN), to solve the problem of inferring substitutable and complementary items. DecGCN is designed to model item substitutability and complementarity in *separated* embedding spaces, and is equipped with a two-step integration scheme, where inherent influences between 1) different graph structures and 2) different item semantics are captured. Our experiments on three real-world datasets demonstrate that DecGCN is more effective than the state-of-the-art baselines for the problem at hand. We also conduct offline and online A/B tests on large-scale industrial data, where the results show that DecGCN is effective to be deployed in real-world applications. We release the codes at `https://github.com/liuyiding1993/CIKM2020_DecGCN`.

CCS CONCEPTS

• **Information systems → Personalization**; **Recommender systems**.

KEYWORDS

Graph Convolution Network; Recommender Systems

ACM Reference Format:
Yiding Liu[1], Yulong Gu[2], Zhuoye Ding[2], Junchao Gao[2], Ziyi Guo[2], Yongjun Bao[2], Weipeng Yan[2]. 2020. Decoupled Graph Convolution Network for Inferring Substitutable and Complementary Items . In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3340531.3412695

This work was conducted during Yiding's internship at JD.com.

1 INTRODUCTION

With the proliferation of e-commerce platforms (e.g., Amazon, JD.com, Taobao), rich information about various kinds of items (i.e., products) has become available. These items are explicitly or implicitly connected via different types of relationships, e.g., co-viewed or co-purchased by the same user, forming massive item graphs, where the items are represented as nodes and their relationships are represented as edges. Exploiting such graph structure is of great significance to understand the characteristics of items, which can facilitate a wide spectrum of applications, such as Recommender Systems [7, 8, 22, 24, 41] and online advertisements [36–38].

Among the multiplex relationships between items, *substitutable* and *complementary* relationships are two representatives, as they are mostly interested and explored by e-commerce users [17]. For example, an online shopper would often view different T-shirts (i.e., substitutable items) before he/she buys one, and would also be interested in a matching pair of jeans (i.e., complementary items). Substitutable and complementary items of users' recently interacted (*e.g.,* clicked or purchased) products are two main sources for candidate generation, which is one of the most crucial stage in recommender systems [17, 24]. Thus, it is of great benefit to study the problem of *inferring substitutable and complementary items.*

In the past decade, extensive studies are conducted for effectively solving this problem [12, 17, 19, 26, 32–35], and the majority of them focus on modeling the content features of items (e.g., visual and textual content) using different methods, such as Latent Dirichlet Allocation (LDA) [17] and Variational Auto-Encoder (VAE) [19] , while much less attention is paid to better perceive the graph structure. Recent advances on Network Embedding [1, 16] and Graph Neural Networks (GNNs) [2, 10, 27, 39] have shown powerful capacity on modeling graph data, and thus shed new light on this task, where a handful of studies leverage the graph structure of items and achieve promising results [26, 29, 33]. Notably, Ying *et al.* [29] is the first to apply Graph Neural Networks (GNNs) for inferring substitutable items, and shows superior performance in real-world production environments, which can be attributed to its unified modeling of both item features and local graph structures.

However, despite the progress made by the previous studies, the **heterogeneity** of item relationships is still largely under-exploited. To be more specific, most of the existing literature miss to incorporate relationship heterogeneity from the following two perspectives:

- **Decoupling multifaceted item semantics**. Some existing models usually rely on a single shared representation for each item (e.g., the set of latent topics [1, 17]) to infer both types of relationships (i.e., substitute and complement), while neglecting the fact that different relationships stretching out from an item could result from the expression of its different aspects [4, 14, 15]. Considering the substitutability and complementarity as two aspects of item, it is non-robust and noisy to simply fuse item substitutability and complementarity together in a single shared item representation [15]. To this end, it would be promising to *decouple item substitutability and complementarity* that can respectively model the two relationships.

- **Modeling influences between different item semantics**. Intuitively, the complementary information of an item could also provide important clues about its substitutability, and vice versa. For example, if two t-shirts are substitutable, i.e., having similar style and size, they are likely to share many common complementary items (e.g., jeans).

To address these two limitations, we contribute a novel Decoupled Graph Convolutional Network (DecGCN) model, for the inference of substitutable and complementary items. To address the first limitation, the DecGCN is designed to include two separated sub-GCNs that decouple the item semantics. More specifically, the two sub-GCNs are trained separately and output **two** embedding vectors for each item, which respectively encode its substitutability and complementarity. Compared with conventional methods that are built up on shared item representations (i.e., directly fuse different item semantics), DecGCN is of more flexible when dealing with item relationships with high complexity and variety.

To address the second limitation, we introduce a *two-step knowledge integration process* in DecGCN, which considers the mutual influence between different graph structures and different item semantics. In particular, the designed integration process consists of *structural integration* and *semantic integration.* We first incorporate structural integration in each convolutional layer of the model, where a co-attention mechanism is adopted to attentively integrate both substitute and complement neighborhoods to formulate item embeddings. After that, we introduce semantic integration, which enables knowledge transfer between the two different semantics, such that the item substitutability and complementarity can mutually benefit each other for better modeling the two types of relationships. Our contributions can be summarized as follows:

- We are the first to propose a GCN method that decouples item semantics for inferring substitutable and complementary items.
- We propose a novel two-step knowledge integration scheme in Decoupled GCN. We propose to 1) integrate two types of local neighborhoods using co-attention mechanism, and 2) integrate two types of semantics (i.e., substitutability and complementarity) via knowledge transfer mechanism.
- Extensive experiments on three public datasets show that our method achieves better performance than the state-of-the-art baselines. We also conduct A/B test on the production data of JD.com and demonstrate the superiority of our method for the real-world applications.

2 RELATED WORK

2.1 The Inference of Substitutable and Complementary Items

Previous studies usually formulate the Substitutable and Complementary Items inference problem as a supervised link prediction problem [17, 19, 26, 34], and try to 1) adopt different representation learning methods for perceiving item features, and 2) build a prediction model upon the item representations to infer substitutable and complementary relationships. For example, McAuley *et al.* [17] propose the system Sceptre, which uses Latent Dirichlet Allocation to learn the topic distribution of items from users' reviews, and exploits logistic regression to predict substitutable and complementary relationships based on the learnt representations.

Recently, a handful of studies start to pay more attention on exploiting Graph Neural Networks [29] for this problem. Notably, Ying *et al.* [29] is the first to deploy Graph Convolutional Network (GCN) in real production environment, and achieves remarkable performance for inferring substitutable contents. Another recent proposal is Cen *et al.* [1], which aims at modeling attributed multiplex networks, and shows the state-of-the-art performance for inferring substitutable and complementary items.

2.2 Graph Neural Networks

Graph Neural Networks [6, 9, 13, 21] have achieved state-of-the-art performance in modeling graph based data. However, most of the conventional GNN methods are designed for homogeneous graphs. In real-world applications, the graph usually comes with multi-types of nodes and edges, also widely known as heterogeneous information network (HIN) [20].

Heterogeneous Graph Neural Networks. To deal with the multiple types of nodes and edges in the heterogeneous graph, some Heterogeneous Network Embedding methods [3, 16, 20] and Heterogeneous Graph Neural Networks [1, 2, 5, 23, 25, 30, 31] based approaches are proposed recently. For example, Zhang *et al.* [31] propose a heterogeneous graph neural network model called HetGNN, which jointly considers node heterogeneous contents encoding, type-based neighbors aggregation, and heterogeneous information combination to obtain the final node embedding. Wang *et al.* [25] propose a Heterogeneous Graph Attention Network based framework called HAN, which leverages attention mechanism to aggregate features from meta-path based neighbors hierarchically.

Disentangled Graph Neural Networks. Recently, some initial work [4, 14, 15] reveal that a node could be multifaceted, and attempt to learn multiple representations for each node. For example, Epasto *et al.* [4] propose Splitter, an unsupervised embedding method that allows nodes in a graph to have multiple embeddings to better encode their participation in multiple overlapping communities. Liu *et al.* [14] propose a polysemous embedding approach for modeling multiple facets of nodes, and Ma *et al.* [15] propose a disentangled graph convolutional network (DisenGCN) to learn disentangled node representations. These solutions could provide robustness and explainable to latent node representations. However, they still miss to explicitly model the mutual influences among different aspects.

3 PROBLEM FORMULATION

In e-commerce sites, the multiple relationships among items form a heterogeneous graph. In this paper, our task is to generate high-quality representations of items that can be used for inferring substitutable and complementary items.

3.1 Heterogeneous Graph

We represent the *multiple relationships* among items as a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of n nodes (*i.e.,* items), and the edges $\mathcal{E} = \mathcal{E}^{(s)} \cup \mathcal{E}^{(c)}$ represent substitutable (i.e., $\mathcal{E}^{(s)}$) and complementary (i.e., $\mathcal{E}^{(c)}$) relationships between nodes. The nodes are associated with attributes $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, where $\mathbf{x}_i$ represents the attributes (*e.g.,* metadata or content information) of node v_i. Note that we consider the relationships as *undirected* edges in this paper. We can consider that the heterogeneous graph $\mathcal{G}$ is composed of two subgraphs:

- **Substitutable item-item subgraph.** This subgraph is denoted as $\mathcal{G}^{(s)} = (\mathcal{V}, \mathcal{E}^{(s)})$, where the edges $\mathcal{E}^{(s)}$ represent the substituable relationships between items. If two items v_i and v_j are substitutable (*e.g.,* frequently co-viewed), there will be a substituable edge $e_{ij}^{(s)}$ between them.
- **Complement item-item subgraph.** This subgraph is denoted as $\mathcal{G}^{(c)} = (\mathcal{V}, \mathcal{E}^{(c)})$, where the edges $\mathcal{E}^{(c)}$ represent the complement relationships between items. If two items v_i and v_j are complement (*e.g.,* frequently co-purchased), there will be a complement edge $e_{ij}^{(c)}$ between them.

3.2 Problem Definition

Following previous studies [17], we formulate the Substitutable and Complementary Items inference task as a link prediction problem. Our goal is to leverage both the nodes' attributes as well as the structure of the heterogeneous graph to generate high-quality embeddings of items, which are then used for inferring substitutable and complementary items. In this paper, to model the multifaceted semantics of nodes, we propose to learn **two** sets of latent node embeddings for $\mathcal{V}$, denoted as $\mathbf{Z}^{(s)} = \{\mathbf{z}_1^{(s)}, \mathbf{z}_2^{(s)}, ..., \mathbf{z}_n^{(s)} | \mathbf{z}_i^{(s)} \in \mathbb{R}^{1\times d}, i = 1, 2..., n\}$ and $\mathbf{Z}^{(c)} = \{\mathbf{z}_1^{(c)}, \mathbf{z}_2^{(c)}, ..., \mathbf{z}_n^{(c)} | \mathbf{z}_i^{(c)} \in \mathbb{R}^{1\times d}, i = 1, 2..., n\}$, which are decoupled representations of nodes that can be used for inferring substitutable and complementary items, respectively. Here, d represents the dimensionality of the latent embeddings.

Definition 1 (**The substitutable and complementary items inference problem.**). *Given the heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which contains items as nodes $\mathcal{V}$ (associated with attributes $\mathcal{X}$) and their substitute and complement relationships $\mathcal{E} = \{\mathcal{E}^{(s)}, \mathcal{E}^{(c)}\}$ as edges, the problem aims to learn two sets of embeddings $\mathbf{Z}^{(s)}$ and $\mathbf{Z}^{(c)}$ for the nodes $\mathcal{V}$, such that the probability of nodes v_i and v_j being substitutable or complementary can be estimated by $p^{(s)}(\mathbf{z}_i^{(s)}, \mathbf{z}_j^{(s)})$ and $p^{(c)}(\mathbf{z}_i^{(c)}, \mathbf{z}_j^{c)})$, where $p^{(s)}$ and $p^{(c)}$ represent the prediction function (*e.g., *dot product) for the substitutable and complementary relationships respectively.*

4 METHODOLOGY

In this section, we elaborate the proposed solution. In particular, Section 4.1 introduces the preliminaries of GCN; Section 4.2 presents the overall architecture of DecGCN; Sections 4.3 and 4.4 introduce the two-step knowledge integration paradigm in detail.

4.1 Graph Convolutional Network

Our method is based on a widely-used GCN method, namely GraphSage [10, 29]. It uses localized convolutional modules to generate multi-layer embeddings for nodes, which is depicted as follows.

Convolutional layer. The core of GCN is a localized convolution operation. For each node v_i, the convolution operation is designed to aggregate and process the information of its neighborhood (denoted as $\mathcal{N}_i$). In particular, it first aggregates the representations of a set of its neighbors using a pooling operation (e.g., a element-wise mean or weighted sum), where a non-linear layer is subsequently employed to formulate the output embedding of v_i. Formally, the convolution operation in the l-th convolutional layer on v_i can be defined as

$$\mathbf{h}_i^l = \sigma(\mathbf{W}^l \cdot \mathrm{CONCAT}(\mathbf{h}_i^{l-1}, \mathbf{h}_{\mathcal{N}_i}^l)), \tag{1}$$

$$\mathbf{h}_{\mathcal{N}_i}^l = \gamma(\{\mathbf{h}_j^{l-1}, j \in \mathcal{N}_i\} \tag{2}$$

where $\mathbf{h}_{\mathcal{N}_i}^l$ denotes the aggregated feature vector of $\mathcal{N}_i$, γ is the aggregation (i.e., pooling) operation, $\mathbf{W}^l$ represents the weight matrix and σ represents the activation function (i.e., ReLU). A GCN model usually stacks multiple convolutional layers (i.e., l={1,2,...,L}) over the shallow feature embeddings (i.e., $\mathbf{h}_i^0$), which extract high-level information of both the item features and graph structure into the final-layer node embedding (i.e., $\mathbf{z}_i = \mathbf{h}_i^L$).

4.2 Overview of Decoupled GCN

We propose a Decoupled GCN with a carefully designed knowledge integration process. The overview of our model is illustrated in on the left side of Figure 1, where the backbone of our proposed method is two separated sub-GCNs, which output two sets of item embeddings as

$$\mathbf{Z}^{(s)} = \mathrm{GCN}^{(s)}(\mathcal{G}^{(s)}, \mathcal{X}) \quad \text{and} \quad \mathbf{Z}^{(c)} = \mathrm{GCN}^{(c)}(\mathcal{G}^{(c)}, \mathcal{X}) \tag{3}$$

Note that the two sub-GCNs take different subgraph structures (i.e., $\mathcal{G}^{(s)}$ and $\mathcal{G}^{(c)}$) and item attributes $\mathcal{X}$ as inputs, and produce decoupled embeddings (i.e., $\mathbf{Z}^{(s)}$ and $\mathbf{Z}^{(c)}$) in different semantics. Compared with those methods that simply fuse different item semantics in a shared item representation, our method encodes different item semantics in different latent space, and thus is more flexible for modeling complex substitute and complement relationships.

However, separated sub-GCNs are unable to capture the influence *between* the two types of relationships. Therefore, to further incorporate such information, we propose to jointly optimize the two sub-GCNs with a two-step knowledge integration paradigm, which contains:

- *structural integration*, considering the influence between two types of subgraph structures (i.e., $\mathcal{G}^{(s)}$ and $\mathcal{G}^{(c)}$) on modeling item semantics, and
- *semantic integration*, considering the influence between two types of semantics (i.e., $\mathbf{z}_i^{(s)}$ and $\mathbf{z}_i^{(c)}$) for each item.

Structural integration. To consider the mutual influence between $\mathcal{G}^{(s)}$ and $\mathcal{G}^{(c)}$, we propose to integrate additional subgraph structure in both sub-GCNs. More formally, we consider the modeling of subgraph $\mathcal{G}^{(s)}$ (or $\mathcal{G}^{(c)}$) can be further refined by leveraging

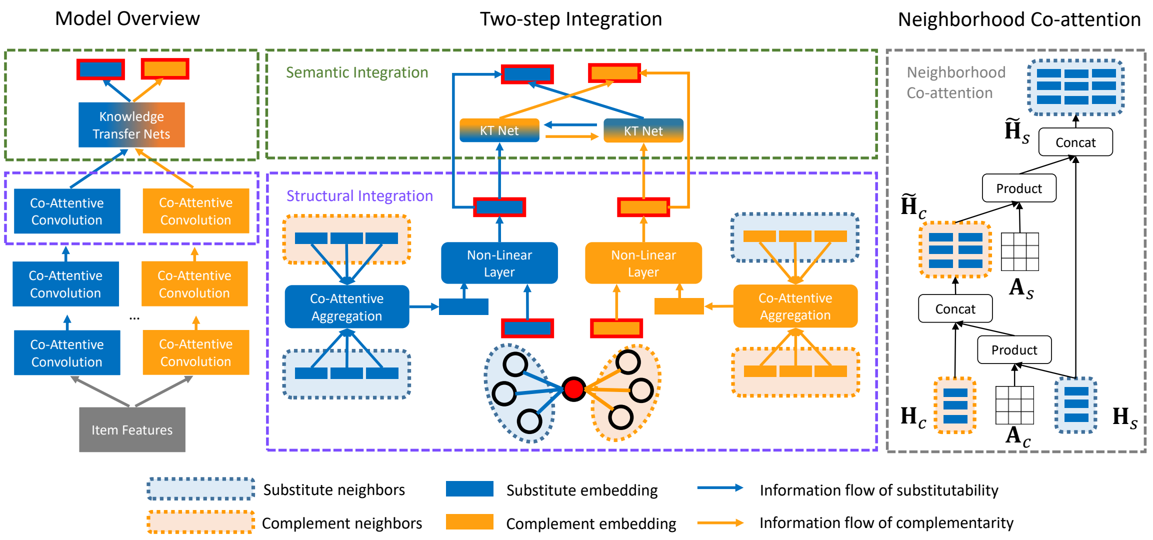


Figure 1: The model architecture. The left part shows the overall framework of our proposed method, where co-attentive convolutions are stacked to aggregate item substitutability and complementarity from local neighborhoods, followed by knowledge transfer networks to consider their mutual influences. The middle part shows the detailed implementations of structural integration and semantic integration. The right part shows how the co-attention between neighborhoods is computed.

additional subgraph $\mathcal{G}^{(c)}$ (or $\mathcal{G}^{(s)}$). Thus, we reformulate Eq. (3) as

$$\tilde{Z}^{(s)} = \text{GCN}^{(s)}(\mathcal{G}^{(s)}|\mathcal{G}^{(c)}, \mathcal{X}) \tag{4}$$

$$\tilde{Z}^{(c)} = \text{GCN}^{(c)}(\mathcal{G}^{(c)}|\mathcal{G}^{(s)}, \mathcal{X}) \tag{5}$$

where $\tilde{Z}^{(s)}$ and $\tilde{Z}^{(c)}$ represent the item embeddings after introducing the structural integration. Here, we use the notation $\mathcal{G}^{(s)}|\mathcal{G}^{(c)}$ to denote the different exploitation of $\mathcal{G}^{(s)}$ in the sub-GCN when $\mathcal{G}^{(c)}$ is also observed. By doing this, we expect the extra subgraph structure integrated in each sub-GCN to provide auxiliary information for better modeling the item semantics.

Semantic integration. To consider the mutual influence between $\mathbf{z}_i^{(s)}$ and $\mathbf{z}_i^{(c)}$, we define an integration function $\eta^{(s)}$ and $\eta^{(c)}$ that learns to refine different item semantics as

$$\hat{Z}^{(s)} = \eta^{(s)}(\tilde{Z}^{(s)}, \tilde{Z}^{(c)}) \quad \text{and} \quad \hat{Z}^{(c)} = \eta^{(c)}(\tilde{Z}^{(c)}, \tilde{Z}^{(s)}) \tag{6}$$

where $\hat{Z}^{(s)}$ and $\hat{Z}^{(c)}$ represent the refined representations. This allows useful knowledge to be extracted and transfer between different item semantics.

In the following subsections, we introduce the implementations of the structural and semantic integration schemes in detail.

4.3 Structural Integration: Multifaceted Co-Attentive Neighborhood Aggregation

The basic intuition of structural integration is that the item relationships represented by different subgraphs may have inherent influence to each other. To incorporate this intuition into the framework of GCN, we propose a *co-attentive neighborhood aggregation* strategy in each convolutional layer, which is inspired by Dynamic Co-attention Network (DCN) [28]. It allows the convlutional layer to simultaneously perceive the local structures of both substitute and complement subgraphs. Next, we show the proposed aggregation strategy in $\tilde{Z}^{(s)} = \text{GCN}^{(s)}(\mathcal{G}^{(s)}|\mathcal{G}^{(c)}, \mathcal{X})$ as an example, and $\tilde{Z}^{(c)} = \text{GCN}^{(c)}(\mathcal{G}^{(c)}|\mathcal{G}^{(s)}, \mathcal{X})$ can be computed similarly.

Definitions of neighbor embeddings. We first denote the substitute and complement neighbors of v_i as $\mathcal{N}_i^{(s)}$ and $\mathcal{N}_i^{(c)}$, respectively. In each convolutional layer, we define the stacked substitute neighbor embeddings as two matrix $\mathbf{H}_s^{(s)} \in \mathbb{R}^{|\mathcal{N}_i^{(s)}|\times d}$ and $\mathbf{H}_c^{(s)} \in \mathbb{R}^{|\mathcal{N}_i^{(c)}|\times d}$, respectively. Note that the neighbor embedding vectors (i.e., the row vectors) in both matrices are substitute embeddings, which represent their substitutability. In other words, we attempt to aggregate *the same type of item semantics* from *different types of neighborhoods*. In the following, we omit the superscripts of $\mathbf{H}_s^{(s)}$ and $\mathbf{H}_c^{(s)}$, and use $\mathbf{H}_s$ and $\mathbf{H}_c$ instead, respectively.

The co-attention process. The co-attention between $\mathbf{H}_s$ and $\mathbf{H}_c$ is an alternative updating process, as depicted in the right part of Figure 1, which contains three steps:

- In the first step, we compute the attention scores of $\mathcal{N}_i^{(c)}$ over $\mathcal{N}_i^{(s)}$ (denoted as $\mathbf{A}_c$) and $\mathcal{N}_i^{(s)}$ over $\mathcal{N}_i^{(c)}$ (denoted as $\mathbf{A}_s$) respectively as

$$\mathbf{A}_c = \text{softmax}(\mathbf{H}_c\mathbf{H}_s^\top), \quad \text{and} \quad \mathbf{A}_s = \text{softmax}(\mathbf{H}_s\mathbf{H}_c^\top). \tag{7}$$

- In the second step, by leveraging $\mathbf{A}_c$, we are able to integrate the influence of $\mathbf{H}_s$ with $\mathbf{H}_c$, to formulate the attended representations of $\mathcal{N}_i^{(c)}$ (denoted as $\hat{\mathbf{H}}_c$) as

$$\hat{\mathbf{H}}_c = [\mathbf{H}_c; \mathbf{A}_c\mathbf{H}_s] \tag{8}$$

  Here, [;] represents row-wise concatenation of two matrices.

- In the thrid step, we integrate $\hat{\mathbf{H}}_c$ back to $\mathbf{H}_s$ to enrich the embeddings of the neighbors in $\mathcal{N}_i^{(s)}$ as

$$\tilde{\mathbf{H}}_s = [\mathbf{H}_s; \mathbf{A}_s\tilde{\mathbf{H}}_c] \tag{9}$$

Finally, the aggregated neighborhood representation can be computed using mean-pooling as $\mathbf{h}_{\mathcal{N}_i} = mean\_pooling(\tilde{\mathbf{H}}_s)$, which operates over the row vectors of $\tilde{\mathbf{H}}_s$.

Compared to the aggregate method (Equation 2) in GCN, the co-attention scheme in our methhod allows the embeddings of $\mathcal{N}_i^{(s)}$ (i.e., $\tilde{\mathbf{H}}_s$) to also carry the information of $\mathcal{N}_i^{(c)}$. We incorporate the co-attention scheme into each convolutional layers of GCN$^{(s)}$, such that the final output $\tilde{\mathbf{z}}_i^{(s)}$ and would contain high-level information of both types of its local subgraph structures.

4.4 Semantic Integration: Knowledge Transfer between Decoupled Node Representations

The structural integration allows each type of item semantics to be aggregated on different subgraph structures. However, the decoupled item embeddings $\tilde{\mathbf{z}}_i^{(s)}$ and $\tilde{\mathbf{z}}_i^{(c)}$ are still represented in two different latent spaces, where their mutual influence might be critical but still absent in the current model. To this end, we further propose a semantic integration scheme, which enables useful semantic information to be transferred across the two embeddings.

Direct transfer of item semantics. Concretely, we design a simple yet effective knowledge transfer scheme to refine the two representations $\tilde{\mathbf{z}}_i^{(s)}$ and $\tilde{\mathbf{z}}_i^{(c)}$. To integrate the knowledge from $\tilde{\mathbf{z}}_i^{(c)}$ into $\tilde{\mathbf{z}}_i^{(s)}$, we leverage a knowledge extraction neural network, which is denoted as $f_{c\to s}(\cdot)$. We use $f_{c\to s}(\cdot)$ to extract transferable information from $\tilde{\mathbf{z}}_i^{(c)}$, and apply it to $\tilde{\mathbf{z}}_i^{(s)}$ as

$$\hat{\mathbf{z}}_i'^{(s)} = (1-\alpha')\tilde{\mathbf{z}}_i^{(s)} + \alpha' f_{c\to s}(\tilde{\mathbf{z}}_i^{(c)}) \tag{10}$$

where $\hat{\mathbf{z}}_i'^{(s)}$ represents the augmented substitute embedding of v_i and α' represents the weight of the integration. Similarly, we can integrate the knowledge from $\tilde{\mathbf{z}}_i^{(s)}$ into $\tilde{\mathbf{z}}_i^{(c)}$, using another knowledge extraction network $f_{s\to c}$, as

$$\hat{\mathbf{z}}_i'^{(c)} = (1-\alpha')\tilde{\mathbf{z}}_i^{(c)} + \alpha' f_{s\to c}(\tilde{\mathbf{z}}_i^{(s)}) \tag{11}$$

Back transfer. In addition, inspired by Dual Learning [11] and CycleGAN [40], we further consider the "back transfer" of the item semantics. We assume that the embedding augmentation $\hat{\mathbf{z}}_i'^{(c)}$ can be transferred back to further improve the quality of $\hat{\mathbf{z}}_i'^{(s)}$. To this end, we extend Eq. (10) and Eq. (11) as

$$\hat{\mathbf{z}}_i^{(s)} = (1-\alpha-\beta)\tilde{\mathbf{z}}_i^{(s)} + \alpha f_{c\to s}(\tilde{\mathbf{z}}_i^{(c)}) + \beta f_{c\to s}(\hat{\mathbf{z}}_i'^{(c)}) \tag{12}$$

$$\hat{\mathbf{z}}_i^{(c)} = (1-\alpha-\beta)\tilde{\mathbf{z}}_i^{(c)} + \alpha f_{s\to c}(\tilde{\mathbf{z}}_i^{(s)}) + \beta f_{s\to c}(\hat{\mathbf{z}}_i'^{(s)}) \tag{13}$$

where α and β are scalars for the combination. Alternatively, we can interpret the "back transfer" as a cycle consistency constrain [40] posed on $f_{s\to c}$ and $f_{c\to s}$, forcing them to be inverse of each other.

It is worth noting that, compared with direct fusion of multiple item semantics, e.g., $\tilde{\mathbf{z}}_i^{(s)} + \tilde{\mathbf{z}}_i^{(c)}$, our semantic integration scheme leverages additional modules (i.e., $f_{s\to c}$ and $f_{c\to s}$) to accomplish knowledge transfer, which has lower risk at undermining the decoupling between different item semantics.

4.5 Model Optimization & Prediction

Optimization. To train DecGCN model, we apply a commonly-used graph-based loss for each sub-GCN, which encourages nearby nodes to have similar representations. For optimizing $\mathbf{z}_i$ in each sub-GCN, the loss can be formally defined as

$$\mathcal{J}(\mathbf{z}_i) = -\log(\sigma(\mathbf{z}_i^\top \mathbf{z}_j)) - Q \cdot \mathbb{E}_{j'\sim P_n(i)}[\log(\sigma(-\mathbf{z}_i^\top \mathbf{z}_{j'}))] \tag{14}$$

where j is a sampled nearby node (e.g., neighbor), $P_n(i)$ is a negative sampling distribution over $\mathcal{V}$, Q is the number of negative samples, and σ is the sigmoid function. The final loss function of DecGCN can be formulated as a multi-task loss:

$$\mathcal{J} = \frac{1}{n}\mathcal{J}(\hat{\mathbf{z}}_i^{(s)}, \hat{\mathbf{z}}_i^{(c)}) = \mathcal{J}(\hat{\mathbf{z}}_i^{(s)}) + \mathcal{J}(\hat{\mathbf{z}}_i^{(c)}) \tag{15}$$

where n is the number of nodes, $\mathcal{J}(\hat{\mathbf{z}}_i^{(s)}, \hat{\mathbf{z}}_i^{(c)})$ is the loss function for node v_i, $\mathcal{J}(\hat{\mathbf{z}}_i^{(s)})$ and $\mathcal{J}(\hat{\mathbf{z}}_i^{(c)})$ are computed as Eq. (14).

Prediction. After the optimization, we can employ the learned embeddings of items via DecGCN to infer substitutes and complements. The prediction score of v_i and v_j being substitutable or complementary can be respectively computed as $p^{(s)}(\hat{\mathbf{z}}_i^{(s)}, \hat{\mathbf{z}}_j^{(s)}) = \sigma(\hat{\mathbf{z}}_i^{(s)}\hat{\mathbf{z}}_j^{(s)\top})$ and $p^{(c)}(\hat{\mathbf{z}}_i^{(c)}, \hat{\mathbf{z}}_j^{(c)}) = \sigma(\hat{\mathbf{z}}_i^{(c)}\hat{\mathbf{z}}_j^{(c)\top})$.

Table 1: Statistics of the datasets.

	Amazon			JD
	Beauty	Clothing	Electronics	
Items	114,792	54,311	115,617	4,046,866
Total edges	5,079,630	889,972	3,641,208	136,052,828
Sub. edges	2,932,446	596,722	1,452,188	84,409,430
Com. edges	2,147,184	293,250	2,189,020	51,643,398

5 EXPERIMENTS

In this section, we conduct experiments, and anticipate the experimental results to answer the following research questions:

- **RQ1**: Does DecGCN outperforms state-of-the-art models on all the datasets? How much is the improvement?
- **RQ2**: What is the impact of the structural integration and semantic integration in our method?
- **RQ3**: What are the effects of the key hyperparameters of our model on the performance?
- **RQ4**: Can we draw any insight or interpretations from our model?
- **RQ5**: Is DecGCN effective to be deployed in real-world scenarios?

5.1 Experimental Settings

Datasets. Our experiments are conducted on three public datasets collected from Amazon: Beauty, Clothing and Electronics, which are available publicly and have been widely used in literature [17, 18, 34]. We follow previous work [17] to construct substitute and complement subgraphs by considering the "also-viewed" and "also-bought" as substitute and complement relationships, respectively. For each item, we use its categories, and brand as features. Moreover, we further include a large-scale industrial dataset, which is collected from JD.com. We also include shop and product keywords as features for the JD data. Details of the data are listed in Table 1.

Baselines. We compare our framework with following state-of-the-art methods:

- **GraphSage** [9]. This is a widely-used inductive GCN based model [29]. We follow the idea of PinSage [29] and Sceptre [17], which learns a shared topic distribution for inferring substitutes and complements. In particular, we use a single GraphSage to model the two types of edges, and output a shared latent embedding for each item. Its convolution can be redefined as

$$\mathbf{h}_i^l = \sigma(\mathbf{W}^l \cdot \text{CONCAT}(\mathbf{h}_i^{l-1}, \mathbf{h}^l_{\mathcal{N}_i^{(s)}}, \mathbf{h}^l_{\mathcal{N}_i^{(c)}})) \tag{16}$$

Separated non-linear layers are applied on the shared embedding to finalize the two representations for each item.

- **PolyGCN & PolyGAT** [14]. PolyGCN uses two separated GCNs (i.e., GraphSage) for modeling substitutes and complements, as indicated by Eq. (3). We also include PolyGAT as a baseline, where Graph Attention Network (GAT) [21] is used as the base model.
- **FuseGCN**. This is a simple variant of PolyGCN model, which has two separated models to produce two sets of item embeddings. However, each GCN of FuseGCN take two subgraphs as input, where the convolution operation is defined as Eq. (16).
- **HAN** [25]. This is a state-of-the-art GNN method for heterogeneous graph embedding. Similar to FuseGCN, we have two separated HAN models that take two subgraphs as input, and output substitutable and complementary embeddings of items.
- **DisenGCN** [15]. This model is originally proposed to disentangle node semantics for homogeneous graph. In our case, we explicitly consider the substitutability and complementarity as two disentangled semantics.
- **GATNE-I** [1]. This model is a state-of-the-art method for modeling attributed multiplex heterogeneous network. For each node, it leverages separate GCNs to learn multiple node embeddings w.r.t. different types of relations.
- **DecGCN**. This is our proposed method. We use **DecGCN/SE** and **DecGCN/ST** to represent the degenerated DecGCN without semantic and structural integration, respectively.

We conduct **link prediction** to validate the effectiveness of the models. For each node (i.e., item), we randomly sample one edge for each type of relationships (i.e., substitute and complement) to construct the test data, and use the rest as the training data. After training the models, we report their performance for link prediction task on the test data. For each item, as previous did [17], we fuse its ground truth with N (*i.e.,* 1000) sampled negative instances (i.e., items) and rank them using different models. The performances are evaluated by three metrics: MRR@K, HR@K and NDCG@K, for each of which the average value over all items is reported.

5.2 Implementation Details

We implement all the methods with Euler[1], which is a GNN library based on Tensorflow[2]. To deploy our model for online candidate generation, we first compute and store the embedding vectors of all candidate items. For the offline experiments, all the models can be trained within few hours on four NVIDIA Tesla P40 GPUs. To deploy the model for online candidate generation, we use Faiss[3] to efficiently retrieve substitutable or complementary items based on approximate nearest neighbor search w.r.t. the prediction score.

Parameter settings. We set the dimension of each feature embedding as 16, and the item embedding as 128 for all the convolutional layers. For the GCN-based methods, we stack two convolutional layers, each of which aggregates 5 sampled neighbors for each type of edge. In DecGCN, we use two 3-layer Multi-Layer Perceptrons (MLPs) as $f_{s\to c}$ and $f_{c\to s}$, where ReLU functions are used after the first two layers. For the model optimization, we set the learning rate as 1e-4 and batch size as 512 for all the methods, and train the models for 20 epochs using Adam optimizer on all datasets. More details can be found in the released codes.

[1]https://github.com/alibaba/euler
[2]https://www.tensorflow.org/
[3]https://ai.facebook.com/tools/faiss/

Table 2: Performance comparison on Amazon Beauty.

Beauty	Substitute			Complement		
	MRR@10	HR@10	NDCG@10	MRR@10	HR@10	NDCG@10
GraphSage	0.225	0.515	0.340	0.238	0.550	0.364
PolyGCN	0.357	0.664	0.487	0.368	0.642	0.490
FuseGCN	0.310	0.623	0.445	0.360	0.681	0.506
PolyGAT	0.330	0.689	0.495	0.280	0.671	0.450
HAN	0.390	0.693	0.531	0.368	0.704	0.534
DisenGCN	0.421	0.700	0.554	0.363	0.637	0.491
GATNE-I	0.426	0.729	0.569	0.439	0.691	0.566
DecGCN/SE	0.410	0.702	0.540	0.416	0.684	0.543
DecGCN/ST	0.405	0.712	0.548	0.394	0.670	0.526
DecGCN	**0.459***	**0.747***	**0.593***	**0.471***	**0.716***	**0.593***

Table 3: Performance comparison on Amazon Clothing.

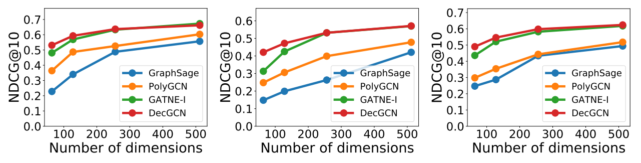
Clothing	Substitute			Complement		
	MRR@10	HR@10	NDCG@10	MRR@10	HR@10	NDCG@10
GraphSage	0.126	0.318	0.199	0.149	0.368	0.234
PolyGCN	0.202	0.462	0.306	0.195	0.429	0.290
FuseGCN	0.212	0.471	0.316	0.226	0.496	0.336
PolyGAT	0.279	0.555	0.397	0.257	0.513	0.366
HAN	0.248	0.526	0.363	0.243	0.499	0.352
DisenGCN	0.330	0.618	0.459	0.248	0.517	0.363
GATNE-I	0.302	0.583	0.425	0.298	0.556	0.411
DecGCN/SE	0.300	0.587	0.425	0.285	0.549	0.400
DecGCN/ST	0.294	0.578	0.417	0.263	0.526	0.375
DecGCN	**0.342***	**0.631***	**0.472***	**0.327***	**0.600***	**0.450***

Table 4: Performance comparison on Amazon Electronics.

Electronics	Substitute			Complement		
	MRR@10	HR@10	NDCG@10	MRR@10	HR@10	NDCG@10
GraphSage	0.184	0.451	0.287	0.188	0.424	0.281
PolyGCN	0.231	0.538	0.354	0.219	0.459	0.317
FuseGCN	0.238	0.542	0.360	0.240	0.506	0.351
PolyGAT	0.288	0.565	0.407	0.239	0.462	0.334
HAN	0.270	0.547	0.387	0.230	0.459	0.327
DisenGCN	0.287	0.587	0.416	0.221	0.456	0.318
GATNE-I	0.379	0.692	0.521	0.296	0.555	0.409
DecGCN/SE	0.354	0.674	0.497	0.284	0.541	0.396
DecGCN/ST	0.335	0.659	0.478	0.267	0.524	0.377
DecGCN	**0.400***	**0.713***	**0.546***	**0.311***	**0.583***	**0.429***

5.3 Overall Performance (RQ1)

The experimental comparisons of DecGCN with its variants and the baselines are presented in Tables 2, 3 and 4, where all the experimental results are obtained by an average of 5 repeat runs and "$*$" indicates the statistically significant improvements (i.e., two-sided t-test with $p < 0.01$) over the best baseline. Tables show the overall performance of different methods on the three Amazon datasets, respectively. In the tables, the **boldfaced** and underlined values represent the best and the second best performance among the compared methods. From the results, we observe that our proposed



(a) Substitute (B) (b) Substitute (C) (c) Substitute (E)

Figure 2: Varying the number of latent dimensions (i.e., d).

Table 5: Varying the size of sampled neighbors in DecGCN.

		Substitutes		Complements	
Dataset	Number of neighbors	HR@10	NDCG@10	HR@10	NDCG@10
B	(5)	0.703	0.550	0.677	0.544
	(5, 5)	0.747	0.593	0.716	0.593
	(5, 10)	0.765	0.619	**0.727**	0.605
	(10, 5)	**0.772**	**0.627**	**0.728**	**0.610**
C	(5)	0.550	0.393	0.519	0.368
	(5, 5)	0.631	0.472	0.600	0.450
	(5, 10)	0.620	0.455	0.599	0.436
	(10, 5)	**0.644**	**0.490**	**0.617**	**0.465**
E	(5)	0.632	0.454	0.486	0.339
	(5, 5)	0.713	0.546	0.583	0.429
	(5, 10)	0.722	**0.560**	0.572	0.422
	(10, 5)	**0.724**	**0.562**	**0.600**	**0.453**

DecGCN significantly outperforms all the baseline methods, w.r.t. to all the evaluation metrics. For example, on the task of inferring substitutable items, DecGCN achieves better performance than the GATNE-I by at least 5.5% on all the datasets w.r.t. MRR@10; It also outperforms the state-of-the-art heterogeneous GNN method, i.e., HAN, by 7.7%, 19.9% and 30.3% on the three datasets, respectively, on the HR@10 metric. The results for inferring complementary items are qualitatively similar. This indicates that DecGCN can better identify substitutes and complements w.r.t. query items.

5.4 Ablation Study (RQ2)

Tables 2, 3 and 4 also show the comparison results among different variants of DecGCN, i.e., DecGCN/SE, DecGCN/ST and DecGCN, which depict the effectiveness of our proposed structural and semantic integration paradigms. First, we can see from the tables that both DecGCN/SE and DecGCN/ST can beat PolyGCN by a large margin. For example, on Electronics dataset, the relative improvements of DecGCN/ST over PolyGCN are at least 22.4% and 14.2% on all metrics, for inferring substitutes and complements, respectively. This is because that PolyGCN models substitute and complement subgraphs separately, while DecGCN/SE, DecGCN/ST try to carefully integrate them for modeling each type of the subgraphs. Therefore, we can conclude that the structural and semantic integration can better leverage the substitute and complement information, i.e., which allows them to facilitate the modeling of both semantics. Moreover, the results also reveal that DecGCN has better performance than both DecGCN/SE and DecGCN/ST, which shows that the two integration schemes can work as a whole to bring superiority to the DecGCN model.

5.5 Hyperparameter Analysis (RQ3)

We further conduct experiments while varying the key hyperparameters of DecGCN. First, we vary the number of latent dimensions (i.e., d) for item embeddings from 32 to 512. Figure 2 illustrates the results for inferring substitutes on all the three datasets w.r.t. NDCG@10. The results for inferring complements are similar and thus omitted. We can see that the performance of different methods increases when setting larger d. For example, on all the datasets, the NDCG@10 values of DecGCN increase by more than 25.0% for inferring substitutes, when adjusting d from 64 to 512. Moreover, we can find out that our DecGCN method consistently outperforms PolyGCN and GraphSage. For example, on the task of inferring substitutes, it is able to achieve higher NDCG@10 values than PolyGCN by over 9.8%, 19.6% and 20.4% on the three datasets, respectively. The improvements of DecGCN over GATNE-I is relatively small, especially when $d = 256$ and $d = 512$.

Table 6: Performance comparison on JD.com.

JD.com	Substitute			Complement		
	PolyGCN	GATNE-I	DecGCN	PolyGCN	GATNE-I	DecGCN
MRR@10	0.461	0.535	**0.553**	0.526	0.566	**0.569**
HR@10	0.842	0.894	**0.902**	0.894	0.892	**0.921**
NDCG@10	0.633	0.716	**0.733**	0.712	0.724	**0.746**

We also investigated the impact of the neighborhood size in DecGCN. In particular, we use DecGCNs with different number of neighbors, denoted as (5), (5, 5), (5, 10), (10, 5). For example, (5, 10) means to aggregate 10 two-hop neighbors and 5 one-hop neighbors in the model. We use B, C and E to denote Beauty, Clothing and Electronics datasets, respectively. The results in Table 5 show that that 1) one-layer DecGCN has the worst performance (worse than (5,5) by 5.4%, 12.8% and 11.4% on the three datasets, respectively, w.r.t. both HR@10 and NDCG@10), as it only aggregates the information of one-hop neighbors for each item; 2) the DecGCN models with (5, 10) and (10, 5) are better than (5,5), which indicate that including more neighbors can provide more information for the modeling; 3) the DecGCN with (10, 5) has the best performance, especially better than (5, 10), which indicates that the one-hop neighbors are more important than the two-hop neighbors in the model.

5.6 Analysis of Model Components (RQ4)

We conduct in-depth analysis of model components in DecGCN, which aim at providing more useful insights of the model. We conclude our key findings: **1) Co-attention vs. self-attention**. We study two variants of structural integration schemes, i.e., co-attention or self-attention. Using self-attention in structural integration means the importance of each neighbor is derived by those neighbors with the same type. Experiments demonstrate that co-attention can achieve more than 3% relative improvements over self-attention on all the datasets w.r.t. to both HR@10 and NDCG@10. This tells us that the attention across different types of neighbors can provide more useful information for the model. **2) Variants of semantic integration**. We investigate two variants of our semantic integration paradigm, i.e., SE-forward and SE-cycle, Here, SE-forward means we only consider the forward transfer in the semantic integration process, as formulated in Eq. (10) and Eq. (11), and SE-cycle is to consider both forward transfer and backward transfer (as shown in Eq. (12) and Eq. (13)). The results show that SE-cycle is able to achieve slightly better performance (over 1.2% and 2.4% on HR@10 and NDCG@10) than SE-forward for inferring substitutes, while no significant improvements are observed for inferring complements.

5.7 Offline and Online A/B Testing (RQ5)

For the offline A/B testing, we compare PolyGCN, GATNE-I and DecGCN on the large-scale industrial JD.com dataset. As shown in Table 1, the JD.com dataset contains over 4 million items, 84 million substitute relations and 51 million complement relations. The parameters are set the same as in Section 5.1. The results in Table 6 show that DecGCN outperform the two baselines on both inference tasks. Compared with GATNE-I, DecGCN is able to achieve better performance by 0.9%–5.6% and 0.5%–6.1% for inferring substitutes and complements, respectively, w.r.t. all evaluation metrics.

For the online A/B testing, we deploy DecGCN and the strongest baseline GATNE-I in the Candidate Generation module in the online Recommender System in JD.com for one month (from June 2020 to July 2020). For each request, we use the model to generate candidates, and merge them with other candidate generation sources for re-ranking. Online experiments show that our method DecGCN significantly outperforms the strongest baseline method GATNE-I by 3.6% ($p\text{-value} < 0.01$) in Click-Through Rate. In addition, we observe 0.3% improvement on browsing depth (i.e., the depth of scrolling recommendation lists).

6 CONCLUSION

In this paper, we propose an effective decoupled Graph Convolutional Network for the task of inferring substitutable and complementary items. The propose DecGCN is able to learn item substitutability and complementarity as separated embeddings vectors, where mutual influence between different graph structures and item semantics are further captured. Experiments on three public datasets and A/B testing on a real-world industrial recommender system demonstrate the remarkable performance of our solution.

REFERENCES

[1] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation learning for attributed multiplex heterogeneous network. In *KDD'19*. 1358–1368.

[2] Weijian Chen, Yulong Gu, Zhaochun Ren, Xiangnan He, Hongtao Xie, Tong Guo, Dawei Yin, and Yongdong Zhang. 2019. Semi-supervised user profiling with heterogeneous graph attention networks. In *IJCAI'19*.

[3] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD'17*. 135–144.

[4] Alessandro Epasto and Bryan Perozzi. 2019. Is a single embedding enough? learning node representations that capture multiple social contexts. In *WWW'19*. 394–404.

[5] Shaohua Fan, Junxiong Zhu, Xiaotian Han, Chuan Shi, Linmei Hu, Biyu Ma, and Yongliang Li. 2019. Metapath-guided Heterogeneous Graph Neural Network for Intent Recommendation. In *KDD'19*. 2478–2486.

[6] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2018. Large-scale learnable graph convolutional networks. In *KDD'18*. 1416–1424.

[7] Yulong Gu, Zhuoye Ding, Shuaiqiang Wang, and Dawei Yin. 2020. Hierarchical User Profiling for E-commerce Recommender Systems. In *WSDM'20*.

[8] Yulong Gu, Zhuoye Ding, Shuaiqiang Wang, Lixin Zou, Yiding Liu, and Dawei Yin. 2020. Deep Multifaceted Transformers for Multi-objective Ranking inLarge-Scale E-commerce Recommender Systems. In *CIKM'20*.

[9] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS'17*. 1024–1034.

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

[11] Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *NeurIPS'16*. 820–828.

[12] Wang-Cheng Kang, Eric Kim, Jure Leskovec, Charles Rosenberg, and Julian McAuley. 2019. Complete the Look: Scene-based Complementary Product Recommendation. In *CVPR'19*. 10532–10541.

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

[14] Ninghao Liu, Qiaoyu Tan, Yuening Li, Hongxia Yang, Jingren Zhou, and Xia Hu. 2019. Is a single vector enough? exploring node polysemy for network embedding. In *KDD'19*. 932–940.

[15] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. 2019. Disentangled graph convolutional networks. In *ICML'19*. 4212–4221.

[16] Yao Ma, Zhaochun Ren, Ziheng Jiang, Jiliang Tang, and Dawei Yin. 2018. Multi-dimensional network embedding with hierarchical structure. In *WSDM'18*. 387–395.

[17] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *SIGKDD*. 785–794.

[18] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR'15*. 43–52.

[19] Vineeth Rakesh, Suhang Wang, Kai Shu, and Huan Liu. 2019. Linked variational autoencoders for inferring substitutable and supplementary items. In *WSDM'19*. 438–446.

[20] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S Yu Philip. 2018. Heterogeneous information network embedding for recommendation. *IEEE TKDE* 31, 2 (2018), 357–370.

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

[22] Lucas Vinh Tran, Tuan-Anh Nguyen Pham, Yi Tay, Yiding Liu, Gao Cong, and Xiaoli Li. 2019. Interact and decide: Medley of sub-attention networks for effective group recommendation. In *SIGIR'19*. 255–264.

[23] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *KDD'19*. 968–977.

[24] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *KDD'18*. 839–848.

[25] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW'19*. 2022–2032.

[26] Zihan Wang, Ziheng Jiang, Zhaochun Ren, Jiliang Tang, and Dawei Yin. 2018. A path-constrained framework for discriminating substitutable and complementary products in e-commerce. In *WSDM'18*. 619–627.

[27] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[28] Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604* (2016).

[29] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *KDD'18*. 974–983.

[30] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph Transformer Networks. In *NeurIPS'19*. 11960–11970.

[31] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *KDD'19*. 793–803.

[32] Mingyue Zhang, Xuan Wei, Xunhua Guo, Guoqing Chen, and Qiang Wei. 2019. Identifying Complements and Substitutes of Products: A Neural Network Framework Based on Product Embedding. *TKDD'19* 13, 3 (2019), 1–29.

[33] Shijie Zhang, Hongzhi Yin, Qinyong Wang, Tong Chen, Hongxu Chen, and Quoc Viet Hung Nguyen. 2019. Inferring substitutable products with deep network embedding. *IJCAI'19* (2019), 4306–4312.

[34] Yin Zhang, Haokai Lu, Wei Niu, and James Caverlee. 2018. Quality-aware neural complementary item recommendation. In *RecSys'18*. 77–85.

[35] Tong Zhao, Julian McAuley, Mengya Li, and Irwin King. 2017. Improving recommendation accuracy using networks of substitutable and complementary products. In *IJCNN'17*. 3649–3655.

[36] Xiangyu Zhao, Changsheng Gu, Haoshenglun Zhang, Xiaobing Liu, Xiwang Yang, and Jiliang Tang. 2019. Deep Reinforcement Learning for Online Advertising in Recommender Systems. *arXiv preprint arXiv:1909.03602* (2019).

[37] Xiangyu Zhao, Long Xia, Jiliang Tang, and Dawei Yin. 2019. Deep reinforcement learning for search, recommendation, and online advertising: a survey. *SIGWEB'19* Spring (2019), 1–15.

[38] Xiangyu Zhao, Xudong Zheng, Xiwang Yang, Xiaobing Liu, and Jiliang Tang. 2020. Jointly Learning to Recommend and Advertise. *arXiv preprint arXiv:2003.00097* (2020).

[39] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2018. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434* (2018).

[40] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV'17*. 2223–2232.

[41] Lixin Zou, Long Xia, Yulong Gu, Xiangyu Zhao, Weidong Liu, Jimmy Huang, and Dawei Yin. 2020. Neural Interactive Collaborative Filtering. In *SIGIR'20*.